Universidade Virtual Africana

INFORMÁTICA APLICADA: CSI 3302

PROGRAMAÇÃO ORIENTADA À OBJECTOS

Célio Barbosa Sengo

Prefácio

A Universidade Virtual Africana (AVU) orgulha-se de participar do aumento do acesso à educação nos países africanos através da produção de materiais de aprendizagem de qualidade. Também estamos orgulhosos de contribuir com o conhecimento global, pois nossos Recursos Educacionais Abertos são acessados principalmente de fora do continente africano.

Este módulo foi desenvolvido como parte de um diploma e programa de graduação em Ciências da Computação Aplicada, em colaboração com 18 instituições parceiras africanas de 16 países. Um total de 156 módulos foram desenvolvidos ou traduzidos para garantir disponibilidade em inglês, francês e português. Esses módulos também foram disponibilizados como recursos de educação aberta (OER) em oer.avu.org.

Em nome da Universidade Virtual Africana e nosso patrono, nossas instituições parceiras, o Banco Africano de Desenvolvimento, convido você a usar este módulo em sua instituição, para sua própria educação, compartilhá-lo o mais amplamente possível e participar ativamente da AVU Comunidades de prática de seu interesse. Estamos empenhados em estar na linha de frente do desenvolvimento e compartilhamento de recursos educacionais abertos.

A Universidade Virtual Africana (UVA) é uma Organização Pan-Africana Intergovernamental criada por carta com o mandato de aumentar significativamente o acesso a educação e treinamento superior de qualidade através do uso inovador de tecnologias de comunicação de informação. Uma Carta, que estabelece a UVA como Organização Intergovernamental, foi assinada até agora por dezenove (19) Governos Africanos - Quênia, Senegal, Mauritânia, Mali, Costa do Marfim, Tanzânia, Moçambique, República Democrática do Congo, Benin, Gana, República da Guiné, Burkina Faso, Níger, Sudão do Sul, Sudão, Gâmbia, Guiné-Bissau, Etiópia e Cabo Verde.

As seguintes instituições participaram do Programa de Informática Aplicada: (1) Université d'Abomey Calavi em Benin; (2) Université de Ougagadougou em Burkina Faso; (3) Université Lumière de Bujumbura no Burundi; (4) Universidade de Douala nos Camarões; (5) Universidade de Nouakchott na Mauritânia; (6) Université Gaston Berger no Senegal; (7) Universidade das Ciências, Técnicas e Tecnologias de Bamako no Mali (8) Instituto de Administração e Administração Pública do Gana; (9) Universidade de Ciência e Tecnologia Kwame Nkrumah em Gana; (10) Universidade Kenyatta no Quênia; (11) Universidade Egerton no Quênia; (12) Universidade de Addis Abeba na Etiópia (13) Universidade do Ruanda; (14) Universidade de Dar es Salaam na Tanzânia; (15) Universite Abdou Moumouni de Niamey no Níger; (16) Université Cheikh Anta Diop no Senegal; (17) Universidade Pedagógica em Moçambique; E (18) A Universidade da Gâmbia na Gâmbia.

Bakary Diallo

O Reitor

Universidade Virtual Africana

Créditos de Produção

Autor

Célio Barbosa Sengo

Par revisor(a)

Eloy Tavares

UVA - Coordenação Académica

Dr. Marilena Cabral

Coordenador Geral Programa de Informática Aplicada

Prof Tim Mwololo Waema

Coordenador do módulo

Jules Degila

Designers Instrucionais

Elizabeth Mbasu

Benta Ochola

Diana Tuel

Equipa Multimédia

Sidney McGregor

Barry Savala

Edwin Kiprono

Kelvin Muriithi

Victor Oluoch Otieno

Michal Abigael Koyier

Mercy Tabi Ojwang

Josiah Mutsogu

Kefa Murimi

Gerisson Mulongo

Direitos de Autor

Apoiado por

Projeto Multinacional II da UVA financiado pelo Banco Africano de Desenvolvimento.

Índice

Descrição Geral do Curso

Bem-vindo(a) a Programação Orientada a Objectos

Esta disciplina visa dotar os estudantes de conceitos básicos sobre a produção de softwares usando o paradigma orientado a objectos.

Pré-requisitos

- Esta disciplina tem como pré-requesitos a programação estruturada, onde devem ser considerados os seguintes conceitos: variáveis, tipos de dados, operadores, estruturas de controle e arrays.

Materiais

Os materiais necessários para completar este curso incluem:

- JDK 1.8
- Qualquer editor de textos como o Notepad ou Notepad++

Objetivos do Curso

Após concluir este curso, o(a) aluno(a) deve ser capaz de:

- Perceber o paradigma orientado a objecto e diferencia-lo das linguagens estruturadas
- Implementar os conceitos OOP em pequenos projectos de software
- Desenhar e programar aplicações GUI
- Desenhar e desenvolver pequenas aplicações usando uma linguagem OOP

Unidades

Unidade 0: Diagnóstico

Princípios básicos sobre programação estruturadas

Unidade 1: Introdução à programação orientada a objectos (POO)

Neste capítulo será discutido o paradigma OOP, comparando este ao paradigma estruturado. Será também realizada uma revisão aos tipos de dados, estruturas de controle discutidas na programação estruturada.

Unidade 2: Classes, Objectos e membros

Nesta unidade serão discutidos os conceitos básicos sobre OOP como: métodos e dados, construtores, visibilidades, métodos e dados estáticos e herança.

Unidade 3: Pacotes e tratamento de erros

Nesta unidade será discutida a estrutura de pacotes da linguagem java e pacotes definidos pelo programador. Também serão discutidas as formas de tratamento de erros e como gerar streams linput/output)

Unidade 4: Introdução à programação GUI

Nesta unidade serão discutidas as diferentes formas de desenhar aplicações gráficas usando os pacotes applet, swing/awt. Serão também discutidos os eventos básicos sobre os objectos GUI.

Avaliação

Em cada unidade encontram-se incluídos instrumentos de avaliação formativa a fim de verificar o progresso do(a)s aluno(a)s.

No final de cada módulo são apresentados instrumentos de avaliação sumativa, tais como testes e trabalhos finais, que compreendem os conhecimentos e as competências estudadas no módulo.

A implementação dos instrumentos de avaliação sumativa fica ao critério da instituição que oferece o curso. A estratégia de avaliação sugerida é a seguinte:

1	Atrabalhos	10%
2	Avaliação Contínua	20%
3	Laboratório	40%
4	Exame final	30%

Calendarização

Unidade	Temas e Atividades	Estimativa do tempo
1 - Introdução à programação orientada a objectos (POO)	3 Actividades e Exercícios prácticos Actividade 1 (6 horas) Actividade 2 (10 horas) Actividade 3 (10 horas)	26 horas
2 - Classes, Objectos e membros	3 Actividades e Exercícios prácticos Actividade 1 (6 horas) Actividade 2 (6 horas) Actividade 3 (6 horas)	18 horas
3 - Pacotes e tratamento de erros	2 Actividades e Exercícios prácticos Actividade 1 (9 horas) Actividade 2 (9 horas)	18 horas
4 - Introdução à programação GUI	3 Actividades e Exercícios prácticos Actividade 1 (6 horas) Actividade 2 (6 horas) Actividade 3 (6 horas)	18 horas
5 - Laboratórios	Laboratórios Laboratório da Unidade 1 (10 horas) Laboratório da Unidade 2 (10 horas) Laboratório da Unidade 3 (10 horas) Laboratório da Unidade 4 (10 horas)	40 horas

Leituras e outros Recursos

As leituras e outros recursos deste curso são:

Unidade 0

Leituras e outros recursos obrigatórios:

- Collins, W. J. (1988). Programação estruturada com estudos de casos em Pascal. McGraw-Hill.
- Jungthon, G., & Goulart, C. M. (2009). Paradigmas de Programação.Monografia (Monografia)—Faculdade de Informática de Taquara, Rio Grande do Sul, 57.

Recursos opcionais:

- Programação Estruturada. Wikipedia https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_estruturada Visitado aos 13 de Maio de 2016 Este site possui os conceitos básicos das linguagens estruturadas. Poderá ser usado para a consulta das características de uma linguagem do paradigma estruturado, funcional e orientado a objecto.

Unidade 1

Leituras e outros recursos obrigatórios:

- The Computer Revolution/Programming/Object Oriented vs. Structured programming. http://en.wikibooks.org/wiki/The_Computer_Revolution/Programming/Object_Oriented_vs._Structured_programming Visitado aos 16/10/2014 as 23:22
- Oracla Java Documentation The Java ™ Tutorial http://docs.oracle.com/javase/tutorial/java/nutsandbolts/flow.html Site visitado aos 25 de Outubro de 2014 pelas 23:40

Recursos opcionais:

- Java - Módulo II - Aula 17: Comparando entre programação estruturada e POO https://www.youtube.com/watch?v=xb70BswUSRI Visitado aos 13 de Maio de 2016 Neste vídeo poderá perceber as diferenças básicas entre programar no paradigam orientado a objectos e programar no paradigma estruturado. Ainda neste vídeo o autor explica de forma simples e clara os princípios básicos da programação orientada a objectos.

Unidade 2

Leituras e outros recursos obrigatórios:

- Balagurusamy, E. Programming with Java a primer. 4th Edition. Tata McGraw-Hill. 2010
- Buyya, Rajkumar. Object-oriented Programming with Java: Essentials and Applications. 2009

Recursos opcionais:

- Aprenda Java Fácil Aula 08 - Métodos https://www.youtube.com/watch?v=S1xElfHx8C8 Visitado aos 13 de Maio de 2016 Neste video poderá perceber de forma muito simples a técnica de programação de métodos em classes.
- Aprenda Java Fácil Aula10 - Mais Sobre Métodos e Classes https://www.youtube.com/watch?v=pAKkEq4z2ZM Visitado aos 13 de Maio de 2016 Neste video poderá um pouco mais sobre métodos e dados em Java. O autor usa a ferramenta Netbeans para programar contudo, o estudante deve continuar usando um editor de textos para escrever os seus programas tal como foi recomendado nas ferramentas do curso.

Unidade 3

Leituras e outros recursos obrigatórios:

- Harvey M. Deitel. Java: Como Programar. 6 ed. São Paulo: Pearson education do Brasil, 2005
- Naughton, Patrick, Dominando o Java, Guia Autorizado da Sun Microsystems, Editora Makron Books, 1997.

Recursos opcionais:

Como utilizar package (pacotes) em Java https://www.youtube.com/watch?v=7snwqkMb8n0 Visitado aos 13 de Maio de 2016 Neste vídeo são demostrados os passos básicos para a criação de pacotes em java

060 Java nivel basico Exceções tratamento de erros no Java https://www.youtube.com/watch?v=XPOUzj1f1yQ Visitado aos 13 de Maio de 2016 Neste vídeo é demostrado o tratamento de erros em java e foi dado como exemplo a excepção: ArraIndexOutOfBounds.

Unidade 4

Leituras e outros recursos obrigatórios:

- Harvey M. Deitel. Java: Como Programar. 6 ed. São Paulo: Pearson education do Brasil, 2005
- Balagurusamy, E. Programming with Java a primer. 4th Edition. Tata McGraw-Hill. 2010

Recursos opcionais:

- Beginner Java - Intro to Swing (GUI) - Lesson 28 https://www.youtube.com/watch?v=uXitpup0l-s Visitado aos 13 de Maio de 2016 Neste vídeo o autor demostra como criar formulários usando o pacote swing. Este permite perceber o quanto é fácil desenhar formulários do pacote swing.
- Beginner Java - Intro to Swing (GUI) - Lesson 28 https://www.youtube.com/watch?v=2ly7d_uMlrI Visitado aos 13 de Maio de 2016 Este vídeo é a continuação do anterior, onde o autor apresenta de forma muito simples o método de adição de eventos aos objectos, com destaque à interface ActionListener.

Unidade 0. Diagnóstico

Introdução à Unidade

O propósito desta unidade é verificar a compreensão dos conhecimentos que você possui relacionados com este curso.

Objetivos da Unidade

Após a conclusão desta unidade, deverá ser capaz de:

- Perceber os níveis de abstração das linguagens
- Descrever as vantagens das linguagens de alto nível
- Desenhar algorítmos e convertê-los em programas
- Desenhar aplicações usando uma linguagem estruturada

Termos-chave

Programa: Um conjunto de instruções com objectivo de resolver uma tarefa

Algoritmo: Uma sequência de passos a serem seguidos para resolver um determinado problema.

Linguagens de alto nível: Linguagens de programação que usam uma notação muito próxima a linguagem humana.

Estruturas de controle: São comandos que permitem desviar ou repetir a sequência de um programa. São elas: IF, DO, WHILE, FOR, BREAK, CONTINUE e SWITCH.

Matrizes: Váriavel que permite armazenar vários elementos, onde cada um deles é identificado por um índice que inicia de 0 a tamanho-1 onde tamanho é o número total de elementos.

Avaliação

1. O que é um compilador?
2. Escreva um algorítmo para o problema que se seque: O utilizador introduz dois números e o sistema calcula a média aritmética dos dois. Não são aceites números negativos.
3. Escreva um programa que permite somar dois números.
4. Escreva uma lista de linguagens de alto nível (quanto mais linguagens maior é a cotação)
5. Escreva um programa que permite calcular o factorial de um número introduzido pelo utilizador.
6. Programa uma aplicação que permite introduzir o nome de alguém e o sistema imprime o número de vogais encontradas no nome.

Critérios de Avaliação

Cada uma das questões têm a cotação de 3 pontos, sendo que o total é de 24 pontos

Responda

1. Tradução da linguagem humana para a linguagem máquina
2. Espera-se um fluxograma ou pseudo código onde os seguintes elementos são executados:

   entrada de dados para os dois números (verificando se os números são positivos), soma dos dois (pode-se armazenar o resultado numa variável) e a impressão da respectiva soma.
3. Espera-se que o aluno use uma linguagem estruturada qualquer como o C e execute os seguintes passos: Entrada de dois números (quer pelo teclado ou não), e a impressão da soma dos dois.
4. 4C, C++, Java, C#, PHP, Ruby, etc
5. Espera-se que: faça a entrada de um número, crie um loop que inicia de 1 até ao número, e dentro do loop acumule a multiplicação de cada contador pelo próximo numa varável como mostra o exemplo a seguir:

```
int main()

{

int fat, n;

printf("Insira um valor para o qual deseja calcular seu
fatorial: ");

scanf("%d", &n);

for(fat = 1; n > 1; n = n - 1)

fat = fat * n;

printf("\nFatorial calculado: %d", fat);

return 0;

}
```

6 - Proposta de programa

```
 #include<stdio.h>
#include<conio.h>
void main()
{
 char ch[10];
 int i,count=0;
 clrscr();
 printf("Enter the string\n");
 gets(ch);
 for(i=0;ch[i]!='\0';i++)
 {
  if(ch[i]=='a' || ch[i]=='e' || ch[i]=='i' || ch[i]=='o'
    || ch[i]=='u')
  {
   count++;
  }
 }
 printf("Vowels = %d",count);
 getch();
```

Fonte: http://www.cprogrambank.com/cprogst3

Leituras e Outros Recursos

As leituras e outros recursos desta unidade encontram-se na lista de "Leituras e Outros Recursos do curso".

- Programação Estruturada. Wikipedia https://pt.wikipedia.org/wiki Programa%C3%A7%C3%A3o_estruturada Visitado aos 13 de Maio de 2016 Este site possui os conceitos básicos das linguagens estruturadas. Poderá ser usado para a consulta das características de uma linguagem do paradigma estruturado, funcional e orientado a objecto.

Unidade 1. Introdução à programação orientada a objectos

Introdução à Unidade

Hoje em dia existem milhões de aplicações desenvolvidas no paradigma orientado a objecto, como por exemplo os sistemas de gestão de conteúdos, sistemas de gestão, fóruns de discussão, entre outros. As linguagens orientadas a objectos vieram mudar a forma como os programas eram desenvolvidos, dando mais ênfase aos dados (em detrimento das funções) onde estes passam a estar protegidos do acessos externos.

Esta unidade trata de introduzir o conceito de programação orientada a objecto (OOP), dando ênfase aos princípios básicos do paradigma. O conceito de objecto representa a base para o presente paradigma pois difere do paradigma estruturado onde o problema é convertido em sub programas com dados próprios. Em OOP o problema é transformado em classes e objectos onde deve-se identificar os dados (características) e os métodos (comportamento). Outros princípios são aplicados em objectos como é o caso do encapsulamento de membros em classes, herança que permite a reusabilidade.

Objectivos da Unidade

Após a conclusão desta unidade, deverá ser capaz de:

- Identificar a diferença entre uma linguagem OOP e uma estruturada
- Descrever os princípios básicos de uma linguagem OOP
- Desenvolver programas em OOP usando tipos de dados e estruturas de controle

Termos-chave

Classe: Conjunto de objectos com características comuns

Objecto: Uma unidade em execução que possui valor na memória

Abstracção: O uso de um recurso (método, classe, estrutura) sem se preocupar com os detalhes de implementação

Encapsulamento: O conceito segundo o qual tudo o que existe no programa deve estar dentro duma classe.

Herança: É a funcionalidade que permite uma classe herdar as características de outra.

Polimorfismo: O conceito segundo o qual algo pode assumir várias formas

Actividades de Aprendizagem

Actividade 1 - OOP versus programação estruturada

Introdução

A programação orientada a objectos veio trazer uma nova abordagem na forma como os programas eram desenhados e organizados onde ao invés de escrever especificar as acções em funções e procedimentos espalhados pelo programa todo passou-se a incorporar todos os dados (variáveis) e métodos (funções) em uma única unidade chamada classe. Esta actividade permite discutir as várias diferenças entre as duas abordagens tendo como ênfase: a complexidade, performance, estrutura e tamanho de softwares (pequenos ou grandes).

Detalhes da actividade

A programação orientada a objectos trouxe uma nova dinâmica no processo de desenho de softwares em detrimento da estruturada. As linguagens estruturadas centram-se na divisão do problema em partes com funcionalidades específicas, usam a abordagem top-down, focam-se em produzir instruções para resolver o problema e não permitem esconder dados. Por outro lado nas linguagens orientadas a objectos o problema é modelado em objectos que possuem dados e métodos que são protegidos. O mais importante é identificar os objectos e as mensagens entre eles. Os programas são mais simples e de fácil perceber pois qualquer coisa que existe no mundo é uma representação de objectos. É desta forma que a presente actividade visa-lhe ensinar as várias diferenças que existem entre estas duas linguagens.

Objectivos específicos da actividade

No final da aprendizagem, você deve ser capaz de:

- Identificar as potencialidades das linguagens OOP
- Diferenciar as linguagens OOP das estruturadas olhando para a complexidade, tratamento de erros, abordagem e

Conceitos chave

Tabela comparativa das linguagens OOP vs estruturadas

Paradigma Orientado a Objectos	Paradigma Estruturado
Usa as seções de um programa para executar determinadas tarefas. Ele divide o programa em objectos que podem ser reutilizados em outros programas Um programa orientado a objecto é decomposto em uma rede de objetos de colaboração. Para qualquer actividade do programa, um objecto responsável por essa actividade pode interagir com outros objectos, invocando os seus comportamentos, ou "métodos", até que a actividade seja concluída.	Assume a abordagem top-bottom. É baseada em torno de estruturas de dados e rotinas. Ele divide as tarefas em formas modulares. Realiza determinadas tarefas para que de uma razão específica. Um programa estruturado é decomposto em uma hierarquia de processos.

Lista de fontes relevantes

Lista de leituras

Leitura nr 1: Programação Estruturada vs Programação Orientada a Objectos (Dr K R Bond 2000)

Resumo: Este texto fala sobre o surgimento da programação orientada a objectos e faz um estudo comparativo das duas abordagens olhando para os seguintes aspectos: estrutura top - down, programação modular, resolução de problemas, complexidade das instruções e compilação.

Justificação: Este pequeno texto permitirá perceber as principais diferenças entre as duas abordagens.

Leitura nr 2: Referência completa: The Computer Revolution/Programming/Object Oriented vs. Structured programming

http://en.wikibooks.org/wiki/The_Computer_Revolution/ProgrammingObject_Oriented_vs._Structured_programming Site visitado aos 25 de Outubro de 2014 pelas 22:16

Resumo: Este website oferece um resumo sobre as principais diferenças entre as linguagens estruturadas e as linguagens orientadas a objectos.

Descrição detalhada da actividade

Esta actividade articula-se em volta das principais diferenças entre a programação orientada a objectos e programação estruturada. Tendo em conta os conhecimentos prévios sobre programação estruturada você deve primeiro caracterizar as linguagens estruturadas e depois realizar um estudo comparativo com a nova abordagem, a orientada a objectos.

Para tal são fornecidas as leituras.

Actividade

Criar uma tabela comparativa entre as duas linguagens. Os pontos de partida são:

	Abordagem	Resolução de problemas	complexidade das intruções
Estruturada			
Orientada a objectos			

Você e é livre de acrescentar mais pontos comparativos à tabela.

O estudo deve ser realizado em grupos de 2 (no máximo)

Conclusão

Nesta actividade foi possível perceber a origem

Avaliação

1. Elabore um quadro comparativo que ilustra as grandes diferenças entre as linguagens estruturadas e orientadas a objectos.

Actividade 2 - Definição e importância das linguagens OOP

Introdução

As linguagens orientadas a objectos permitemm reduzir a complexidade de programas robustos para grandes empresas permitindo que este esteja organizado de acordo com a estrutura orgânica destas empresas. A presente actividade vai-lhe permitir desenvolver capacidades para definir e perceber os benefícios duma linguagem OOP em especial a linguagem java.

Detalhes da actividade

Objectivos específicos da actividade

No final da aprendizagem, você deve ser capaz de:

- Identificar as potencialidades das linguagens oop
- Definir o conceito oop e os seus princípios básicos

Conceitos chave

OOP - Programação Orientada a objectos

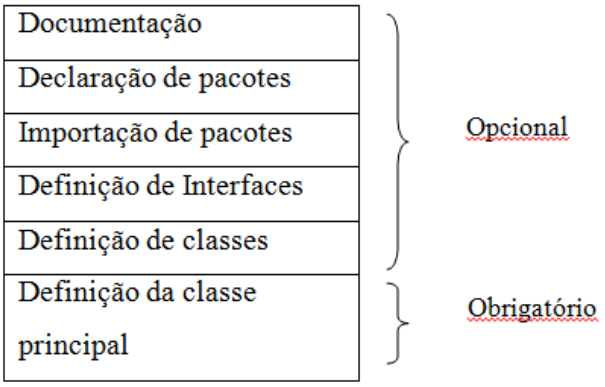
Para que uma linguagem seja considerada OOP ela deve ser capaz de implementar os seguintes conceitos básicos:

- Classes
- Objectos
- Abstracção e Encapsulamento
- Herança
- Polimorfismo

Java

Linguagem de programação orientada a objectos criada pela Sun Microsystems.

A programação básica em java consiste na técnica de fazer com que o computador esteja pensando através de estruturas de controle que operam sobre variáveis. A seguir iremos discutir a estrutura do programa em java:



Sendo assim, um programa básico em java deve possuir o formato que se segue:

```
class MinhaClasse {
        public static void main(String args[])
        {
                ....
        }
}
```

Onde main é onde serão invocados todos os métodos.

Lista de fontes relevantes

Lista de leituras

Leitura 1: Programação Orientada a Objectos: Uma abordagem com o Java

Referência Completa: Extraido do livro de Ivan Luiz Marques Ricarte da Universidade Estadual de Campinas

http://www.dca.fee.unicamp.br/cursos/PooJava/Aulas/poojava.pdf Site visitado aos 25 de Outubro de 2014 pelas 22:20

Resumo: Este manual infere sobre os princípios básicos duma linguagem orientada a objectos: Classe, Objectos, herança e polimorfismo, onde são apresentadas definições e exemplos concretos sobre estes conceitos.

Justificativa: Este manual irá lhe ajudar a a perceber o conceito de programação orientada a objectos (OOP) e os seus princípios básicos.

Leitura 2: Introdução a Programação Orientada a Objectos

Referência Completa: **Video aula sobre a programação orientada a objectos do Grupo SAE** - UCB

https://www.youtube.com/watch?v=gKp0pBkxASE Site visitado aos 25 de Outubro de 2014 pelas 23:16

Resumo: Este vídeo apresenta o conceito OOP de forma explícita e motivadora sob a perspectiva da linguagem java.

Justificativa: Este vídeo mostra a combinação entre os princípios OOP previamente discutidos e a linguagem java. Este demostra também o porquê da linguagem java ser independente de plataforma através da máquina virtual java (JVM).

Descrição detalhada da actividade

Esta actividade sobre a definição e importância das linguagens OOP insere-se em leitura individual sobre os princípios mencionados nas referências previamente mencionadas. Você deverá ler e realizar as seguintes actividades:

Tarefa 1: Leitura sobre as linguagens OOP e o java

Tarefa 2: Resumo de no máximo 5 páginas sobre os conceitos OOP, benefícios das linguagens OOP e a linguagem java como OOP.

Actividades de aprendizagem

Tarefa 1:Leituras sobre linguagens OOP e o java

Leia as referência à sua disposição sobre as linguagens OOP

Tarefa 2:Resumo

Prepare um resumo de no máximo 5 páginas onde deve inferir sobre os aspectos relevantes sobre o tópico em causa. Está livre de consultar outras referências.

Conclusão

Nesta actividade foi possível perceber os conceitos básicos de uma linguagem OOP, seus benefícios e exemplos.

Avaliação

1. A avaliação desta actividade consiste na produção de um resumo de no máximo 5 páginas onde você deverá falar das linguagens OOP, suas vantagens e a linguagem java (historial, vantagens e estrutura)

Actividade 3 - Variáveis, tipos de dados e estruturas de controle

Introdução

As variáveis permitem reservar espaço na memória do computador de modo que possamos usa-las e estão ligadas aos tipos de dados que definem o tipo de símbolos que a varável suporta bem como o intervalo. Desta forma a presente actividade visa a discutir como é que são declaradas as variáveis em java e quais os seus tipos de dados. Iremos também discutir as estruturas de controle (sequência, condição, repetição e salto) que permitem desviar o fluxo ou a sequência de um programa, dando a percepção de que o computador é inteligente (enquanto não)

Detalhes da atividade

Objectivos específicos da actividade

No final da actividade , você deve ser capaz de:

- Declarar variáveis e escolher os respectivos tipos de dados
- Perceber as estruturas de controle em oop

Conceitos chave

variáveis: Um nome que indica um endereço de memória que permite armazenar dados tipos de dados: Dados que indicam as características do endereço de memória, isto é, o conjunto de valores possíveis e o intervalo suportado. Variável consiste em qualquer coisa capaz de armazenar um valor ou uma expressão e que expressa um endereço (posição) na memória.

Por exemplo, se quisermos armazenar a idade de alguém podemos criar uma variável idade. Contudo, não importa dar o nome como também importa definir o intervalo de valores possíveis que a variável pode armazenar, bem como o seu tamanho na memória. É a isto que nós chamamos de tipo de dado.

Principais tipos de dados

Tipo	Descrição	Tamanho na memória	Intervalo de valores
byte	Números inteiros	1 byte	-128 a 127
short	Números inteiros	2 bytes	-32768 a 32767
int	Números inteiros	4 bytes	-2147483648 a 2147483648
long	Números inteiros	8 bytes	-263 a 262
float	ponto flutuante	4 bytes	3.4e-038 a 1.7e038
double	ponto flutuante	8 bytes	3.4e-308 a 1.7e+308
string	sequência de caracteres	2 bytes (por caracter)	
classe	objectos	depende	
array	variável capaz de armazenar uma sequência de valores distintos (números, caracteres, objectos, etc)	depende do tipo	

Estruturas de controle

As estruturas de controle são usadas para desviar o fluxo tradicional de um programa (top-down), fazendo com que o computador pareça estar tomando decisão. As estruturas de controle ajudam-nos também a processar de forma repetitiva uma série de valores.

A seguir vamos discutir as categorias existentes no Java:

Condicionais	Repetição	Salto
IF	DO	BREAK
SWITCH	WHILE	CONTINUE
	FOR e FOREACH	

Lista de fontes relevantes

Lista de leituras

Leitura 1: Extracto do livro de Deitel 2010. Java Como programar. 81p

Capítulo relativo as estruturas de controle em java com ênfase nas condicionais, repetição e salto.

Resumo: Este extracto aborda aspectos práticos sobre como programar em java usando operadores das estruturas de controle. Apresenta sintaxes e exemplos claros para cada controlador (if, switch, do, while, for e foreach, break e continue).

Justificativa: Este livro ajudará ao estudante a perceber melhor a programação em java pois apresenta ilustrações e programas simulados.

Leitura 2: Oracla Java Documentation

The Java ™ Tutorials http://docs.oracle.com/javase/tutorial/java/nutsandbolts/flow.html Site visitado aos 25 de Outubro de 2014 pelas 23:40

Resumo: Este documento apresenta todas as estruturas de controle, variáveis e tipos de dados em Java.

Justificativa: Este texto é de grande valia pois provêm da própria Sun Microsystems (Oracle) a empresa fundadora da linguagem java. Por isso este constitui referência para qualquer programador.

Descrição detalhada da actividade

Esta actividade visa a proporcionar-lhe habilidades de programar os tipos de dados e estruturas de controle onde terá que realizar as seguintes tarefas:

tarefa 1: Leitura

tarefa 2: Experiência prática no computador

Actividades de aprendizagem

A presente actividade de aprendizagem está inserida no contexto das variáveis, tipos de dados e estruturas de controle em java, pelo que consiste nas seguintes actividades:

Tarefa 1 Leitura

Realize uma leitura sobre as diferentes formas de desviar o fluxo de um programa usando as estruturas de controle disponíveis em java. Use as referências da disciplinas para tal.

Tarefa 2 Experiências práticas no computador

Use um computador para compilar e executar programas sobre estruturas de controle onde deve correr os exemplos mencionados nas leituras da actividade.

Conclusão

Nesta unidade foi possível perceber os principais princípios que norteiam uma linguagem OOP, sua importância na programação e os conceitos básicos de programação OOP.

Avaliação

1. Quais os principais princípios de uma linguagem orientada a objecto?
2. O que é uma variável?
3. Quais as regras para dar nome a uma variável?
4. Desenvolva um programa que permite introduzir dois números e imprimir a média dos dois. Não são permitidos números negativos.

Actividade 4 - Laboratório da Unidade 1

Aulas laboratoriais da unidade 2

Laboratório 1

Objectivo do exercício

Conceber um programa numa linguagem estruturada e converte-lo para uma linguagem orientada a objecto

Recursos

JKD 1.8 e Note pad

Tempo

5 horas

Descrição do exercício

1.1 Considere uma aplicação que possui uma função para calcular a média de dois números. Desenvolva a aplicação usando uma linguagem estruturada e depois converta a mesma aplicação para OOP.

a. Usando Java

b. Usando o C

c. Enumere as diferenças existentes nos dois programas sob o ponto de vista de organização

Resultados e Submissão

Proposta de solução

a) Usando o C

```
#include<stdio.h>

float media(float n1, float n2){

      return (n1+n2)/2;

}

int main(){

      float n1=11, n2=7;

      printf("Hellooo
\n A media de %f e %f =
%f",n1,n2,media(n1,n2));

      return 0;

}
```

b)Usando o Java

```
class Prog1{

      public static float
media(float n1,float n2){

            return (n1+n2)/2;

      }

      public static void main
(String args[])

      {

            float n1=11, n2=7;

            System.out.
print("Hello \n a media de"+n1+"
e"+n2+"="+media(n1,n2));

      }

}
```

O output:

Hello

a media de11.0 e7.0=9.0

c) Diferenças existentes nos dois programas sob o ponto de vista de organização

i) Usando o C, o nome do ficheiro não tem nada a ver com o conteúdo do programa, contudo, usando o Java, o nome do ficheiro deve ser igual ao nome da classe

ii) O método média: no C está dentro do programa enquanto no Java este está dentro do programa e dentro da classe.

Submissão: A solução deve ser submetida no final da aula laboratorial. Você deve executar a classe e demonstrar a solução. Este pode ser submetido via email caso não termine a actividade no laboratório.

Critérios de avaliação

O presente exercício consiste na execução com sucesso do programa no laboratório:

Peso da avaliação 5%

Referências ou Links chaves

The Computer Revolution/Programming/Object Oriented vs. Structured programming.

http://en.wikibooks.org/wiki/The_Computer_Revolution/Programming/Object_Oriented_vs._Structured_programming

Laboratório 2

Objectivo do exercício

Desenvolver programas em OOP usando as variáveis, tipos de dados e estruturas de controle de Java

Recursos

JKD 1.8 e Note pad

Tempo

5 horas

Descrição do exercício

1. Qual será o output do seguinte bloco de instruções em java?

```
int a=5,b=2;
float R=a/b;
System.out.print(R);
```

Solução:

O Resultado será 2.0 pois a operação entre inteiros o resultado será inteiro.

2. Aplique a conversão manual no exercício anterior (casting) de modo que o resultado seja o desejável.

Solução:

int a=5,b=2;

float R=(float)a/b;

System.out.print(R);

Agora sim o resultado será 2.5

3. Escreva um programa que imprime todos os números ímpares no intervalo de 1 a 100. Podemos recorrer ao continue para ignorar a impressão dos números pares. Recorde que números pares são aqueles cuja divisão modular por 2 é sempre zero.

Solucão

```
class Prog3 { public static void main(String args[]){
     for(int i=1; i<=100; i++) {
            if(i%2==0) {
                  continue;
            } System.out.print(i);
      }
 }
}
```

O output será:

135791113151719212325272931333537394143454749515355575961636567697
173757779818385878991939597 99

4. Escreva um programa que imprime a seguinte série:

1

12

123

1234

12345

Solução:

Para tal é necessário criar dois cilcos, um para as linhas (L) e o outro para as colunas (C).

```
class Prog5 { public static void main(String args[]){
    for(int L=1;L<=5;L++)
      {          for(int C=1; C<=L; C++)
            { System.out.print(C); }System.out.
println();
      }
    }
}
```

Submissão

A solução deve ser submetida no final da aula laboratorial. O estudante deve executar a classe e demonstrar a solução. Este pode ser submetido via email caso não termine a actividade no laboratório.

Critérios de avaliação

O presente exercício consiste na execução com sucesso do programa no laboratório:

Peso da avaliação 4%

Referências ou Links chaves

Oracla Java Documentation The Java ™ Tutorials

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/flow.html

Site visitado aos 25 de Outubro de 2014 pelas 23:40

Resumo da Unidade

As linguagens OOP permitem desenvolver softwares de forma mais fácil e eficaz, pois todo o problema é transformado para classes e objectos que se assemelham ao mundo real. Na presente unidade foram discutidos as principais diferenças entre as linguagens estruturadas e as orientadas objectos onde foi realizado um estudo comparativo com recurso ao acervo bibliográfico disponível na web. Logo depois foram discutidos os princípios fundamentais de uma linguagem orientada a objecto onde foi possível constatar que todas as linguagens OOP suportam os princípios: classes, objectos, herança, abstracção, encapsulamento e polimorfismo. Na última actividade foram discutidos em forma de revisão as principais estruturas de controle (condição, repetição e salto) da linguagem Java.

Avaliação da Unidade

Verifique a sua compreensão!

Exercícios Práticos

Instruções

Responda com atenção as questões da avaliação. Pode usar o computador para executar os programas.

Critérios de Avaliação

A cotação geral será dividida por cada uma das 3 questões, tendo a questão 1 7 pontos, a questão 2 7 pontos e a questão 3 6 pontos.

1. Escreva um programa onde o utilizador digita três notas dos seus testes e o sistema deve imprimir a média e dizer se está aprovado, reprovado ou dispensado, dependendo da média.
2. Escreva um programa onde o utilizador digita o valor em USD e o sistema imprime a sua conversão para Meticais.
3. Qual será o resultado (output) do programa que se segue?

```
            class ABC {
               public static void main(String args[])
{   int x=0, y=1;
     y++;
     int R=x++  +  --y  +   ++x;
     System.out.println("R="+R+"x="+x+"y="+y);
}
            }
```

Responda

1. Média dos testes:

```
import java.util.Scanner;

public class Media {

    public static void main(String[] args) {

        Scanner ent = new Scanner(System.in);

        int nota1, nota2, nota3;
```

```
        int media, i, contAluno = 0;
        for(i = 0; i < 3; i++){
            contAluno++;
            // recebe a 1° nota
            System.out.println("Aluno " + contAluno + ",
digite sua 1ª nota");
            nota1 = ent.nextInt();
            // recebe a 2° nota
            System.out.println("Aluno " + contAluno + ",
digite sua 2ª nota");
            nota2 = ent.nextInt();
            // recebe a 3° nota
            System.out.println("Aluno " + contAluno + ",
digite sua 3ª nota");
            nota3 = ent.nextInt();
            // calcula a média
            media = (nota1 + nota2 + nota3) / 3;
            System.out.println("A média do aluno " +
contAluno + « é « + media);
            // mostra a nota do aluno
            if( (media >= 0) && (media <4) ){
                System.out.println("Nota E");
            } else if(media < 5){
                System.out.println("Nota D");
            } else if(media < 7){
                System.out.println("Nota C");
            } else if(media < 8){
                System.out.println("Nota B");
            } else if(media <= 10){
                System.out.println("Nota A");
            }
        }
```

```
        }
}
```

2. Convertendo de Meticais para Dollar

```
import java.util.Scanner;

public class Media {

  Scanner scan = new Scanner(System.in);

    float MT;

    System.out.println("Convertendo Dolar para MT");

    System.out.println("Cotação do dolar: 1USD = 50MT");

    System.out.println("Digite a quantia em MT para ser
convertida: ");

    MT = scan.nextFloat();

    //pegamos o valor em dolares e multiplicamos pelo valor
do cambio, em MT

    System.out.println("O valor em MT é"+ MT*50);

}
```

3. R=3 x=2 y=1

Leituras e outros Recursos

As leituras e outros recursos desta unidade encontram-se na lista de Leituras e Outros Recursos do curso.

Beginner Java - Intro to Swing (GUI) - Lesson 28 https://www.youtube.com/watch?v=uXitpup0l-s Visitado aos 13 de Maio de 2016 Neste vídeo o autor demostra como criar formulários usando o pacote swing. Este permite perceber o quanto é fácil desenhar formulários do pacote swing.

Beginner Java - Intro to Swing (GUI) - Lesson 28 https://www.youtube.com/watch?v=2ly7d_uMlrI Visitado aos 13 de Maio de 2016 Este vídeo é a continuação do anterior, onde o autor apresenta de forma muito simples o método de adição de eventos aos objectos, com destaque à interface ActionListener.

Unidade 2. Classes, Objectos e Membros.

Introdução à Unidade

A linguagem java é puramente orientada a objectos, pelo que suporta todas as regras do paradigma OOP. O conceito de classes e objectos é fundamental para se programar neste paradigma. Deste modo, todas as classes possuem características e comportamentos. As funções, que agora são designadas de métodos definem o comportamento e as variáveis, que agora serão designadas de dados dizem respeito as características da classe. Ao conjunto de métodos e dados duma classe chamamos de membros.

Assim, a presente unidade visa discutir os conceitos básicos relacionados as classes e objectos: Construtores, métodos e dados estáticos, visibilidade e herança e seus tipos. Estes tópicos serão divididos em três actividades de modo a facilitar a compreensão.

Objectivos da Unidade

Após a conclusão desta unidade, deverá ser capaz de:

- Criar objectos e inicializar os seus dados
- Implementar as técnicas de partilha de membros entre objectos através da palavra reservada "static"
- Dominar as diferentes visibilidades e heranças em OOP

Termos-chave

Membros da classe: Conjunto de métodos e dados

Construtores: Método que serve para inicializar os dados do objecto

Static: Pertence à classe e não ao objecto

Herança: Propriedade que permite que uma classe herde os membros da outra.

Actividades de Aprendizagem

Actividade 11 - Membros da classe

Introdução

Todos os dados e métodos duma classe são designados de membros. Um método que pertence a uma classe pode aceder a todos os membros da mesma, contudo, se estivermos no main será necessário criar um objecto para aceder aos membros da classe. Por isso, nesta actividade iremos discutir os aspectos relevantes na criação e manipulação destes.

Detalhes da actividade

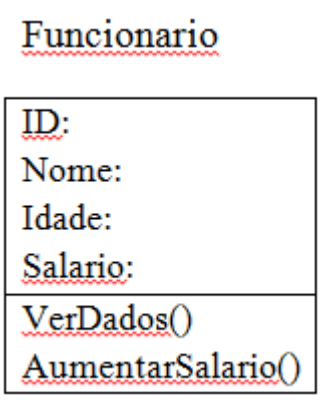
Como foi discutido no capítulo anterior, as linguagens OOP consistem na modelação da realidade em classes e objectos. Nesta unidade , iremos discutir as técnicas e conceitos práticos sobre classes e objectos bem como as regras da sua manipulação em Java.

Dados

Uma classe criada em java possui a seguinte estrutura:

```
class nome da classe [extends]
{
	[definição de dados]
	[definição de métodos]
}
```

Por exemplo: pretende-se criar a seguinte classe:



Class Funcionario

```
{
        public int ID;
        public String Nome;
        public int Idade;
        public int Salario;
        public static void main(String args[])
        {
                Funcionario F1=new Funcionario();
                Fncionario F2=new Funcionario();
                F1.ID = 100;
                F1.Nome=”Joao”;
                F1.Salario=1200;
                F2.ID=101;
                F2.Nome=”Joana”;
                F2.Idade=25;
        }
}
```

Neste caso foram criados dois funcionários F1 e F2, e usou-se o operador “.” para aceder a cada membro de cada objecto.

Métodos

Métodos são um conjunto de instruções com a finalidade de resolver uma certa tarefa. Por exemplo, se temos instruções que simulam uma calculadora, podemos criar um método e chama-lo: calculadora(). Assim, sempre que pretendemos calcular a soma de dois números não precisamos repetir o código, mas sim, chamar o método. Os métodos podem ser invocados diversas vezes no programa.

Exemplo: Pretendemos criar um método que imprime “Hello World!!!”

class ABC

```
{
        public void ola()
        {
                System.out.print("Hello World!!");
        }
}
```

Como chamar o método no main: Usamos a mesma notação que foi usada para os dados, através do operador ".".

class ABC

```
{
        public void ola()
        {
                System.out.print("Hello World!!");
        }
        public static void main(String args[])
        {
                ABC obj  = new ABC();
                    obj.ola();
}
}
```

Assim, o output será de facto: "Hello World"

Lista de fontes relevantes

Lista de leituras

Leitura 1: Extrato do livro de Deitel 2010. Java Como programar. 58p

Capítulo relativo ao processo de criação de uso de objectos, onde também se discutem as formas de aceder aos dados e métodos do objecto (membros).

Leitura 2: Extrato do livro de Balagurusamy 2010. Programming with Java a primer. 123-144p

Neste capítulo poderá aprender o processo de criação de classes e objectos, bem como a programação de métodos e dados. Apesar de estar em inglês, o autor usa uma linguagem simples e com exemplos que permitem perceber os conceitos.

Conclusão

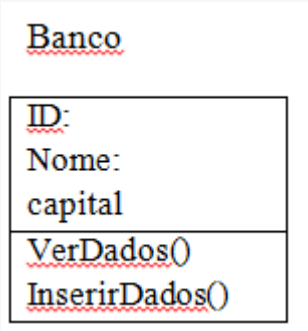
Nesta actividade foi possível discutir a técnica da criação de objectos através de classes previamente criadas e as formas de inicializar ou visualizar os dados ou métodos dos objectos criados.

Este é o princípio fundamental para se poder programar usando linguagens orientadas a

Avaliação

1. Uma clínica pretende gerir os dados dos médicos que atendem pacientes. Todos os médicos possuem ID, Nome, Especialidade e Nível Académico. Crie uma classe que permite armazenar e visualizar dados de pelo menos 5 médicos.

2. Considere a seguinte classe:



a) Complete os métodos da classe e crie três Bancos(objectos) dela

b) Imprima a soma dos capitais dos três Bancos.

Actividade 2 - Construtores e Métodos/Dados estáticos

Introdução

Por vezes pretendemos definir dados padronizados para todos objectos logo que forem criados, isto é, definir um valor inicial aos dados do objecto logo após a sua criação. Isto é possível através do uso de construtores. Por outro lado há situações nas quais pretendemos ter membros partilhados entre objectos. Para tal declaramos o tal método ou dado como estático.

A presente actividade visa desenvolver programas em OOP usando estes dois conceitos.

Detalhes da atividade

Construtores

Construtor é um método que tem o mesmo nome com o da classe e será implicitamente executado imediatamente após a criação do objecto

REGRAS BÁSICAS

- Não deve ser void e nem não void (não tem tipo de retorno não)
- Tem o mesmo nome com o da classe
- É um método que pode ter argumentos como não

Se por exemplo temos a classe Funcionário podemos criar um construtor: public Funcionário que inicializa os dados funcionários, isto é, todo o funcionário que for criado terá por defeito os dados definidos no construtor.

class Funcionario

```
{
        public int ID;
        public String Nome;
        public int Idade;
        public double Salario;
        public void VerDados()
        {
                System.out.print("Nome =  "+Nome);
                System.out.print("Idade =  "+Idade);
                System.out.print("Salário = "+Salario);
        }
```

```
            public Funcionario( )
    {
            Nome = "xxx";
            Idade = 0;
            Salario = 1000;
    }
    public static void main(String args[])
            {           Funcionario F1=new Funcionario();
                        Funcionario F2=new Funcionario();
                        F1.VerDados();  F2.VerDados();
            }
    }
```

No exemplo acima podemos constatar que tanto o F1 como F2 por defeito terão Nome=xxx, idade= 0 e Salario=1000. Ai podemos alterar os dados individuais de cada um.

Métodos e dados estáticos

Nos capítulos anteriores vimos que ao criar um objecto, este passa a conter todos os dados e métodos da classe. Contudo existem situações em que pretendemos criar um método ou dado que seja comum a todos objectos da classe de modo que estes possam partilha- lo. Isto só é possível se declararmos um método ou dado como static.

REGRAS BÁSICAS

- Um membro estático pertence à classe e não ao objecto
- Um dado/método estático pode ser executado mesmo que não exista nenhum objecto criado, basta usar o nome da classe
- Um dado estático tem o valor zero por defeito
- Um método estático só pode aceder a dados/métodos estáticos

O exemplo a seguir pretende criar um programa onde temos vários objectos a efectuarem uma certa contribuição, depositando um valor numa variável chamada doação que por defeito inicia com o valor zero.

Cada objecto deverá efectuar o incremento por 100 da variável doação e no fim, usaremos o nome da classe para imprimir o valor existente na doação.

class Morador

```
{
        public static int doacao;
        public String Nome;
        public Morador( String  n)
{   Nome = n;
}
public static void main(String args[])
        {           Morador M1=new Morador("Carlos");
                    Morador M2=new Morador("Joao");
                    M1.doacao=M1.doacao+100;
                    M2.doacao=M2.doacao+100;
                    System.out.println("Valor total"+Morador.
doacao);
        }
}
```

Teremos o seguinte output: Valor total 200, pois M1 e o M2 incrementam a mesma variável por ela ser estática.

Lista de fontes relevantes

Lista de leituras

Leitura 1: Extrato do livro de Deitel 2010. Java Como programar. 58p

Capítulo relativo ao processo de criação de uso de objectos, onde também se discutem as formas de aceder aos dados e métodos do objecto (membros).

Actividade 3 - Herança e seus tipos

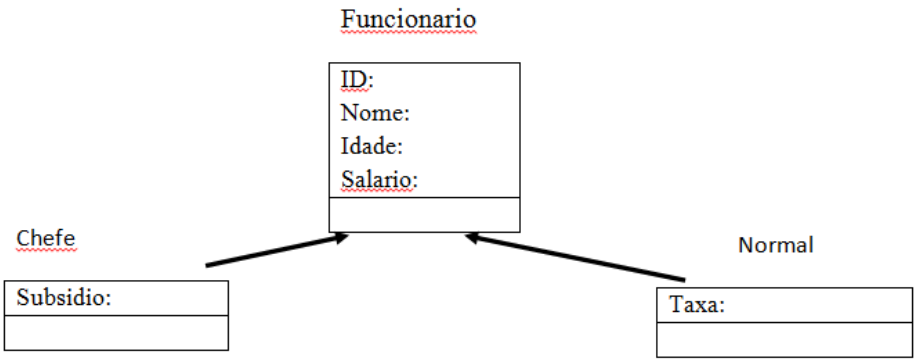
Introdução

A herança consiste na habilidade de uma classe herdar as características de uma outra. Este conceito é muito importante, pois permite-nos fazer o uso dos métodos/dados de uma classe, sem necessariamente termos que reprogramar, mas sim, reutilizar o que já existe.

Na herança, uma classe herda a outra sem modificar o seu conteúdo, onde a classe que herda é chamada de sub classe ou classe derivada, e a classe que é herdada é chamada de super classe ou classe derivada.

Detalhes da actividade

A herança é importante pois permite reutilizar o que já existe numa outra classe. Considere uma empresa onde todos os funcionários possuem ID, nome, idade e salário. Contudo o funcionário chefe possui ainda o subsídio e o funcionário normal possui taxa. O modelo desta classe ficaria:



Deste modo, para efectuar a herança em java usamos a palavra reservada "extends"

class Funcionario

```
{
        public int ID;
        public String Nome;
        public int Idade;
        public double Salario;
}
class FChefe extends Funcionario
{
        public int subsidio;
}
```

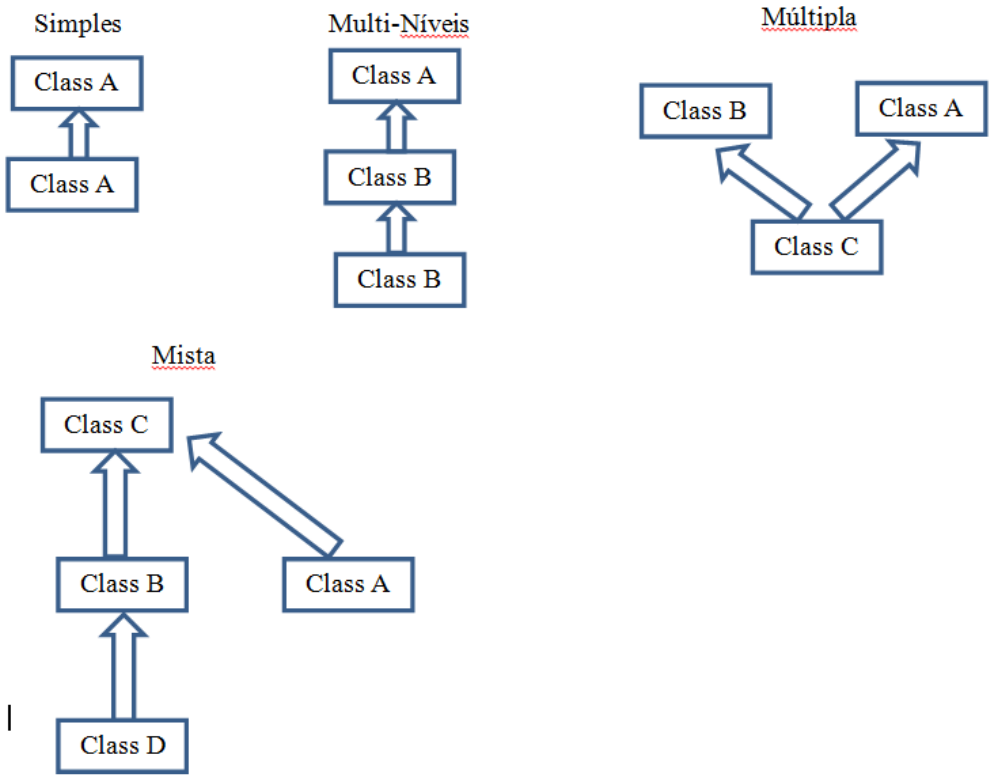
```
class FNormal extends Funcionario
{
        public int taxa;
}
class Programa{
        public static void main(String args[])
        {
                FChefe F1=new FChefe();
                F1.Nome=”Jafete”;
                F1.subsidio=1000;
                F1.Idade=35;
                FNormal F2=new FNormal();
                F2.Nome=”Julia”;
                F2.taxa=1250;
        }
}
```

O Objecto F1 do tipo FChefe só pode aceder aos dados/métodos de Funcionário e seus próprios dados que é o subsídio. O mesmo acontece com o F2 que só pode aceder aos dados do Funcionário mais a taxa.

Contudo, se criamos um objecto do tipo funcionário, este não pode aceder a nenhum dado da sua subcasse pelo a instrução:

Tipos de Herança

A combinação entre classes através da herança é importante pois permitem que os sistemas desenvolvidos espelhem a realidade das organizações que muitas das vezes estão organizadas em sectores e hierarquias. Deste modo, a linguagem Java permite os seguintes tipos:



Interfaces

Como foi dito, Java não suporta herança múltipla e para tal faz o uso do conceito de interfaces.

Interface é uma classe como qualquer outra, contudo os seus dados são constantes (não mudam o seu valor) e os seus métodos não tem corpo, sendo necessário fazer a herança e a sobreposição destes para poder usa-los.

REGRAS BÁSICAS

- Os dados são constantes
- Os métodos não tem corpo, apenas a definição.
- Podem fazer o extends de várias interfaces
- Não podem ser instanciadas, isto é, não podem ter objectos.

Exemplo:

interface Mae

```
{
        public int ID=100;
        public void show();
}
interface Pai{
        public int ID=111;
        public void imprime();
}
class Filho implements Mae,Pai {
public void show() {System.out.println("Sou classe Mae"); }

public void imprime(){ System.out.println("Sou classe Pai");
}

public static void main(String args[])
        {
                Filho F = new Filho();
                F.show();
                F.imprime();
        }
}
```

A classe filho faz a herança das classes Mãe e Pai e faz o override dos métodos show e imprime.

É da responsabilidade da classe derivada fazer a sobreposição de todos os métodos das interfaces nas quais ela fez a herança (implements).

Actividade 4 - Laboratório da Unidade 2

Aulas laboratoriais da unidade 2

Laboratório 1

Objectivo do exercício

Compreender e implementar a herança entre classes na resolução de um problema concreto

Recursos

JKD 1.8 e Note pad

Tempo

4 horas

Descrição do exercício

Considere a seguinte classe:

Devedor
String ID: String Nome: Float Dívida: Float Pagamento
void Ver Dados() void Inserir Dados(String, String, float, float) void Estado()

a) Complete os métodos da classe de modo que:

i) Ver Dados visualize todos os dados do Devedor

ii) Inserir Dados permite inicializar o ID, o Nome e a Dívida do devedor

iii) Estado verifica se o devedor está em dívida ou não.

b) Crie dois objectos (devedores) e imprima a soma dos pagamentos e das dívidas dos dois.

Resultados e Submissão

```
Proposta de solução

class Devedor {

   public String id;

   public String nome;

   public float divida;
```

```
   public float pagamento;

   //Inicialização dos dados

   public void verDados(){

         System.out.println("ID:"+this.id+"Nome:"+this.
nome+"\n");

         System.out.println("Pagamentos:"+this.
pagamento+"Divida:"+this.divida+"\n");

   }

   //Inserção de dados

   public void inserirDados(String ID,String NOME,float
DIVIDA,float PAGMENTO)

   { this.id=ID; this.nome=NOME;this.divida=DIVIDA; this.
pagamento=PAGMENTO;}

   //Verificação do estado.

   public void estado()

   { float saldo = pagamento-divida;

     if(saldo>=0)

         System.out.print(this.nome+":Sem dívida, com
saldo"+saldo);

         else

         System.out.print(this.nome+":Devedor, com
saldo"+saldo);

   }

   public static void main(String args[]){

         //criação dos objectos

         Devedor d1 = new Devedor();

         Devedor d2 = new Devedor();

         //insercão e visualizacao

         d1.inserirDados("D1","Abel",1200,200);

         d1.verDados();

         d2.inserirDados("D1","Joelma",2000,2400);

         d2.verDados();
```

```
                //Visualizacao do saldo
                d1.estado();
                d2.estado();
            }
        }
```

Output será:

ID:D1Nome:Abel

Pagamentos:200.0Divida:1200.0

ID:D1Nome:Abel

Pagamentos:2400.0Divida:2000.0

Abel:Devedor, com saldo-1000.0Joelma:Sem d??vida, com saldo400.0

Submissão: A solução deve ser submetida no final da aula laboratorial. O estudante deve executar a classe e demonstrar a solução. Este pode ser submetido via email caso não termine a actividade no laboratório.

Critérios de avaliação

O presente exercício consiste na execução com sucesso do programa no laboratório:

Peso da avaliação 4%

Referências ou Links chaves

Harvey M. Deitel. Java: Como Programar. 6 ed. São Paulo: Pearson education do Brasil, 2005

Naughton, Patrick, Dominando o Java, Guia Autorizado da Sun Microsystems, Editora Makron Books, 1997.

Laboratório 2

Objectivo do exercício

Implementar os construtores e métodos estáticos em classes

Recursos

JKD 1.8 e Note pad

Tempo

4 horas

Descrição do exercício

Considere o programa anterior e realize as seguintes tarefas:

1. Crie um construtor com parâmetros de modo que este inicialize os dados do devedor, tal como o método inserirDados() faz.
2. Crie um dado estático que permite que os devedores com saldo positivo possam doar os seus pagamentos neste dado. Após a doação, imprima o estágio de todos

Resultados e Submissão

Proposta de solução

class Devedor2

```
{
   public String id;
   public String nome;
   public float divida;
   public float pagamento;
   public static float doacao;
   //Inicialização dos dados
   public void verDados(){
         System.out.println("ID:"+this.id+"Nome:"+this.
nome+"\n");
         System.out.println("Pagamentos:"+this.
pagamento+"Divida:"+this.divida+"\n");
   }
   //Construtor
   public Devedor2(String ID,String NOME,float DIVIDA,float
PAGMENTO)
   { this.id=ID; this.nome=NOME;this.divida=DIVIDA; this.
pagamento=PAGMENTO;}

   //Verificação do estado.
   public void estado()
   { float saldo = pagamento-divida;
     if(saldo>=0)
         System.out.print(this.nome+":Sem divida, com
```

```
saldo”+saldo);
        else
        System.out.print(this.nome+”:Devedor, com
saldo”+saldo);
   }
   //método para doacao: verificar o saldo, se for positivo
transferir o valor a doar
   //do pagamento para a variável estática
   public void doar(float v){
        float saldo = pagamento-divida;
        if(saldo>=0)
        {     this.pagamento-=v;
              doacao+=v;
        }
        else
        System.out.println(this.nome+”Nao tem saldo
suficiente!!!”);
   }
   public static void main(String args[]){
        Devedor2 d1 = new Devedor2(“D1”,”Abel”,1200,200);
        Devedor2 d2 = new Devedor2(“D2”,”Joelma”,2000,2400);
        d1.verDados();
        d2.verDados();
        d1.estado();
        d2.estado();
        System.out.println(“-------------Doando 50-------”);
        d2.doar(50);
        d1.doar(50);
        System.out.println(“-------------Apos
doacao-------”);
        d1.verDados();
        d2.verDados();
```

```
            System.out.println("Valor existente na
doacao:"+Devedor2.doacao);

   }

}
```

Output:

E:\java Devedor2

ID:D1Nome:Abel

Pagamentos:200.0Divida:1200.0

ID:D2Nome:Joelma

Pagamentos:2400.0Divida:2000.0

Abel:Devedor, com saldo-1000.0Joelma:Sem divida, com saldo400.0----------Doan

do 50-------

AbelNao tem saldo suficiente!!!

----------Apos doacao-------

ID:D1Nome:Abel

Pagamentos:200.0Divida:1200.0

ID:D2Nome:Joelma

Pagamentos:2350.0Divida:2000.0

Valor existente na doacao:50.0

Submissão: A solução deve ser submetida no final da aula laboratorial. Você deve executar a classe e demonstrar a solução. Este pode ser submetido via email caso não termine a actividade no laboratório.

Critérios de avaliação

O presente exercício consiste na execução com sucesso do programa no laboratório:

Peso da avaliação 4%

Referências ou Links chaves

Balagurusamy, E. Programming with Java a primer. 4th Edition. Tata McGraw-Hill. 2010

Laboratório 3

Objectivo do exercício

Compreender e implementar a herança entre classes na resolução de um problema concreto

Recursos

JKD 1.8 e Note pad

Tempo

3 horas

Descrição do exercício

Considere a classe Devedor desenvolvida na actividade anterior. Cria uma outra classe Principal que faz a herança de Devedor e chama os seus métodos. Note que:

Principal deve inicializar os dados de Devedor.

A classe Devedor não deve ter o método main, mas sim a classe Principal

Cada uma das classes devem ser compiladas em separado e armazenadas no mesmo ficheiro

Resultados e Submissão

Proposta de solução

public class Principal extends Devedor

```
{
   public static void main(String x[]){
   Principal d1 = new Principal();
   d1.inserirDados("D1","Abel",1200,200);
   Principal d2 = new Principal();
   d2.inserirDados("D2","Joelma",2000,2400);
   d1.verDados(); d2.verDados();
   }
}
```

Output:

E:\java>java Principal

ID:D1Nome:Abel

Pagamentos:200.0Divida:1200.0

ID:D2Nome:Joelma

Pagamentos:2400.0Divida:2000.0

Submissão: A solução deve ser submetida no final da aula laboratorial. O estudante deve executar a classe e demonstrar a solução. Este pode ser submetido via email caso não termine a actividade no laboratório.

Critérios de avaliação

O presente exercício consiste na execução com sucesso do programa no laboratório:

Peso da avaliação 3%

Referências ou Links chaves

Harvey M. Deitel. Java: Como Programar. 6 ed. São Paulo: Pearson education do Brasil, 2005

Naughton, Patrick, Dominando o Java, Guia Autorizado da Sun Microsystems, Editora Makron Books, 1997

Conclusão

Nesta unidade foi possível aprender a criar objectos e adicionar dados e métodos a ele. A protecção de membros da classe é importante pois permite decidir quais os membros são acessível fora da classe ou não. Foi possível discutir o conceito de herança que permite reutilizar o objecto.

Avaliação

1. Indique se as afirmações que se seguem são verdadeiras ou falsas:

   a. É um erro criar uma classe sem construtor

   b. Em java é proibido declarar dois métodos com mesmo nome na mesma classe.

   c. Em java é proibido declarar duas variáveis com o mesmo nome dentro da mesma classe

   d. Se dois métodos que são totalmente iguais (mesmo nome e argumentos) tiverem diferença no tipo (void o não void) podemos afirmar que houve sobrecarga.

   e. Uma variável estática pode ser usada no main() sem necessariamente ter que se criar objecto da classe

   f. Um método estático pode aceder uma variável qualquer.

Resumo da Unidade

Na presente unidade foram discutidas as ideias iniciais sobre o como desenhar programas orientados a objectos. Na actividade 1 foi possível implementar de forma prática a criação de classes e seus objectos, onde foi possível também criar métodos e dados de tais classes. Na actividade 2 discutiu-se o uso de dados e métodos estáticos, que permitiu que vários objectos partilhassem os mesmos dados. Na actividade 3 discutiu-se a implementação prática do conceito de herança e seus tipos, tendo-se usado as interfaces para o caso da herança múltipla.

Avaliação da Unidade

Verifique a sua compreensão!

Instruções

Utilize o notepad do seu computador para resolver os programas que se seguem (pode compilar usando o JDK)

Critérios de Avaliação

A pontuação total será dividida por cada uma das questões onde a questão 1 vale 10 pontos e a questão 2 valor 10 pontos também.

1. Explique os seguintes conceitos:

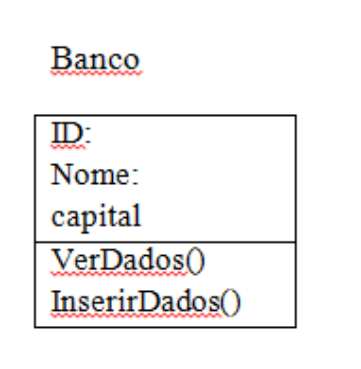
   Herança

   Static

   Instância

   main

2. Considere a seguinte classe:



   a) Complete os métodos da classe e crie três Bancos(objectos) dela

   b) Imprima a soma dos capitais dos três Bancos.

   c) Crie um construtor que inicializa os dados através de parâmetros.

3. Quais são as regras básicas para se criar um construtor?

4. Crie uma classe Principal e faça a herança da classe Banco, fazendo por seguinte o uso de todos os seus métodos no main.

Responda

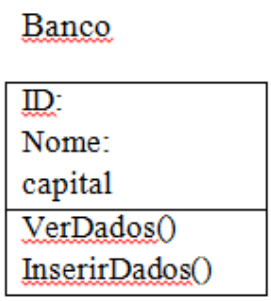
1. Herança: Capacidade de uma classe herdar métodos e dados de outra.

   Static: Pertence à classe e é partilhado pelos objectos

   Instância: Objecto

   main: Método principal, o primeiro a ser carregado na memória.

2.



import java.util.Scanner;

class Bancos

```
{
   public String ID;
   public String Banco;
   public float capital;
   public void VerDados(){
   System.out.print(ID+Banco+capital);
   }
   public Bancos(String id, String b, float c){
   ID = id; Banco = b; capital=c;
   }
   public void InserirDados(){
   Scanner T = new Scanner(System.in);
   ID = T.next(); Banco=T.next(); capital=T.nextFloat();
   }
   public static void main(String[] args) {
         Bancos b1 = new Bancos(); b1.VerDados();
```

```
        Bancos b2 = new Bancos(); b2.VerDados();
        Bancos b3 = new Bancos(); b3.VerDados();
        float R = b1.capital+b2.capital+b2.capital;
        System.out.println("Soma dos capitais:"+R);
    }
}
```

3. Não deve ser void e nem não void (não tem tipo de retorno não)

   Tem o mesmo nome com o da classe

   É um método que pode ter argumentos como não

4. class Bancos

```
{
    public String ID;
    public String Banco;
    public float capita;

    public void VerDados(){
    System.out.print(ID+Banco+capital);
    }
    public void InserirDados(){
    Scanner T = new Scanner(System.in);
    ID = T.next(); Banco=T.next(); capital=T.nextFloat();
    }
}
class Principal extends Bancos{
    public static void main(String[] args) {
        Principal b1 = new Principal(); b1.VerDados();
        Principal b2 = new Principal(); b2.VerDados();
        Principal b3 = new Principal(); b3.VerDados();
    }
}
```

Unidade 3. Pacotes e Tratamento de erros

Introdução à Unidade

Toda a linguagem de programação possui suas próprias bibliotecas onde cada uma é adicionada ao programa de acordo com as necessidades. A linguagem java organiza suas bibliotecas em pacotes, onde cada pacote possui uma série de classes.

Por outro lado a linguagem java permite gerir os erros que possam vir a surgir durante a execução de um programa, sendo assim possível personalizar as mensagens de erros, isto é, fazer com que o sistema retorne mensagens de erro definidas pelo programador.

Desta forma, a presente unidade visa dotar aos estudantes conhecimentos básicos na gestão de pacotes e tratamento de erros.

Objectivos da Unidade

Após a conclusão desta unidade, deverá ser capaz de:

- Identificar os pacotes básicos da linguagem java
- Perceber as principais classes de erros em java
- Gerir os pacotes, erros e excepções

Termos-chave

Pacote: Um conjunto de classes que pertencem à mesma pasta.

Erro de execução: Os erros de execução constituem uma ocorrência anormal durante a execução de um programa compilado com sucesso.

Erro de compilação: Os erros de compilação são erros detectados durante a escrita do programa, erros que derivem de violação da sintaxe ou gramática da linguagem.

Excepção: Erro que ocorre durante a execução de um programa

Actividades de Aprendizagem

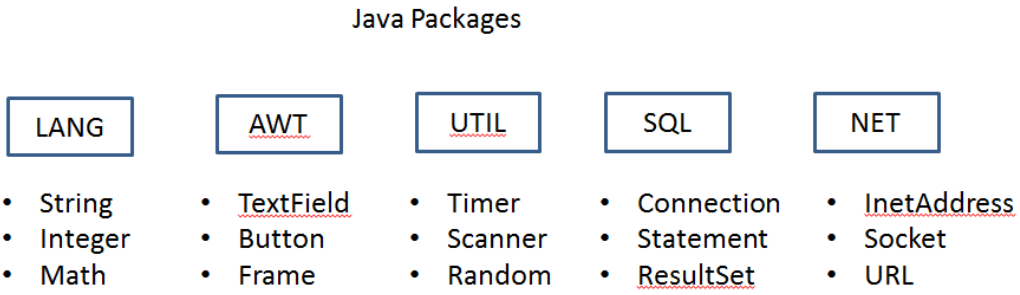
Actividade 1 - Pacotes em java

Introdução

Pacotes permitem agregar classes com funcionalidades comuns numa unidade só. A linguagem java possui pacotes que podem ser usados em qualquer programa.

Para importar o pacote usamos a palavra reservada "import" seguido do nome do pacote.

A estrutura básica dos pacotes em java:



Detalhes da actividade

A presente actividade consiste na importação de pacotes em java, bem como na criação de pacotes definidos pelo programador.

Para visualizar as classes de um pacote usamos o programa javap na linha de comando, por exemplo, para o caso da classe String do pacote lang:

javap java.lang.String

```
Importando a classe Scanner do pacote util

import java.util.Scanner

class A{

public static void mani(String args[])

{

        Scanner T = new Scanner(System.in);

        System .out.print("Digite seu nome");

        String n=T.next();

        System.out.print("Ola" + n);

}

}
```

Neste programa foi possível importar a classe Scanner do pacote util. Poderíamos apenas importar todas as classes do pacote util usando: import java.util.*

Criando pacotes em java:

Para criar um pacote pessoal em java precisamos seguir os seguintes passos:

1. Cria uma pasta com o nome do pacote, por exemplo pac1
2. Dentro do directório crie uma classe qualquer indicando o nome do pacote:

```
package pac1;

public class A{

   public void show(){ System.out.print("Ola"); }

}
```

3. Fora do directório crie uma classe qualquer que importa o pacote pac1 e faz o uso da classe A que está dentro do pacote:

```
import java.pac1.*;

class P

{  public static void main(String z[])

   {

         A obj = new A();

             obj.show();

   }

         }
```

Sendo assim, é possível criar o objecto de A em P ou mesmo fazer a herança de A em P.

Lista de fontes relevantes

Leitura nr 1:

Referência completa: Pacotes - organizando suas classes e bibliotecas

http://www.caelum.com.br/apostila-java-orientacao-objetos/pacotes-organizando-suas-classes-e-bibliotecas Site visitado aos 13 de Novembro de 2014 pelas 22:16 Este website oferece uma variedade de técnicas sobre o processo de criação de pacotes em java, dando exemplos concretos e claros.

Conclusão

Os pacotes permitem agrupar várias classes numa só entidade. Podemos aceder as diferentes classes do java através dos seus pacotes bem como criar nossos próprios pacotes

Avaliação

1. Crie um pacote P que contenha uma classe A. Use uma outra classe para importar o pacote P e fazer o uso da classe A.

Actividade 2 - Gestão de erros e excepções

Introdução

Todos os programas escritos em java devem ser compilados antes da sua execução, onde são corrigidos os erros detectados pela linguagem. Apesar da compilação bem sucedida podem ocorrer erros durante a execução do programa pois erros como divisão por zero não são detectados durante a compilação. Sendo assim, a presente actividade visa capacitar os estudantes na gestão dos erros de execução, de modo a poder formatar estes erros como bem entender.

Detalhes da actividade

Os erros em java são geridos através de exepções definidas pelas diferentes classes, onde cada classe representa um determinado tipo de erro:

- ArithmeticException (Erros aritméticos)
- NullPointerException (Ponteiro apontando para um espaço vazio)
- ArrayIndexOutOfBoundsException (Indice do array fora da posição)
- Exception (Erros de natureza global)

A classe Exception faz menção a todos os tipos de erros contidos nas classes anteriores.

O Bloco Try...Catch

Para fazer o tratamento de erros usamos o bloco try que contêm o conjunto de instruções e o bloco try que contêm o tratamento de erros, isto é, o que vai acontecer se alguma excepção (erro) for detectada. Para cada bloco try podemos ter vários blocos catch.

O programa a seguir pretende evitar a divisão por zero de um número qualquer.

```
import java.lang.Exception;

class exc extends Exception
```

```
{
public static void main(String args[])
{
   int a,b,r;
   try
   {
         a=0;
         b=7;
         r=b/a;
   }
   catch(ArithmeticException e)
   {
         System.out.println("Divisão por zero");
   }
   catch(Exception e)
   {
         System.out.println("Erro qualquer!!!!");
   }
}
```

Lista de fontes relevantes

Leitura nr 1:

Referência completa: Excepções e controle de erros

http://www.caelum.com.br/apostila-java-orientacao-objetos/excecoes-e-controle-de-erros/ Site visitado aos 13 de Novembro de 2014 pelas 23:16

Esta apostila descreve os aspectos relevantes sobre o processo de gestão de erros e as formas de criar classes de erros personalizadas.

Conclusão

Nesta unidade foram discutidas as principais técnicas de criação de pacotes bem como os mecanismos de gestão de erros durante a execução do programa.

Avaliação

1. Crie um pacote Px e nele adicione duas classes a sua escolha. Depois crie uma outra classe fora do pacote e importe todas as classes de Px.
2. Escreva um programa que permite detectar erros aritméticos e erros em arrays (caso o índice não exista no array).

Actividade 3 - Laboratório da Unidade 3

Exercícios laboratoriais da unidade 3

Laboratório 1

Objectivo do exercício

Perceber o processo de criação de pacotes em java

Implementar a técnica de importação de pacotes em Java

Recursos

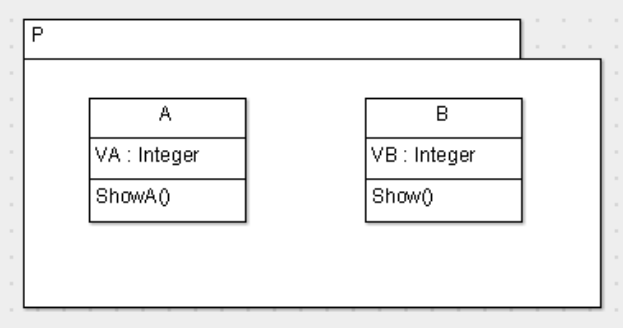
JKD 1.8 e Note pad

Tempo

5 horas

Descrição do exercício

Crie um pacote P e nele crie e compile duas classes A e B tal como ilustra a figura abaixo:

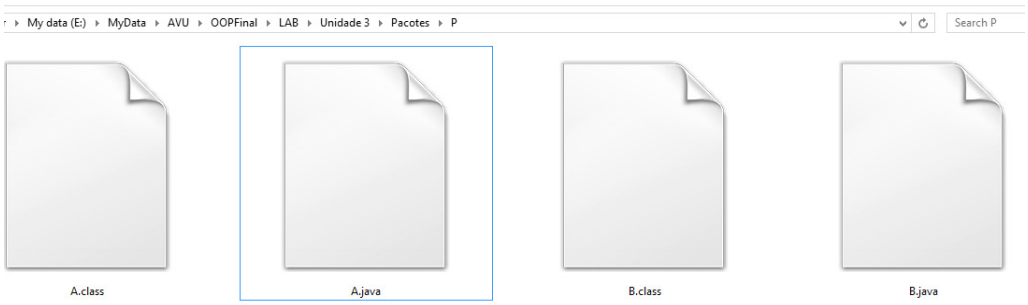


A seguir Crie uma outra classe Principal (fora do pacote P) que deve fazer a importação de P e fazer o uso das classes A e B. A classe A deve ser acedida através da herança e a classe B via objecto.

Resultados e Submissão

Proposta de solução

Passo 1: Crie uma pasta P e nela crie as seguintes classes em separado:




```
package P;
public class B{
      public int VB;
      public void showB(){
      System.out.println("Eu sou da classe B");
      }
}
```

```
package P;
public class A{
      public int VA;
      public void showA(){
      System.out.println("Eu sou da classe A");
      }
}
```

Passo 2: A seguir crie a classe principal fora do directório P e nela faça a importação do pacote P e faça a herança de A. Para a classe B será criado o objecto no main.

```
import P.*;
class Principal extends A
        {
           public static void main(String args[])
           {     //Chamando A através da herança
                 Principal objA = new Principal();
                 objA.VA=10; System.out.print("VA="+objA.VA);
                 //chamando B via objecto
                 B objB = new B();
                 objB.VB=15; System.out.print("VB="+objB.VB);
           }
        }
```

Output:

VA=10VB=15

No presente programa foi possível constatar que a classe Principal consegue facilmente aceder aos métodos e dados da classe A via herança e aos métodos e dados de B via objecto.

Submissão: A solução deve ser submetida no final da aula laboratorial. O estudante deve executar a classe e demonstrar a solução. Este pode ser submetido via email caso não termine a actividade no laboratório.

Critérios de avaliação

O presente exercício consiste na execução com sucesso do programa no laboratório:

Peso da avaliação 5%

Referências ou Links chaves

Balagurusamy, E. Programming with Java a primer. 4th Edition. Tata McGraw-Hill. 2010

Laboratório 2

Objectivo do exercício

Aplicar técnicas de gestão de erros durante a execução do programa

Recursos

JKD 1.8

Note pad

Tempo

5 horas

Descrição do exercício

Faça a simulação da detenção de erros em arrays (índice fora do intervalo – ArrayIndexOutOfBoundsException). Para tal crie um array com dimensão 3 ou mais e tente preencher um elemento com índice 3. Tendo em conta que o índice 3 está fora dos limites do array, o sistema irá compilar com sucesso mas durante a execução irá disparar uma excepção. A tarefa é fazer o tratamento desta excepção usando o bloco try e catch.

Resultados e Submissão

Proposta de solução

```
class Principal{
      public static void main(String args[])
      {
            int Arr[] = new int[3];
            try{
            Arr[0]=10;
            Arr[3]=6; //Erro, pois índice 3 está fora dos limites do
array
            }
            catch(ArrayIndexOutOfBoundsException e)
            { System.out.print("Indice fora do intervalo do array");}
            catch(Exception e2)
            {System.out.print("Erro qualquer");}
      }
}
```

O programa acima será compilado com sucesso, contudo, durante a sua execução o sistema irá disparar a excepção "ArrayIndexOutOfBoundsException" pois o índice 3 está fora dos limites do array (lembrando que se o array tem tamanho 3, então os índice vão de 0 a tamanho-1, neste caso 2.

Output:

Indice fora do intervalo do array

O outro bloco catch(Exception e2) será disparado caso ocorra um outro tipo de erro, durante a execução.

Submissão: A solução deve ser submetida no final da aula laboratorial. O estudante deve executar a classe e demonstrar a solução. Este pode ser submetido via email caso não termine a actividade no laboratório.

Critérios de avaliação

O presente exercício consiste na execução com sucesso do programa no laboratório:

Peso da avaliação 5%

Referências ou Links chaves

Balagurusamy, E. Programming with Java a primer. 4th Edition. Tata McGraw-Hill. 2010

Resumo da Unidade

Nesta unidade foi possível discutir as diferentes bibliotecas/pacotes em java e também o processo de criação de pacotes. Este conceito permite ao programador organizar melhor suas classes pois cada grupo pertence a um único pacote. O conceito de gestão de erros foi discutido, permitindo personalizar os erros do sistema. Foi também possível discutir as técnicas de tratamento de erros (excepções) que decorrem durante a execução de um programa, permitindo assim definir mensagens de erro definidas pelo programador.

Avaliação da Unidade

Instruções

A seguinte avaliação consiste em programar no computador os casos que seguem. Explique os resultados que obteve.

Critérios de Avaliação

O exercício 1 vale 6 pontos, o exercício 2 vale 6 pontos e o exercício 3 vale 8 pontos.

1. Escreva um programa que possui um pacote com uma classe que contém 1 método protected. Crie uma outra classe que importa a classe do pacote criado e chame este o método protected da classe no main através do objecto da classe derivada.
2. Escreva um programa que retorna uma excepção caso o índice do array seja inválido (fora dos limites do array).
3. Considere a seguinte classe que se encontra dentro do pacote PX. class Produto

```
{
private  String ID, Nome;
private float quantidade;
}
```

   a. Crie um construtor com parâmetros que permite inserir dados e um método que permite visualizar os dados do Produto.

   b. Crie um método não void Calcula() que retorna o valor a pagar do produto, sendo este igual a quantidade X 150 (Deve incluir IVA).

   c. Faça o uso desta classe fora do pacote.

   d. O que vai acontecer se o construtor do produto for privado?

Responda

1.
```
package pac1;
class Celio {
    protected void print(String S)
    {
      System.out.println(S);
    }
}
import pac1.*;
class Main extends Celio{
   public static void main(String x[])
   {
     Main obj=new Main();
     obj.print("Ola!!!");
   }
}
```

2.
```
public class Exemplo_GetMessage
{

   public static void main(String[] args) {
         try {
               int[] numero = new int[5];

               for(int i = 0; i <= 10; i++){
                     numero[i] = i;
                     System.out.println(i);
               }
         } catch (ArrayIndexOutOfBoundsException e) {
               System.out.println("Array fora do índice: "+e.
getMessage());
         }
   }
}
```

3.

```
package Px;

class Produto

{

private  String ID, Nome;

private float quantidade;

public Produto(String id, String N, float q){

   ID=id;

   Nome=N;

   quantidade = q;

}

public float Calcula(){

return (quantidade*150);

}

public void print(){

   System.out.print(ID+Nome+quantidade);

}

}

import Px.*;

class M extends Produto{

   public static void main(String[] args) {

M obj=new M("1000","ABEL",2.5);

obj.print();

System.out.print(obj.Calcula());

   }


}
```

Leituras e Outros Recursos

As leituras e outros recursos desta unidade encontram-se na lista de "Leituras e Outros Recursos do curso".

- Como utilizar package (pacotes) em Java https://www.youtube.com/watch?v=7snwqkMb8n0 Visitado aos 13 de Maio de 2016 Neste vídeo são demostrados os passos básicos para a criação de pacotes em java
- 060 Java nivel basico Exceções tratamento de erros no Java https://www.youtube.com/watch?v=XPOUzj1f1yQ Visitado aos 13 de Maio de 2016 Neste vídeo é demostrado o tratamento de erros em java e foi dado como exemplo a excepção: ArraIndexOutOfBounds.

Unidade 4. Introdução à programação GUI

Introdução à Unidade

Com o avanço das linguagens de programação, muitas empresas têm aderido aos software para gerir os seus processos. A linguagem Java permite também desenvolver aplicações GUI (Graphical User Interface) através de uma variedade de classes próprias. Cada uma destas classes permite representar objectos como caixas de textos, botões, listas, etc. O pacote principal é o awt que permite desenhar formulários simples, e mais tarde deu lugar ao pacote swing que trouxe mais recursos visuais não disponíveis no seu antecessor. Deste modo, na presente actividade serão discutidas as técnicas de desenho de aplicações GUI onde a posterior serão discutidos os procedimentos para adicção de eventos nestas. Na última actividade será discutida a técnica de desenho de imagens como rectângulos, rectas, circunferências, etc.

Objetivos da Unidade

Após a conclusão desta unidade, deverá ser capaz de:

- Desenhar e programar aplicações GUI
- Adicionar eventos básicos em objectos AWT ou SWING
- Programar diferentes figuras geométricas usando a classe Graphics

Termos-chave

AWT: Significa: Abstract Window Toolkit, Pacote que contem as classes básicas usadas no desenho de aplicações com interface gráfica.

GUI: Graphical User Interface, que são às aplicações que usam formulários, caixas de texto, etc onde o mouse é a ferramenta principal para a sua manipulação.

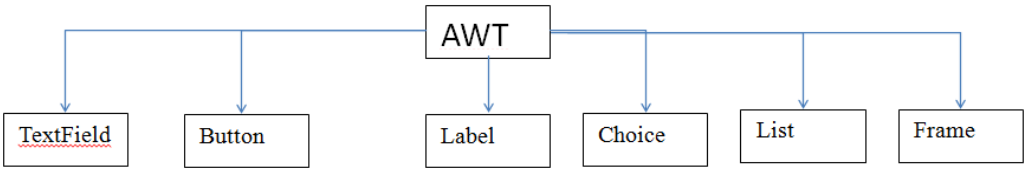
Eventos: Eventos são acções que resultam de ventos como clique, duplo click, etc, adicionadas aos diferentes objectos do formulário.

Actividades de Aprendizagem

Actividade 1 - Programação gráfica GUI

Introdução

A linguagem java possui um conjunto de classes que permitem desenhar aplicações GUI básicas. Essas classes estão contidas no pacote AWT como mostra a figura a seguir:

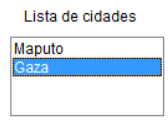


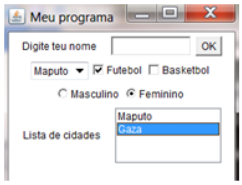
Cada uma destas classes representa um objecto e possui uma variedade de métodos que permitem manipular tais objectos.

A tabela a seguir apresenta alguns métodos básicos de cada uma das classes:

Classe	Construtor	Métodos	Como criar
TextField	TextField() TextField(String) TextField(String, int) TextField(int)	void setVisible(boolean) void setEnabled(boolean) void setText(String) String getText(String) Char getEchoChar(Char) void setEchoChar(Char) void selectAll() void select(int,int) boolean isEditable(boolean) void setEditable(boolean)	Caixa de texto Maputo Ex: TextField T1; T1=new TextField(10);
Button	Button() Button(String)	void setVisible(boolean) void setEnabled(boolean)	Botão de comando OK
		void setLabel(String) String getLabel(String)	Ex: Button B; B = new Button("OK");
Label	Label() Label(String) Label(String, int)	void setVisible(boolean) void setEnabled(boolean) void setText(String) String getText(String)	Label de texto Digite teu nome Ex: Label L1; L1 = new Label("Digite o teu nome")

Choice	Choice()	void setVisible(boolean)
		void setEnabled(boolean)
		void add(String)
		String getItem(int)
		int getSelectedIndex()
		String getSelectedItem()
		void select(int)
			Caixa de textos (Combobox)
			Maputo
			Maputo
			Sofala
			Choice ch;
			ch = new Choice();
			ch.add("Maputo");
			ch.add("Sofala");

List	List()
	List(int)
	List(int, boolean muiti_select)
		add(String)
		delItem(int)
		removeAll()
		int countItems();
		boolean allowsMultipleSelections();
		setVisible(boolean)
		setEnabled(boolean)
		getItem(int)
		getSelectedItem()
		select(int)
			Lista de textos
			 
			Ex:
			List L;
			L=new List();
			L.add("Maputo");
			L.add("Gaza");

Frame	Frame() Frame(String)	void add(Object) void setLayout(Object) void setTitle(String) void resize(int,int)	Janela principal que contêm todos os objectos  public class ABC extends **Frame** { }

	Checkbox(String) Checkbox(String, boolean)	setEnabled(boolean) setLabel(String) getLabel(String) getState() setState(boolean)	 Checkbox ck; ck = new Checkbox("Futebol");

Checkbox Group	CheckboxGroup()
	Checkbox(String,CheckboxGroup, boolean)
		setVisible(boolean)
		setEnabled(boolean)
		setCheckboxGroup(CheckboxGroup)
		getCheckboxGroup()
			Radio Buttons
			
			genero=new CheckboxGroup();
			masc=new Checkbox("Masculino",genero,false);
			fem=new Checkbox("Feminino",genero,true);

Detalhes da actividade

A presente actividade visa dotar aos estudantes de técnicas de como desenhar uma aplicação GUI usando o java AWT ou o SWING.

Pretende-se desenhar uma calculadora de dois números introduzidos pelo utilizador.

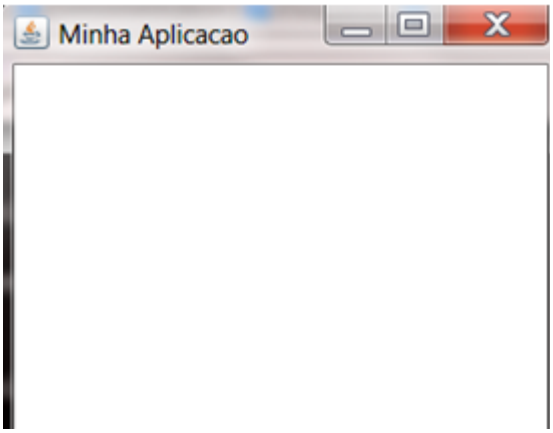
Passo 1: Desenhamos a janela (Frame)

```
import java.awt.*;

class APP extends Frame

{
```

```
        public APP()
        {
        }
        public static void main(String z[])
        {
              APP form = new APP();
                    form.setLayout(new FlowLayout());
                    form.resize(400,400);
                    form.setTitle("Minha Aplicacao");
                    form.show();
        }
}
```



Passo 2: Adicionamos os objectos na classe e inicializados no construtor da classe.

Normalmente declaram-se os objectos na classe e inicializam-se no construtor. Isso vai permitir que as caixas de textos,botões,etc estejam acessíveis para todos os métodos da classe.

import java.awt.*;

class APP extends Frame

```
{
        TextField txtn1,txtn2,txtR;
        Label lbln1,lbln2;
        Button B;
        public APP()
        {
```

```
                                        txtn1 = new TextField(10);
                                        txtn2 = new TextField(10);
                                txtR = new TextField(10);
                                        lbln1=new Label("Digite N1");
                                        lbln2=new Label("Digite N2");
                                        B = new Button("=");
                }
                public static void main(String z[])
                {
                        APP form = new APP();
                        form.setLayout(new FlowLayout());
                        form.resize(400,400);
                        form.setTitle("Minha Aplicacao");
                        form.show();
                }
        }
```

Passo 3: Adicionamos os objectos na janela (Frame)

Até agora os objectos apenas foram adicionados e inicializados na classe APP, contudo ainda não foram adicionados na janela (Frame) e ao fazermos a execução deste programa ele continuará mostrando uma janela simples como a anterior. Sendo assim, usa-se o método add() no construtor para adicionar cada um dos objectos na classe. Ora vejamos,

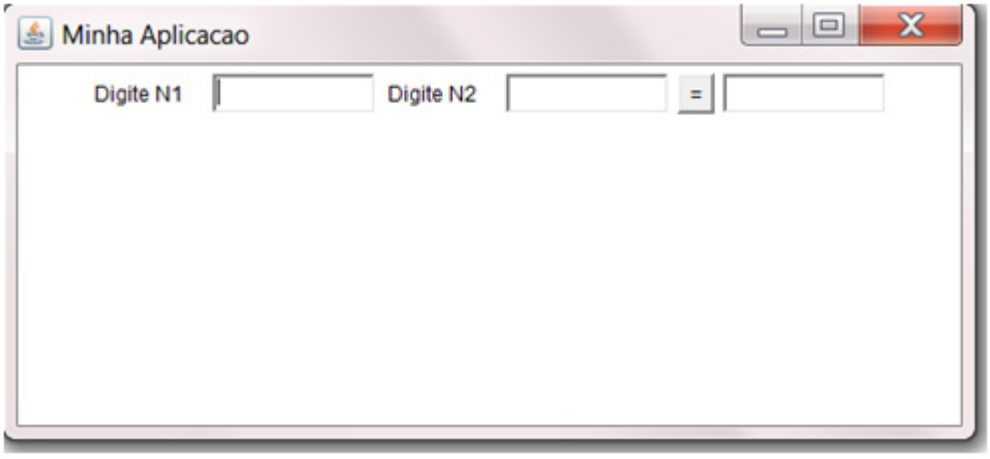
import java.awt.*;

class APP extends Frame

```
        {
                TextField txtn1,txtn2,txtR;
                Label lbln1,lbln2;
                Button B;
                public APP()
                {
                                        txtn1 = new TextField(10);
                                        txtn2 = new TextField(10);
```

```
                                        txtR = new TextField(10);
                                        lbln1=new Label("Digite N1");
                                lbln2=new Label("Digite N2");
                                        B = new Button("=");
                                        add(lbln1);
                                        add(txtn1);
                                        add(lbln2);
                                        add(txtn2);
                                        add(B);
                                        add(txtR);
                        }
                        public static void main(String z[])
                        {
                                        APP form = new APP();
                                form.setLayout(new FlowLayout());
                                        form.resize(400,400);
                                        form.setTitle("Minha Aplicacao");
                                        form.show();
                        }
        }
```

Note que, os objectos serão organizados na mesma ordem na qual eles foram adicionado no Frame:



Assim completamos o desenho da nossa primeira aplicação.

Conclusão

Nesta actividade foi possível criar aplicações GUI usando o java awt. Cada uma das classes awt representa um objecto (caixa de texto, botão, janela) e cabe ao programador inicializar e chamar os métodos de cada um destes objectos.

Avaliação

1. Desenhe uma calculadora que permite ao utilizador introduzir dois números.

Actividade 2 - Delegação de Eventos

Introdução

Em aplicações GUI os movimentos do mouse são controlados pelo sistema operativo, pois é ele quem decide que programa está ligado ao tal evento (click, duplo click, etc). Sendo assim, quando fazemos o click do mouse o sistema operativo armazena as coordenadas do cursor e estabelece a comunicação com o tal programa. O sinal que o programa recebe chamamos de evento. Em java os eventos são gerados através das classe que estão no pacote java.awt.event. Cada uma das classes está ligada a outras classes (interfaces) chamadas Listeners. Os Listeners conectam o objecto em causa (botão por exemplo) com o tal evento.

A tabela a seguir apresenta alguns eventos, seus listeners e os métodos disponíveis em Java.

Evento	Descrição	Listener	Métodos obrigatórios
ActionEvent	Clique sobre um objecto	ActionListener	void actionPerformed(ActionEvent x){}
TextEvent	Ao escrever na caixa de texto	TextListener	void textValueChanged(TextEvent e){}
KeyEvent	Ao accionar qualquer tecla no teclado	KeyListener	void keyPressed(KeyEvent x){} void keyReleased(KeyEvent x){} void keyTyped(KeyEvent x){}
MouseEvent	Ao accionar o mouse sobre o objecto	MouseListener	void mousePressed(MouseEvent e){} void mouseReleased(MouseEvent e){} void mouseEntered(MouseEvent e){} void mouseExited(MouseEvent e){} void mouseClicked(MouseEvent e){}
MouseMotionEvent	Ao mover o mouse	MouseMotion Listener	void mouseMoved(MouseMotionEvent x) void mouseDragged(MouseMotionEvent x)

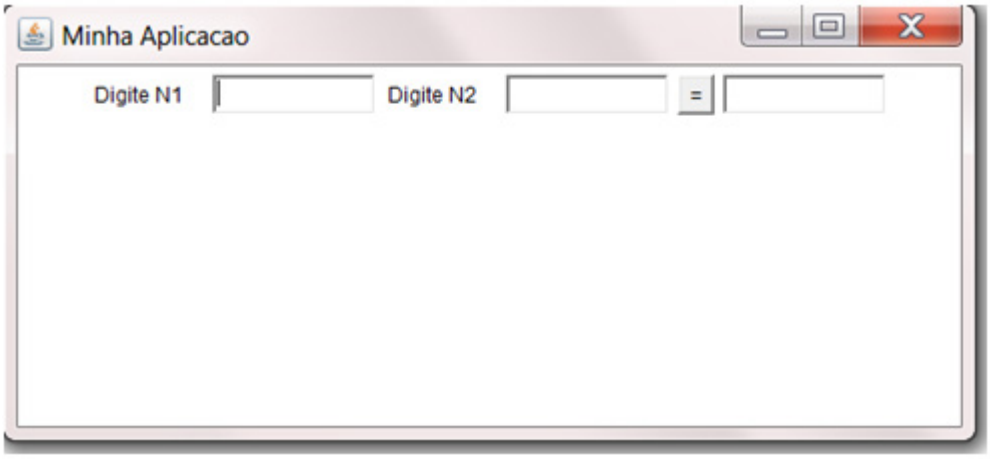
WindowEvent	Qualquer acção sobre a janela do formulário.		void windowOpened(WindowEvent x);
			void windowClosing(WindowEvent x);
			void windowClosed(WindowEvent x);
			void windowIconified(WindowEvent x);
			void windowDeiconified(WindowEvent x);
			void windowActivated(WindowEvent x);
			void windowDeactivated(WindowEvent x);

Existem outras tantas classes usadas para outros tipos de eventos, contudo, o mais importante é escolher o evento certo para cada objecto. Por exemplo, usa-se o ActionEvent (e o seu ActionListener) para os botões.

Detalhes da atividade

A presente actividade consiste em adicionar o evento ActionEvent aos objectos.

Existem outras tantas classes usadas para outros tipos de eventos, contudo, o mais importante é escolher o evento certo para cada objecto. Por exemplo, usa-se o ActionEvent (e o seu ActionListener) para os botões.



O primeiro passo consiste em escolher o evento para o botão. Como o botão deve responder ao click do mouse, temos duas opções:

MouseListener : possui o método void mouseClicked(MouseEvent e){} que é accionado quando se efectua o click sobre o mouse.

ActionListener: possui o método void actionPerformed(ActionEvent x){} que também é accionado quando o click do mouse é efectuado.

ATENÇÃO: Ao escolher o listener é obrigatório chamar todos os métodos obrigatórios (override), por mais que não façam nada.

Vejamos o programa:

```
import java.awt.*;

import java.awt.event.*;

class APP extends Frame implements ActionListener

{        TextField txtn1,txtn2,txtR;

         Label lbln1,lbln2;

         Button B;

         public APP()

         {

                                  txtn1 = new TextField(10);

                                  txtn2 = new TextField(10);

                                  txtR = new TextField(10);
```

```
                                        lbln1=new Label("Digite N1");
                                        lbln2=new Label("Digite N2");
                                        B = new Button("=");
                                        add(lbln1);
                                        add(txtn1);
                                        add(lbln2);
                                        add(txtn2);
                                        add(B);
                                        add(txtR);
                                        B.addActionListener(this);
        }
    public void actionPerformed(ActionEvent e){
        int v1 = Integer.parseInt(txtn1.getText());
        int v2 = Integer.parseInt(txtn2.getText());
        int R=v1+v2;
        txtR.setText(Integer.toString(R));
    }
        public static void main(String z[])
        {           APP form = new APP();
                    form.setLayout(new FlowLayout());
                    form.resize(400,400);
                    form.setTitle("Minha Aplicacao");
                    form.show();
        }
}
```

Nota-se o uso dos métodos de conversão de inteiro para String e vice-versa.

- Integer.toString() Converte um dado qualquer para String (pois as caixas de texto só recebem textos)
- Integer.parseInt() Converte um dado qualquer para int

Nota-se também o uso dos métodos das caixas de textos

- getText() Busca o conteúdo existente na caixa de texto
- setText() Armazena um conteúdo qualquer na caixa de texto

Actividade 3 - Desenho gráfico

Introdução

O desenho de aplicações GUI não consiste apenas no desenho de objectos pré-definidos como botões e caixas de texto, como também no desenho de qualquer figura geométrica a escolha do programador. Deste modo o java possui uma classe que tem métodos que permitem desenhar qualquer figura geométrica através de linhas, rectângulos e circuferências, a classe Graphics do pacote awt.

Para tal precisamos chamar o método void paint(Graphics g){...} que recebe o objecto do tipo graphics que permite desenhar.

A seguir alguns métodos da classe Graphics:

Classe Graphics

```
public abstract void drawLine(int, int, int, int);
public abstract void fillRect(int, int, int, int);
public void drawRect(int, int, int, int);
public abstract void clearRect(int, int, int, int);
public abstract void drawRoundRect(int, int, int, int, int, int);
public abstract void fillRoundRect(int, int, int, int, int, int);
public void draw3DRect(int, int, int, int, boolean);
public void fill3DRect(int, int, int, int, boolean);
public abstract void drawString(java.lang.String, int, int);
```

Detalhes da atividade

Pretende-se desenhar a figura abaixo:

Para tal, usamos o seguinte programa:

```
import java.awt.*;
import java.awt.event.*;
class APP extends Frame
{
   public void paint(Graphics g)
   {
         g.drawString("Bem vindo ao meu sistema",100,100);
         g.setColor(Color.yellow);
         g.fillOval(20,200,100,100);
         g.drawOval(20,200,100,100);
         g.setColor(Color.blue);
         g.drawRect(200,200,150,100);
   }
         public static void main(String z[])
         {           APP form = new APP();
                     form.setLayout(new FlowLayout());
               form.resize(400,400);
                     form.setTitle("Minha Aplicacao");
                     form.show();
         }
```

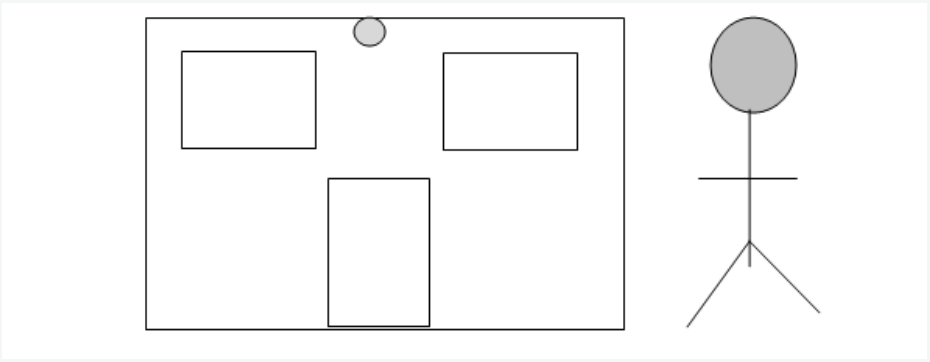
Preste atenção na forma como são chamados os métodos de Graphics: respeitando definição dos seus parâmetros

Conclusão

A presente actividade visava perceber o processo e desenho de figuras geométricas usando alguns métodos da classe Graphics. Várias figuras podem ser combinadas, contudo, o mais importante é a definição exacta de coordenadas.

Avaliação

1. Escreva o programa que permite desenhar a seguinte figura:



Actividade 4 Laboratório da Unidade 4

Aulas laboratoriais da unidade 4

Laboratório 1

Objectivo do exercício

Perceber como programar o desenho de formulários contendo botões, caixas de texto, listas, etc.

Recursos

JKD 1.8

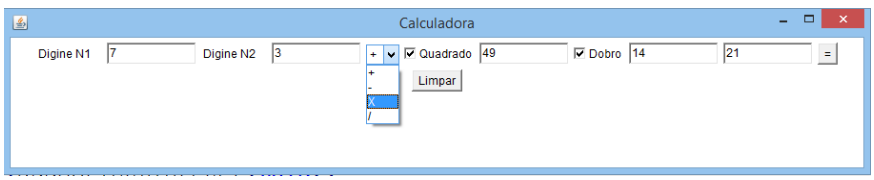
Note pad

Tempo

2 horas

Descrição do exercício

Use o pacote AWT ou SWING para desenhar o seguinte formulário:



Resultados e Submissão

Proposta de solução

```
import java.awt.*;

import java.awt.event.*;

public class Prog1 extends Frame
implements ActionListener,ItemListener{

      TextField
txtn1,txtn2,txtquad,txtdobro,txtr;

      Checkbox chkq,chkd;

      Label l1,l2;

      Choice ch; Button b,clr;

      public Prog1(){

      l1=new Label("Digine N1");

      l2=new Label("Digine N2");

      txtn1 = new TextField(10);

      txtn2 = new TextField(10);

      txtquad = new TextField(10);

      txtdobro = new TextField(10);

      txtr = new TextField(10);

      chkq = new Checkbox("Quadrado");

      chkq.addItemListener(this);

      chkd = new Checkbox("Dobro");

      chkd.addItemListener(this);

      b = new Button("=");

      clr = new Button("Limpar");

      b.addActionListener(this);
```

```
public static void
main(String g[])

      {

            Prog1
form=new Prog1();

            form.
setLayout(new
FlowLayout());

            form.
resize(400,400);

            form.
setTitle("Calculadora");

            form.
setVisible(true);


      }

}
```

```
      clr.addActionListener(this);

      ch=new Choice();ch.add("+");ch.ad-
d("-");ch.add("X");ch.add("/");

      add(l1);add(txtn1);add(l2);add(tx-
tn2);add(ch);add(chkq);add(txtquad);ad-
d(chkd);add(txtdobro);add(txtr);

      add(b);add(clr);

      }
```

Submissão: A solução deve ser submetida no final da aula laboratorial

Critérios de avaliação

O presente exercício consiste na execução com sucesso do programa no laboratório:

Peso da avaliação 2%

Referências ou Links chaves

Harvey M. Deitel. Java: Como Programar. 6 ed. São Paulo: Pearson education do Brasil, 2005

Balagurusamy, E. Programming with Java a primer. 4th Edition. Tata McGraw-Hill. 2010

Laboratório 2

Objectivo do exercício

Perceber como adicionar eventos nos diferentes objectos tendo em conta um problema específico.

Recursos

JKD 1.8

Note pad

Tempo

5 horas

Descrição do exercício

Considerando o formulário desenvolvido na actividade anterior, adicione eventos no objectos de modo que:

- Ao accionar no botão "=", o sistema calcula a operação entre N1 e N2 de acordo com a opção escolhida nas listas (+, -, X ou /).
- Ao accionar a check box Quadrado o sistema calcula o quadrado de N1
- Ao accionar a checkbox Dobro o sistema calcula o dobro de N1
- Ao accionar o botão " limpar", o sistema esvazia as caixas de texto N1 e N2.

Resultados e Submissão

Proposta de solução

```
import java.awt.*;

import java.awt.event.*;

public class Prog1 extends Frame
implements ActionListener,ItemListener{

      TextField
txtn1,txtn2,txtquad,txtdobro,txtr;

      Checkbox chkq,chkd;

      Label l1,l2;

      Choice ch; Button b,clr;

      public Prog1(){

      l1=new Label("Digine N1");

      l2=new Label("Digine N2");

      txtn1 = new TextField(10);

      txtn2 = new TextField(10);

      txtquad = new TextField(10);

      txtdobro = new TextField(10);

      txtr = new TextField(10);

      chkq = new Checkbox("Quadrado");

      chkq.addItemListener(this);
```

```
public void
actionPerformed(ActionEvent
g){

            if(g.
getSource()==b)

            {

                  int n1
= Integer.parseInt(txtn1.
getText());

                  int n2
= Integer.parseInt(txtn2.
getText());

                  String sel
= ch.getSelectedItem();

                  int R=0;

                  if(sel.
equals("+")){


R=n1+n2;

                   }

                   else
if(sel.equals("-")){
```

```
chkd = new Checkbox("Dobro");
      chkd.addItemListener(this);
      b = new Button("=");
      clr = new Button("Limpar");
      b.addActionListener(this);
clr.addActionListener(this);
      ch=new Choice();ch.add("+");ch.
add("-");ch.add("X");ch.add("/");
add(l1);add(txtn1);add(l2);add(txt-
n2);add(ch);add(chkq);add(txtquad);ad-
d(chkd);add(txtdobro);add(txtr);
      add(b);add(clr);
      }
      public static void main(String
g[])
      {
            Prog1 form=new Prog1();
form.setLayout(new FlowLayout());
            form.resize(400,400);
            form.setTitle("Calculado-
ra");
            form.setVisible(true);
      }
//Continua na outra coluna --→
```

```
R=n1-n2;
                    }
                    else
if(sel.equals("X")){
R=n1*n2;
                    }
                    else
if(sel.equals("/")){
                      R=n1/
n2;
                    }
txtr.setText(Integer.
toString(R));
            }
            else if(g.
getSource()==clr){
                    txtn1.
setText("");
                    txtn2.
setText("");
            }
}
      public void itemStat-
eChanged(ItemEvent e){
                    if(e.get-
Source()==chkq)
                    {
```

```
                                int
n1 = Integer.parseInt(txtn1.
getText());

                                int
R=n1*n1;

                                tx-
tquad.setText(Integer.toS-
tring(R));



                        }

                        else if(e.
getSource()==chkd){

                                int
n1 = Integer.parseInt(txtn1.
getText());

                                int
R=n1*2;

                                tx-
tdobro.setText(Integer.toS-
tring(R));

                        }



        }


}
```

Submissão: A solução deve ser submetida no final da aula laboratorial. O estudante deve executar a classe e demonstrar a solução. Este pode ser submetido via email caso não termine a actividade no laboratório.

Critérios de avaliação

O presente exercício consiste na execução com sucesso do programa no laboratório:

Peso da avaliação 4%

Referências ou Links chaves

Harvey M. Deitel. Java: Como Programar. 6 ed. São Paulo: Pearson education do Brasil, 2005

Balagurusamy, E. Programming with Java a primer. 4th Edition. Tata McGraw-Hill. 2010

Laboratório 3

Objectivo do exercício

Compreender as diferentes técnicas de desenho de figuras geométricas usando a classe Graphics

Associar o desenho de gráficos aos eventos do java

Recursos

JKD 1.8

Note pad

Tempo

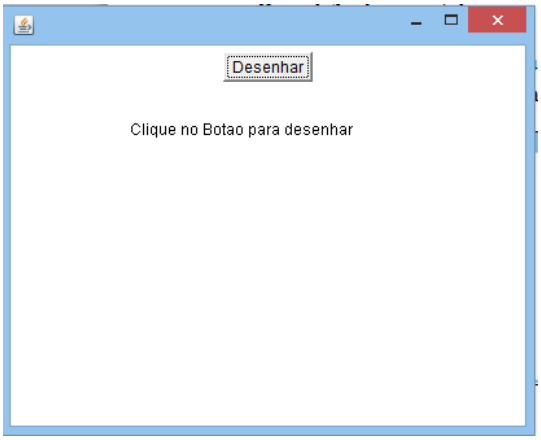
3 horas

Descrição do exercício

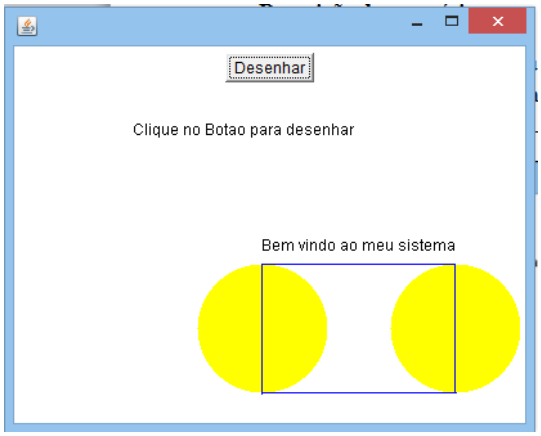
Use o pacote AWT ou SWING para desenhar a figura abaixo

Ao accionar o botão, a figura é desenhada

Antes de accionar o botão



Após accionar o botão



Resultados e Submissão

Proposta de solução

```
import java.awt.*;

import java.awt.event.*;

public class ABC extends Frame
implements ActionListener{

Button b1;

      public ABC()

      {

            b1=new
Button("Desenhar");

            add(b1);


b1.addActionListener(this);

      }

      public void
paint(Graphics g)

      {
g.drawString("Clique no Botao
para desenhar",100,100);

      }
```

```
public void actionPerformed(Action-
Event e)

      {     if(e.getSource()==b1)

            {

                  Graphics g = get-
Graphics();

                  g.drawString("Bem
vindo ao meu sistema",200,190);

                  g.setColor(Color.
yellow);

                  g.fill-
Oval(300,200,100,100);

                  g.fill-
Oval(150,200,100,100);

                  g.setColor(Color.
blue);

                  g.draw-
Rect(200,200,150,100);

            }

      }

      public static void main(String
x[])

      {

            ABC f1=new ABC();

            f1.setLayout(new FlowLay-
out());

            f1.resize(900,900);

            f1.show();

      }
```

Submissão: A solução deve ser submetida no final da aula laboratorial. O estudante deve executar a classe e demonstrar a solução. Este pode ser submetido via email caso não termine a actividade no laboratório.

Critérios de avaliação

O presente exercício consiste na execução com sucesso do programa no laboratório:

Peso da avaliação 4%

Referências ou Links chaves

Harvey M. Deitel. Java: Como Programar. 6 ed. São Paulo: Pearson education do Brasil, 2005

Balagurusamy, E. Programming with Java a primer. 4th Edition. Tata McGraw-Hill. 2010

Resumo da Unidade

Esta unidade consistiu na discussão de técnicas de desenho e programação de aplicações GUI onde foram abordados os termos: Programação de aplicações usando o pacote AWT onde foi possível desenhar formulários e seus objectos. Na actividade 2 foi discutida a delegação de eventos onde foi possível adicionar o evento ActionListener aos botões e na actividade 3 foi discutido o desenho gráfico usando a classe Graphics onde foi possível desenhar diversas figuras geométricas. Com estas técnicas, os(as) estudanteoderão desenvolver diversas aplicações GUI para o uso comercial, tais como: os sistemas de gestão, jogos e simulação.

Avaliação da Unidade

Verifique a sua compreensão!

Avaliação da unidade: Aplicações GUI

Instruções

Use seu computador para programar as aplicações que se sequem

Critérios de Avaliação

Os exercicio 1 e 2 valem 5 pontos cada e o exercício 3 vale 10 pontos.

1. Desenhe um aplicação GUI que permite introduzir o nome e o sobrenome usando caixas de diálogo. Logo a seguir o sistema imprime os dados introduzidos.
2. Desenhe uma aplicação GUI que permite introduzir dois valores e o sistema calcula e imprime a média entre eles..
3. Programe uma calculadora aritimética onde o utilizador introduz dois valores e escolhe a operação (adição ou multiplicação) através de uma lista (Choice). Dependendo da operação, o sistema imprime o resultado na caixa de texto. Pode usar Frames ou Applets.

Responda

1.
```
import javax.swing.JOptionPane;

public class SaidaValoresInteiros

{

    public static void main(String[] args) throws
NumberFormatException {

        int valorA = Integer.parseInt(JOptionPane.
showInputDialog("Informe o valor A: "));

        int valorB = Integer.parseInt(JOptionPane.
showInputDialog("Informe o valor B: "));

        int resultado = valorA * valorB;

        JOptionPane.showMessageDialog(null,"Resultado final:
"+resultado,"Resultado",JOptionPane.INFORMATION_MESSAGE);


    }

}
```

2.
```
import javax.swing.JOptionPane;

public class SaidaValoresInteiros

{

    public static void main(String[] args) throws
NumberFormatException {

        int valorA = Integer.parseInt(JOptionPane.
showInputDialog("Informe o valor A: "));

        int valorB = Integer.parseInt(JOptionPane.
showInputDialog("Informe o valor B: "));

        int resultado = valorA * valorB;

        JOptionPane.showMessageDialog(null,"Resultado final:
"+resultado,"Resultado",JOptionPane.INFORMATION_MESSAGE);

    }

}
```

3.
```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
/*
<applet code="ap2.class" width="300" height="300" align="center">
</applet>
*/
public class ap2 extends Applet implements ActionListener
{
   TextField t1,t2,t3;
   Choice ch;
   Button b1,b2;
   public void init()
   {
         t1=new TextField("0");
         t2=new TextField("0");
         t3=new TextField("Result");
         b1=new Button("=");
         b2=new Button("Reset");
         ch=new Choice();
         ch.add("+");
         ch.add("X");
         add(t1);
         add(ch);
         add(t2);
         add(t3);
         add(b1);
         add(b2);
         b1.addActionListener(this);
         b2.addActionListener(this);
```

```
    }
    public void actionPerformed(ActionEvent e)
    {
          if(e.getSource()==b1)
          {
                int v1=Integer.parseInt(t1.getText());
                int v2=Integer.parseInt(t2.getText());
                if(ch.getSelectedItem().equals("+"))
                {
                      int sum=v1+v2;
                      t3.setText(Integer.toString(sum));
                }
                else if(ch.getSelectedItem().equals("X"))
                {
                      int sum2=v1*v2;
                      t3.setText(Integer.toString(sum2));
                }
          }
          else if(e.getSource()==b2)
          {
                t1.setText("0");
                t2.setText("0");
          }
}
```

Avaliação do Módulo

Instruções

Responda com atenção as questões da avaliação. Todas elas devem ser respondidas no papel e sem o apoio de qualquer instrumento ou manual. As questões são do tipo mutilpa escolha e apenas uma alternativa é correta.

Critérios de avaliação

Cada uma das questões vale 2 pontos, sendo que o total de questões equivale a 20.

Avaliação Final 1

1. Considere a expressão: Qual o output sendo x=0?

```
if(x>=0)
if(x>0) System.out.print("Maior"); else  System.out.print("Menor");
```

a) Maior b) Menor c) Maior e Menor d) Nenhuma das alternativas

2. Na estrutura do Java podemos concluir que:

   a) Apénas 1 método main é permitido por ficheiro

   b) Não podemos ter uma classe sem main()

   c) Não podemos ter duas classes no mesmo ficheiro

   d) Todas alternativas estão correctas.

3. Considere as seguintes afirmações:

   a) Um método estático só pode aceder a dados estáticos

   b) Podemos usar um objecto para aceder a um método estático.

   c) Podemos usar uma classe para aceder a um método estático

   d) Todas afirmações estão correctas

4. Qual será o output do seguinte trecho de um programa:

```
float f=5, g=2;
int R=f/g;
System.out.print(R);
```

a) 2.5 b) 2 c) 3 d)erro

5. A sobrecarga consiste em ter:

   a) Dois métodos com o mesmo nome e diferença tipo (void/não void).

   b) Dois métodos com o mesmo nome mas com parâmetros diferentes.

   c) Dois métodos totalmente iguais.

   d) Nenhuma das alternativas

6. Considere as seguintes afirmações:

   a) Em herança, uma classe pode herdar(extends) apenas uma classe.

   b) Em herança, uma interface não pode fazer o implements de outra.

   c) Em herança, uma classe herdar uma outra classe (extends) e implementar uma interface (implements)

   d) Todas alternativas estão correctas

7. Se declaramos um método como privado:

   a) Não pode ser acedido fora da classe

   b) Pode ser acedido por uma sub classe que está no mesmo pacote

   c) Nenhuma das alternativas está correcta

   d) A alternativa a) e b) estão correctas

8. Qual será o output do seguintes bloco de instruções:

```
int a=0,b=100;
System.out.print( (a>0) || (b>=100) );
```

   a) true b) 1 c) 0 d) 100

9. Quantos erros existem na instrução que se segue?

```
int a=0; b=9;
System.out.write(a+b)
```

   a) 1 b)2 c) 3 d)4

10. Qual será a visibilidade de uma variável se ela for declarada sem public, private ou protected ?

    a) private b) protected c) friend d) public

Responda

Nenhuma das alternativas

1. Apénas 1 método main é permitido por ficheiro
2. Todas afirmações estão correctas
3. erro
4. Dois métodos com o mesmo nome mas com parâmetros diferentes.
5. Todas alternativas estão correctas
6. A alternativa a) e b) estão correctas
7. true
8. 3
9. friend

Avaliação do Módulo

Avaliação Final 2

Instruções

Responda com clareza as instruções que se seguem.

Critérios de avaliação

Cada uma das questões vale 2 valores, talizando 20 valores.

1. Questões de "Cultura Geral" Java
   a) Quais as implicações de declarar uma classe como final?
   b) O que acontece quando declaramos um construtor como privado?
   c) Como é que fazemos para obrigar que uma classe seja herdada?
   d) Como é que evitamos que uma classe tenha sub-classes?
   e) Mostre como é que resolve o problema do override em herança.
   f) Explique o significado de: "public static void main(String args[])"
   g) O que podemos fazer numa classe abstracta que não podemos fazer numa interface?
   h) Será que podemos declarar duas variáveis dentro da mesma classe?
   i) Explique as razões do main ser sempre declarado como static?
   j) Será que podemos ter dois métodos iguais dentro da mesma classe?

Responda

a. Não pode ser herdada

b. Não se pode criar objecto da classe

c. Decramaos como abstracta

d. Declaramos como final

e. Criando um método que chama o método sobreposto através da palavra reservada "super".

f. public - visibilidade pública

   static - Pertence a classe e não aos objectos

   void - Não retorna nada

   String args[] - Array de argumentos recebidos do exterior

g. Declarar métodos não abstractos

h. Não

i. Deve ser o primeiro a ser carregado, antes da criação dos objectos, assim ele deve pertencer a classe e puder chamar os outros métodos.

j. Sim, é o conceito de sobrecarga.

Avaliação do Módulo

Avaliação Final 3

Instruções

Use o computador para responder as questões que se seguem

Critérios de avaliaçã

A execução dos programas com sucesso são a condição para obter a pontuação máxima de 20 pontos.

Avaliação do Módulo

Avaliação Final 3

Instruções

Use o computador para responder as questões que se seguem

Critérios de avaliaçã

A execução dos programas com sucesso são a condição para obter a pontuação máxima de 20 pontos.

Programe a seguinte aplicação:

A Universidade Pedagógica (UP) precisar de um software para gerenciar pagamentos de seus professores. Cada professor tem um nome de identificação e uma lista de 100 pagamentos. Criar um programa que armazena e exibe as informações de 4 professores. Depois que ele também deve exibir o montante total pago a todos os 4 professores

Responda

A: Class Teacher

```
{
public int ID;
public int payment[100];
public String Name;
public static int Total;
  public void store()
  {
   Scanner T = new Scanner(System.in);
   ID=T.nextInt();
   Name=T.next();
   for(int i=0;i<100;i++){
         payment[i]=T.nextInt();
   Total = Total+payment[i];
   }
```

```
  }
public void show()
{
   System.out.print(“Information”);
   System.out.print(“ID:”+ID+”Name:”+Name+”List of
payments”);
   for(int i=0;i<100;i++){
        System.out.println(payment[i]);
}
}
public static void main(String args[])
{   Teacher T1,T2,T3,T4;
T1.store(); T2.store(); T3.store();T4.store();T5.store();
System.out.print(“Total amount:”+T1.Total);
}
}
}
```

Leituras e outros Recursos

As leituras e outros recursos desta unidade encontram-se na lista de Leituras e Outros Recursos do curso.

- Beginner Java - Intro to Swing (GUI) - Lesson 28 https://www.youtube.com/watch?v=uXitpup0l-s Visitado aos 13 de Maio de 2016 Neste vídeo o autor demostra como criar formulários usando o pacote swing. Este permite perceber o quanto é fácil desenhar formulários do pacote swing.
- Beginner Java - Intro to Swing (GUI) - Lesson 28 https://www.youtube.com/watch?v=2ly7d_uMlrI Visitado aos 13 de Maio de 2016 Este vídeo é a continuação do anterior, onde o autor apresenta de forma muito simples o método de adição de eventos aos objectos, com destaque à interface ActionListener.

Resumo do módulo

Neste módulo foram discutidas as bases teóricas e práticas da programação orientada a objectos, tendo como base a linguagem Java.

Na Unidade 0 foi realizado um teste diagnóstico de modo a verificar os pré-requisitos do módulo (programação estrututrada). Na Unidade 1, actividade 1 foram discutidos as principais diferenças entre uma linguem estruturada e uma linguagem orientada a objecto através de um quadro comparativo com o enfoque na forma como os programas são estruturados e executados. Na actividade 2 foi apresentado o conceito OOP e suas características com recurso a exemplos, leituras adicionais e videos ilustrativos. Na actividade 3 foram apresentadas (em forma de trabalho prático) as estruturas de controle (decisão, repetição e salto) em java, também na base de exercícios práticos, por se tratar de uma revisão (discutido na programação estruturada).

Na unidade 2 diz respeito à implementação das classes e objectos em Java onde na actividade 1 foram discutidas os mecanismos de criação de objectos e sua inicialização, na actividade 2 foi discutidos os métodos e dados estáticos e na actividade 3 foram apresentadas as diferentes formas de herança e as visibilidades existentes em java.

Na unidade 3 foram implementados os conceitos de pacotes (ou bibliotecas) em java onde discutiram-se os principais pacotes e a seguir as técnicas de criação de pacotes pelo programador. Nesta unidade foi discutido o processo de tratamento e gestão de erros/ excepções em java, dando-se ênfase às principais classes de gestão de erros e também à gestão de excepções num caso prático.

A unidade 4 foi reservada ao desenvolvimento de aplicações GUI, onde primeiro foram discutidas as técnicas de desenho de formulários e seus elementos(caixas de texto, botões, labels, etc) e depois os métodos de adição de eventos às aplicações, seguindo-se o processo de desenho de figuras geométricas.

Conclusão do módulo

As linguagens orientadas a objetos suportam os princípios de classes, objetos, herança, abstração, encapsulamento e polimorfismo. O presente módulo visava abordar os técnicas básicas da implementação de tais princípios onde usou-se a linguagem Java para compilar e executar os programas. Deste modo, foi possível desenvolver aplicações CLI (Command Line Interface) e aplicações GUI (Graphical User Interface) de modo a implementar na prática os conceitos OOP na resolução de problemas.

Referências do Curso

- Oracle Java Documentation The Java ™ Tutorials http://docs.oracle.com/javase/tutorial/java Site visitado aos 25 de Outubro de 2015 pelas 23:40
- ·Balagurusamy, E. Programming with Java a primer. 4th Edition. Tata McGraw-Hill. 2010
- Buyya, Rajkumar. Object-oriented Programming with Java: Essentials and Applications. 2009
- Harvey M. Deitel. Java: Como Programar. 6 ed. São Paulo: Pearson education do Brasil, 2005
- Naughton, Patrick, Dominando o Java, Guia Autorizado da Sun Microsystems, Editora Makron Books, 1997.

Sede da Universidade Virtual africana

The African Virtual University
Headquarters

Cape Office Park

Ring Road Kilimani

PO Box 25405-00603

Nairobi, Kenya

Tel: +254 20 25283333

contact@avu.org

oer@avu.org

Escritório Regional da Universidade Virtual Africana em Dakar

Université Virtuelle Africaine

Bureau Régional de l'Afrique de l'Ouest

Sicap Liberté VI Extension

Villa No.8 VDN

B.P. 50609 Dakar, Sénégal

Tel: +221 338670324

bureauregional@avu.org