EN05170 - Programação de Computadores II

Introdução a Programação Orientada a Objetos

Prof. Dr. Lidio Mauro Lima de Campos

limadecampos@gmail.com

Universidade Federal do Pará – UFPA
ICEN
PPGCC

Plano de Ensino

- **PLANO DE ENSINO: EN05170 - PROGRAMACAO DE COMPUTADORES II**
- **CURSO : BACHARELADO EM SISTEMAS DE INFORMAÇÃO**
- **DOCENTE: LÍDIO MAURO LIMA DE CAMPOS**

- **Ementa:**
- Conceitos do paradigma da programação orientada a objetos. Abstração. Introdução a classes e objetos. Agregação e Composição de objetos. Encapsulamento Herança. Polimorfismo. Tratamento de Exceções. Projeto orientado a objetos. Linguagem de programação orientada a objetos.

Plano de Ensino

- **Objetivos**
- **Geral**
- Compreender os conceitos de programação orientada a objetos.
- Projetar, desenvolver programas utilizando o paradigma de programação orientado a objetos e uma linguagem de programação orientada a objetos.
- **Específicos**
-
- Propiciar ao aluno uma adaptação (transição) entre a programação estruturada e a programação orientada a objetos
- Explicar os conceitos básicos do paradigma de programação orientado a objetos.
- Conceituar classes, objetos e interação entre objetos.
- Conceituar abstração, encapsulamento, herança e polimorfismo.
- Realizar tratamento de exceções.
- Entender os fundamentos do Paradigma Orientado a Objetos;
- Aprender uma linguagem de Programação Orientada a Objetos

Plano de Ensino

- **Conteúdo Programático**
- Conceitos do paradigma da programação orientada a objetos.
- Introdução a classes e objetos. Introdução ao Diagrama de Classes da UML
- Atributos, métodos e interação entre objetos.
- Sintaxe de linguagem de programação orientada a objetos
- O que é abstração em orientação a objetos
- Classes e Métodos; Encapsulamento e Sobrecarga; Sobreposição de Métodos; Construtores
- Agregação e Composição de objetos.
- Encapsulamento
- Classes Abstratas e Interfaces
- Herança e Polimorfismo.
- Tratamento de Exceções.

Plano de Ensino

- **Metodologia de Ensino Utilizada:**

- O curso será baseado em aulas expositivas com auxílio do quadro e projetor multimídia.
- Para fixação dos tópicos estudados, os alunos receberão, ao longo do curso, tarefas para entrega em via SIGAA. Por fim, destacam-se as aulas práticas nos laboratórios de informática para fixação dos conteúdos através de do uso de ambientes de desenvolvimento (IDEs).
- Cópias das transparências apresentadas em aula serão disponibilizadas em arquivos PDF após as aulas, no SIGAA. Esse material deve ser utilizado como auxílio ao estudo e como complementação aos apontamentos realizados em aula. Espera-se que os alunos revisem em casa esse material, procurando compreender os conteúdos.

- **Recursos Didáticos**
- Notebook e Projetor multimídia em sala de aula. Quadro magnético e pincel para quadro magnético. Laboratório de informática.
-

Plano de Ensino

- **Critérios de Avaliação**
-
- O aluno será avaliado com base no desempenho nas provas, trabalhos, projetos e outras atividades. As provas, trabalhos e projetos serão avaliados com nota entre 0.0 e 10.0. Conforme regulamento da Universidade, a frequência às aulas é obrigatória. Ao longo do período, serão considerados para a avaliação:
-
- Duas provas: P1 e P2, Listas de Exercícios, Trabalho Prático e Escrito (TPE1)
-
- A média geral (MG) será obtida por meio da seguinte fórmula:
- MG = (P1 + P2 + TPE1)/3.0
- Dos Conceitos de Avaliação:
- Art. 178 (RG da UFPA). Para fins de avaliação qualitativa e quantitativa dos conhecimentos serão atribuídos aos alunos da graduação e da pós-graduação os seguintes conceitos, equivalentes às notas:
- EXC – Excelente (9,0 - 10,0) BOM – Bom (7,0 - 8,9) REG – Regular (5,0 - 6,9) INS – Insuficiente (0 - 4,9)

Plano de Ensino

- Art. 179 (RG da UFPA). Considerar-se-á aprovado o discente que, na disciplina ou atividade correspondente, obtiver o conceito REG, BOM ou EXC e pelo menos setenta e cinco por cento (75%) de frequência nas atividades programadas.
- § 1o O conceito SA (Sem Avaliação) será atribuído ao discente que não cumprir as atividades programadas.
- § 2o Registrar-se-á SF (Sem Frequência) no histórico escolar quando o discente não obtiver a frequência mínima exigida

- Art. 111. A Avaliação Substitutiva é uma oportunidade oferecida ao discente que não obteve conceito à aprovação na atividade curricular, mas com frequência mínima de setenta e cinco por cento.
- § 2º Os procedimentos e orientações para aplicação da avaliação substitutiva são definidos pelo professor da turma.
- § 3º O conceito final deverá ser substituído pelo novo conceito obtido na avaliação substitutiva, até cinco dias após a conclusão do processo.

Plano de Ensino

- Bibliografia

- **Básica**
- Peter Jandl Junior. **Java - Guia do Programador: Atualizado Para Java 16 - 4ª edição. NOVATEC. 2021.**
- Herbert Schildt. Java: a referência completa. ALTA BOOKS Editora. 2020.
- DEITEL, P.; DEITEL, H. **Java: how to program**. 10th ed. Perason, 2016. 1496 p.
- RAMNATH, S.; DATHAN, B. **Object-oriented analysis and Design**. New York: Springer, 2015. 440 p.
- **Complementar**
- BOOCH, G.; MAKSIMCHUK, R. A.; ENGLE, M. W.; YOUNG, B. J.; CONALLEN J.; HOUSTON, K. A. **Object-oriented analysis and design with applications**. 3rd ed. Addison Wesley, 2017. 720 p.
- FARRELL, J. **An object-oriented approach to programming logic and design**. 4th ed. Cengage Learning, 2012. 560 p.
- GOODRICH, M. T.; TAMASSIA, R. **Estruturas de dados e algoritmos em Java**. 5a ed. Porto Alegre: Bookman, 2013. 600 p.

Plano de Ensino

- Bibliografia

Agenda

- Introdução: Programação Estruturada x Programação Orientada a Objetos.
- Java
- Principais Plataformas Java
- A Máquina Virtual Java
- Coleta automática de lixo

Programação Estruturada x Programação Orientada a Objetos

- Como a maioria das atividades que fazemos no dia a dia, programar também possui modos diferentes de se fazer.
  - Esses modos são chamados de **paradigmas de programação.**
    - **programação orientada a objetos** (POO) e a programação estruturada....

- Quando começamos a utilizar linguagens como Java, C#, Python e outras que possibilitam o paradigma orientado a objetos, é comum errarmos e aplicarmos a programação estruturada achando que estamos usando recursos da orientação a objetos.

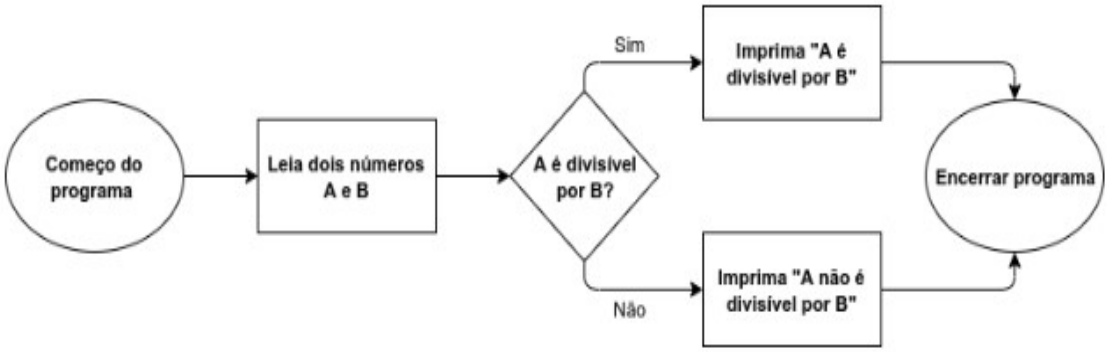
Programação Estruturada x Programação Orientada a Objetos

- Na programação estruturada, um programa é composto por três tipos básicos de estruturas:

  - **sequências:** são os comandos a serem executados
  - **condições:** sequências que só devem ser executadas se uma condição for satisfeita (exemplos: if-else, switch e comandos parecidos)
  - **repetições:** sequências que devem ser executadas repetidamente até uma condição for satisfeita (for, while, do-while etc)

Programação Estruturada x Programação Orientada a Objetos

- Essas estruturas são usadas para processar a entrada do programa, alterando os dados até que a saída esperada seja gerada. Até aí, nada que a programação orientada a objetos não faça, também, certo?
- A diferença principal é que na programação estruturada, um programa é tipicamente escrito em uma única rotina (ou função) podendo, é claro, ser quebrado em sub-rotinas.



Programação Estruturada x Programação Orientada a Objetos

- Além disso, o acesso às variáveis não possuem muitas restrições na programação estruturada.
  - uso da palavra-chave static, na linguagem C),
  - não se consegue dizer de forma nativa que uma variável pode ser acessada por apenas algumas rotinas do programa.
- Práticas danosas de programação: como o uso excessivo de variáveis globais. Vale lembrar que variáveis globais são usadas tipicamente para manter estados no programa, marcando em qual parte dele a execução se encontra.

Programação Estruturada x Programação Orientada a Objetos

- A **programação orientada a objetos** surgiu como uma alternativa a essas características da programação estruturada.
- O intuito da sua criação também foi o de aproximar o manuseio das estruturas de um programa ao manuseio das coisas do mundo real, daí o nome "objeto" como uma algo genérico, que pode representar qualquer coisa tangível.
- Esse novo paradigma se baseia principalmente em dois conceitos chave: **classes** e **objetos**.
- Todos os outros conceitos, igualmente importantes, são construídos em cima desses dois.

Java

- O JAVA é uma linguagem de programação de proposito geral , concorrente, baseada em classes e orientada a objetos. Projetada para ser simples o bastante para que a maioria s dos programadores se torne fluente na linguagem. Java tem relação com C e C++, porém é organizada de forma diferente, com vários aspectos de C e C++ omitidos e algumas ideias de outras linguagens incluídas (JUNIOR,2021).

Java

- Linguagem para eletrodomésticos, começo com o nome Oak , linguagem interpretada, portátil, interpretada.
- Concebida por James Gosling, Patrick Maughton, Chris Warth, Ed Frank e Mike Sheridan da SUN Microsystems, 1995.
- Familiar(sintaxe parecida com C – também sensitive case).
- Relativamente Simples.
- Robusta – Executa verificações em tempo de execução.
- Coleta automática de lixo.
- Independente de plataforma.
- Segura.
- Sintaxe fortemente tipada e rigorosa.
- Distribuída – Foi projetada visando o ambiente dinâmico da internet.

JAVA

- 80% dos celulares de todo mundo já vêm com Java instalado.
- Eo Google utiliza Java nos seus principais produtos como Android, Google Maps.

POO

- É A essência do Java
- POO pegou as melhores ideias da programação estruturada e combinou-as com vários conceitos novos.
- Em POO você define os dados e as rotinas que podem atuar sobre eles.
  - Características
    - Encapsulamento
    - Polimorfismo
    - Herança

Principais Plataformas Java

- Java S E (Standard Edition) – Integra os elementos padrão da plataforma e permite o desenvolvimento de aplicações de pequeno e médio porte. Inclui todas as APIs consideradas de base, além da máquina virtual padrão. Para Desktops em Geral.

- Java E E (Enterprise Edition)- Aplicações corporativas complexas. Adiciona APIs especificas aos elementos padrão da plataforma. Desenvolvimento Web .

Ambiente Java

- Java é uma linguagem independente de plataforma porque seus programas são compilados em um formato próprio denominado bytecodes, instruções de tamanho fixo que constituem a JVM, armazenados em arquivos de classe (.class).

- Em cada combinação específica de HW e SO deve existir uma JVM apropriada, capaz de interpretar os bytecodes ou transformá-los em código nativo que possa ser executado pelo processador do sistemas.

- A JVM sempre utiliza os serviços oferecidos pelo SO em uso. Assim o ambiente JAVA é composto com a JVM , sua API e com as classes da aplicação.

Recursos Necessários

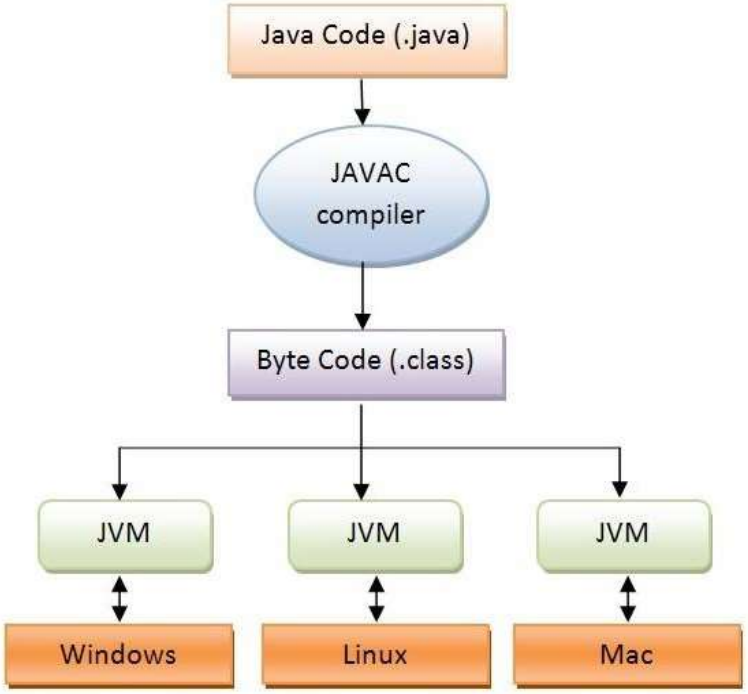
- Um ambiente de desenvolvimento mínimo para construção de aplicações Java, requer o JavaSE , conhecido como JDK (Java Development Kit), um conjunto útil de ferramentas de desenvolvimento padrão e que inclui o ambiente de execução JavaRE.

- JRE- Java Runtime Environment (Ambiente de Tempo de Execução Java)
  - É composto por Bibliotecas (APIs), pela Máquina Virtual Java (JVM) e outros componentes necessários para rodar aplicações desenvolvidas em ambiente Java.

- JDK– Java Development Kit (Kit de Desenvolvimento Java)
  - O JDK contém o JRE, além de um conjunto de ferramentas necessárias para o desenvolvimento de aplicações Java. As ferramentas incluem: compilador (javac.exe), depurador e outros utilitários.

A máquina virtual JAVA

- JVM - É um interpretador de código.
- JVM - Responsável pela execução de pilhas, gerenciamento de memória, threads e etc.
- Coleta automática de lixo (Garbage Collector).
- JVM é dependente de plataforma.
- Processa os byte codes que são independentes de plataforma.
- Cada JVM deve ser capaz de executar qualquer classe compilada java (bytecode).



Coleta de Lixo

- Memória alocada que não é mais utilizada deve ser liberada;
- Em outras linguagens, desalocar memória é responsabilidade do programador;
- Java provê um mecanismo que monitora e libera memória;
- Garbage Collector:
  - Checa e libera memória que não é mais utilizada;
  - Acontece automaticamente – não é necessário disparar um processo;
  - Pode variar entre JVMs.

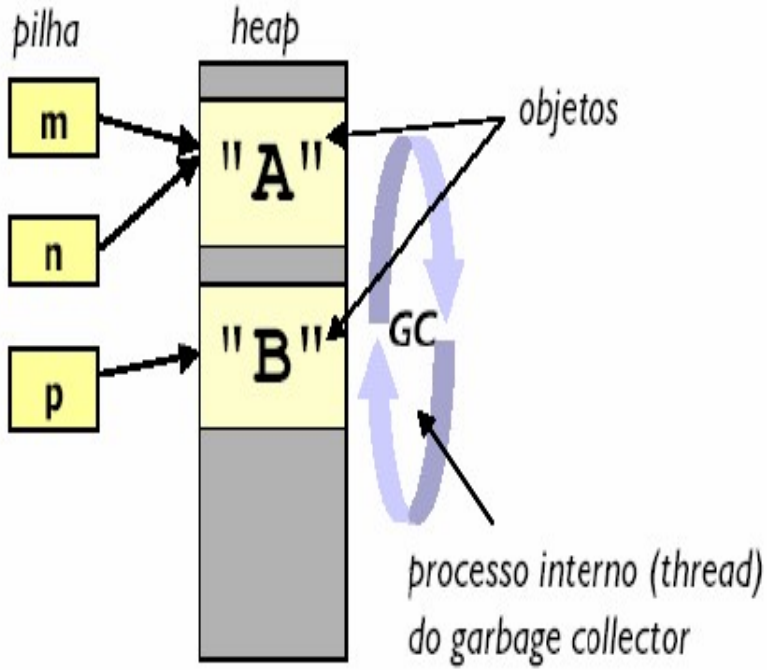
Coleta de Lixo

```
Mensagem m, n, p;

m = new Mensagem("A");

n = m;

p = new Mensagem("B");
```



Referências

Peter Jandl Junior. **Java - Guia do Programador: Atualizado Para Java 16 - 4ª edição. NOVATEC. 2021.**

Herbert Schildt. Java: a referência completa. ALTA BOOKS Editora. 2020.