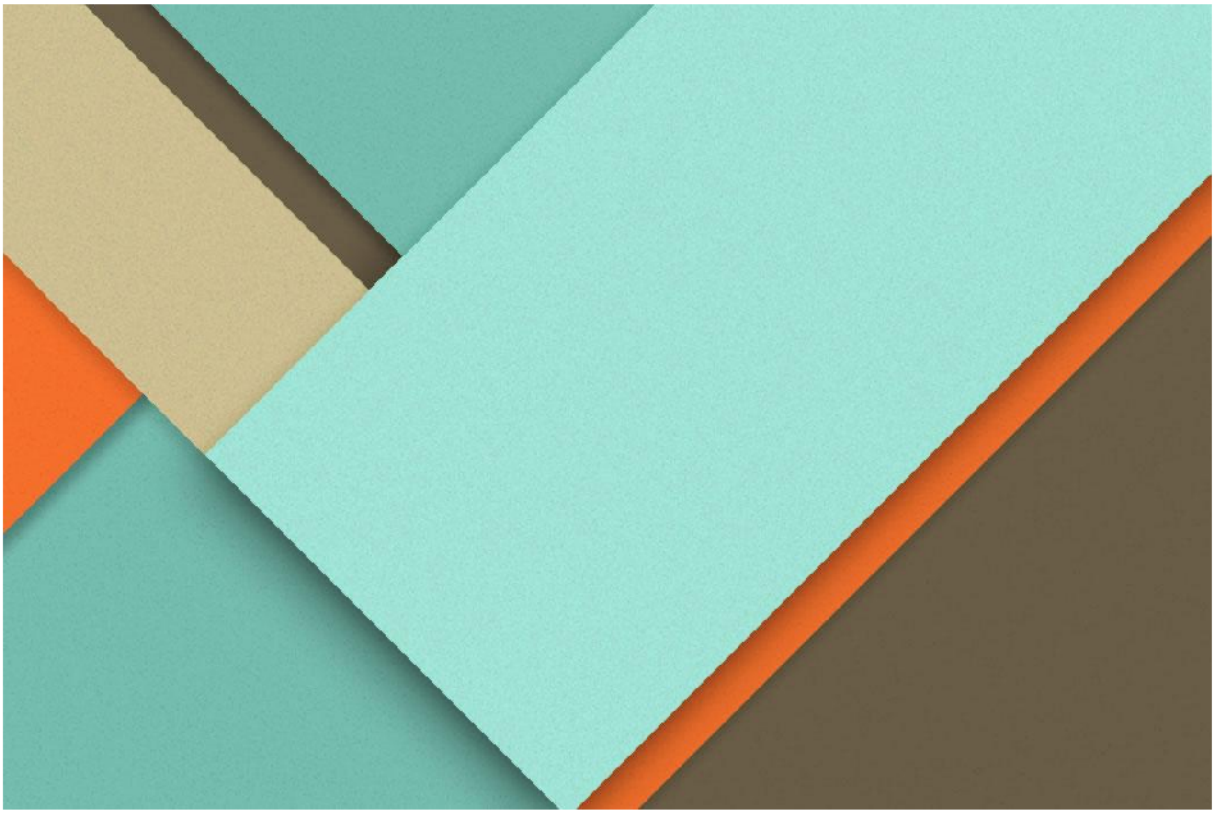

Apostila de Programação Orientada a Objetos

Prof.: Lucio Agostinho Rocha

Disciplina: Programação Orientada a Objetos

Curso: Bacharelado em Engenharia de Computação

Turma: POCO4A - 1o. Semestre de 2022

Importante: Essa apostila não substitui os livros da referência básica e complementar da disciplina. O objetivo desse material é que ele sirva como material didático-pedagógico auxiliar para a disciplina.

Visão geral

Este curso de Programação Orientada a Objetos (POO) realizado em conjunto com os estudantes de Bacharelado em Engenharia de Computação apresentará os conceitos básicos de programação orientada a objetos na linguagem de programação Java. Entende-se que os alunos deste curso já têm conhecimentos básicos de programação, razão pela qual os conteúdos de revisão são apresentados de maneira sucinta.

Objetivos

1. Material compartilhado com o conteúdo das aulas.
2. Material de estudo para a disciplina

Especificações

-Qualquer aluno do curso poderá contribuir com o conteúdo desse material.

Capítulo 1

I. Programação Orientada a Objetos

A Programação Orientada a Objetos (POO) é um tipo de programação em linguagem de alto nível. Propriedades e Comportamentos são armazenados em objetos:

- Propriedades: são atributos do objeto.
- Comportamentos: são as ações/operações que o objeto realiza.

Um POO é constituído de objetos. Cada objeto possui atributos (variáveis de instância) e comportamentos (métodos).

Classe: Na POO, a classe é um modelo que descreve os atributos e funcionalidades dos objetos. Os objetos são criados (instanciados) a partir da definição prévia definida na classe. As classes são utilizadas para construir objetos.

```
public class Estudante {
  private String nome;
  private int idade;
}
```

Objeto: o objeto é uma instância da classe. O objeto armazena o seu estado através de seus atributos. Objetos também são capazes de interagir uns com os outros através de seus métodos.

Objetos são instanciados com a palavra reservada "new", por exemplo:

```
Estudante  e  = new  Estudante( );
```

O operador ponto é utilizado para acessar os atributos do objeto.

```
e.nome = "Maria";
e.idade = 28;
```

O operador ponto também é utilizado para acessar os métodos do objeto.

Método: o método é muito similar às funções de linguagens procedimentais. O método define as ações/operações que o objeto realiza.

Exemplo 1: Métodos da classe Estudante:

```
public class Estudante {

  private String nome;
  private int idade;

  public String getNome(){
    return this.nome;
  }
  public void setNome(String nome){
    this.nome = nome;
  }
  public int getIdade(){
    return this.idade;
  }
  public void setIdade(int idade){
    this.idade = idade;
  }

  public static void main(String [ ] args){
    Estudante e = new Estudante();
    e.nome = "Maria";
    e.idade = 28;
    System.out.println(e.getNome() + " " + e.getIdade() );
  }//fim main
}//fim classe
```

Métodos acessores: por convenção, iniciam com a palavra "get" embora não seja obrigatório, e permitem retornar atributos encapsulados nos objetos.

Métodos mutadores: por convenção, iniciam com a palavra "set" embora não seja obrigatório, e permitem alterar os atributos encapsulados nos objetos.

Encapsulamento: é um princípio de projeto no qual cada componente do programa deve manter toda a informação relevante como uma unidade (cápsula). Na prática, a classe é o modelo que define o encapsulamento dos componentes do programa.

Além disso, os objetos são capazes de proteger seus atributos e comportamentos com modificadores de acesso. Isso significa que quaisquer classes externas que utilizem

objetos de outras classes só terão acesso aos atributos e métodos públicos da classe. A visibilidade de atributos e métodos dentro da própria classe é sempre pública.

Como exemplo, o código do Exemplo 1 mostra que os atributos e métodos private não são acessíveis e modificáveis por classes externas.

Exemplo 2: Encapsulamento do componente Sala.

```
public class Sala {

   public static void main(String [ ] args ){

      Estudante e = new Estudante();
      System.out.println( e.nome );   //Indisponível para acesso externo: private
      System.out.println( e.getNome() ); //OK
   }

}
```

Métodos static: Métodos static são métodos de classe e que estão disponíveis para uso sem necessidade de instanciar objetos. Como exemplo, a linguagem possui uma classe para cada tipo primitivo (por exemplo, int --> Integer, float --> Float, double --> Double, string --> String, e assim por diante). O método static dessas classes oferece diversas operações (métodos) úteis para tratamento de informações:

Exemplo 3: Métodos static

```
String idade_texto = "28";
int idade = Integer.parseInt( idade_texto );
```

Métodos static não têm acesso às variáveis de instância, apenas têm acesso às variáveis que também são declaradas "static".

II. História da Programação Orientada a Objetos

Leitura Recomendada:

CAPRETZ, L. F. “A Brief History of the Object-Oriented Approach”. In: ACM SIGSOFT Sofware Engineering Notes, vol.28, no.2, p.6, 2003.

III. Linguagem de Programação Java

Leitura Recomendada:

The Java Tutorials.
Disponível em: https://docs.oracle.com/javase/tutorial. Acesso em Maio de 2021.

Java API:
Disponível em: https://docs.oracle.com/en/java/javase/16/docs/api/index.html. Acesso em Maio de 2021.

JUnit:
Disponível em: http://junit.sourceforge.net/javadoc/org/junit/Assert.html. Acesso em Maio de 2021.

Diferentes nomes Java ao longo dos anos:

https://www.oracle.com/java/technologies/javase/naming-and-versions.html

https://www.whizlabs.com/blog/java-version-history/

https://www.java.com/pt-BR/download/help/techinfo_pt-br.html

Java SE inclui JDK e JRE (https://www.oracle.com/java/technologies/faqs-jsp.html)

IV. Exercícios em Sala

POO_Aula1Prog1: https://codeboard.io/projects/250930

Aula1Prog2: https://codeboard.io/projects/247036

Aual1Prog3: https://codeboard.io/projects/247044

Aula1Prog4: https://codeboard.io/projects/247056
(javax.swing): Pasta programas_exemplo

Aula1Prog5: https://codeboard.io/projects/247264

Aula1Prog6: https://codeboard.io/projects/247305

Github

https://github.com/poco4a/aula1

REFERÊNCIAS

DEITEL, Harvey M.; DEITEL, Paul J. **Java, como programar.** 6. ed. São Paulo, SP: Pearson Prentice Hall, 2005. xl, 1110 p. + 1 CD-ROM (4¾ pol.) ISBN 8576050196.

DEITEL, Harvey M.,; DEITEL, Paul J. **C++ como programar.** 5. ed. Porto Alegre, RS: Pearson Prentice Hall, 2006. xlii, 1163 p. + 1 CD-ROM ISBN 8576050560.

MEYERS, Scott. **C++ moderno e eficaz.** 1. ed. Rio de Janeiro, RJ: Alta Books, 2016. 366 p. ISBN 9788550800035 (broch.).

ZIVIANI, Nivio. **Projeto de algoritmos:** com implementações em Java e C++. São Paulo, SP: Cengage Learning, c2007..
621 p. ISBN 9788522105250.

SIERRA, Kathy; BATES, Bert. **Use a cabeça!** Java. 2. ed. Rio de Janeiro, RJ: Alta Books, 2007. 470 p. ISBN 9788576081739.

HORSTMANN, Cay S.; CORNELL, Gary. **Core Java.** 8. ed. São Paulo, SP: Pearson, c2010. xiii, v. ISBN 9788576053576.

VOTRE, Vilmar Pedro. **C++** explicado e aplicado. Rio de Janeiro, RJ: Alta Books, 2016. 662 p. ISBN 9788576089957 (broch.)

STROUSTRUP, Bjarne. **The C++ programming language.** 4th ed. Upper Saddle River, NJ: Addison-Wesley, c2013. xiv, 1347 ISBN 9780321563842.