PRADRÕES DE PROJETOS
com JAVA*

Msc. Angelo F. Dias Gonçalves

Hello!

Eu sou Angelo F. Dias

- Cientista da computação (Bacharel);
- Engenheiro da Computação pela Escola Politécnica de Pernambuco da Universidade de Pernambuco – UPE (Mestre);
  - Inteligência Artificial e Mineração de Dados
- Mais de 12 anos inserido no mercado de trabalho;
- Professor da UNIESP;
- Analista QA Sênior;
  - Programador C# com Framework Unity 3D/2D por opção.

EMENTA Simplificada

- Padrões de Projeto
  - Básico de JAVA
  - Introdução a POO
  - POO com JAVA
- GRASP
- GOF
- Projeto de Padrões de Projeto

EMENTA dos módulos

- Estudo de uma linguagem de programação orientada a objetos.
  - JAVA
- Paradigma orientado a objetos: classes, objetos, encapsulamento, polimorfismo, herança.
- Tipos de dados.
- Estrutura da linguagem.
- Coleções (vetor, lista, conjunto). Declarações.
- Comandos de atribuição, condicionais e de repetição.
- Arquitetura de sistemas Orientado a Objetos.

CONTEÚDOS GERAIS

- Introdução à Orientação a Objetos:
  - Explicação do paradigma e suas aplicações;
- Introdução à Orientação a Objetos:
  - Conceituar Classe, Método, Atributo e Objeto;
- Construtores, manipulação de objetos em java e interação entre classes;
- Herança e classes abstratas;
- Polimorfismo e encapsulamento;
- Estrutura de dados (com java).

Padrões de Projeto (Design Partner – D.P.) vai me ajudar a ser um profissional melhor (Clean Code)?

DESAFIOS

- Mercado de Trabalho

- Boas práticas de programação
  - Design Partner
  - Clean Code
  - SOLID*

Design Partterns e Clean Code

- As Design Patterns (ou **Padrões de Projeto**) são soluções que facilitam bastante a vida dos desenvolvedores, promovendo a **reusabilidade** do código. Por sua vez, a **reusabilidade** facilita a clareza e o entendimento do que está sendo produzido.
- Assim como é importante ter-se um padrão, é preciso ter um Código Limpo (o chamado **Clean Code**), a fim de **diminuir o retrabalho** e a ilegibilidade, adotando práticas que estabeleça um **padrão** para todos os integrantes das equipes ter como base para codificar.

DESAFIOS – "NOVOS"

- Mudar de linguagem 'facilmente';

- Encaixar OO em qualquer linguagem
  - Design Partner
  - Clean Code
  - SOLID*

- ...

JAVA.COM

- https://www.java.com/

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

Eclipse IDE for JAVA

- Versão Enterprise

Eclipse IDE for JAVA

- Versão EE

```
public class Main {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }

}
```

A IDE É SUA

Você pode escolher qualquer uma que lhe atenda!

https://metodoprogramar.com.br/melhores-ides-para-voce-programar-em-java/

1.
POO

Conceito e prática

O que é o Paradigma Orientado a Objetos (POO)?

- “A orientação a objetos, também conhecida como Programação Orientada a Objetos (POO) ou ainda em inglês Object-Oriented Programming (OOP) é um paradigma de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas de objetos.”

O que é POO?

- “Técnica de modelagem de software que procura construir sistemas complexos a partir de componentes.”

  Khoshafian, S. e Abnous, R

- Na programação orientada a objetos, implementa-se um conjunto de classes que definem os objetos presentes no sistema. Cada classe determina o comportamento (definido nos métodos) e estados possíveis (atributos) de seus objetos, assim como o relacionamento com outros objetos

O que é POO?

- C++, C#, Java, Object Pascal, Objective-C, Python, Ruby e Smalltalk são exemplos de linguagens de programação orientadas a objetos

- ActionScript, ColdFusion, Javascript, PHP (a partir da versão 4.0), Perl (a partir da versão 5) e VB.NET são exemplos de linguagens de programação com suporte a orientação a objetos

O que é POO?

- Possui 6 características marcantes
  1. **Abstração:** Aspectos essenciais
  2. **Estrutura de Objetos (Classes e Objetos):** Especificação
  3. **Herança:** Compartilhamento de estrutura
  4. **Encapsulamento:** Separação dos aspectos externos e internos de um objeto
  5. **Polimorfismo:** Combinação de dados e comportamentos
  6. **Sinergia:** Todas as características de maneira simultânea

1. O que é Abstração?

- Eliminação do irrelevante e amplificação do essencial
- Denota as características essenciais de um objeto que o distingue de outros objetos
- Oferece uma fronteira conceitual claramente definida na visão do observador
- Deve ser entendida e analisada independentemente do mecanismo que a implementa

1. O que é Abstração?

- Exemplos de Abstração
  - Animais (Vertebrados, Invertebrados)
  - Vertebrados (Anfíbios, Répteis, Mamíferos)
  - Mamíferos (Monotremata, Marsupiais, Placentários)
  - Placentários (Primatas, Carnívoras)
  - Primatas (Hominidae, Aotidae)
  - Hominidae (Homo, Pan)
  - Homo (Neanderthalensis, Sapiens)
  - ...

1. O que é Abstração?

- Vamos Abstrair:
  - Faça abstrações para:
    - Meios de Transporte (2 portas, 4 portas, sedan)
    - Computadores (Desktop – computador de mesa, Notebook-portátil, UltraBook)
    - Pessoas
    - Transações bancárias

2. O que é um Objeto?

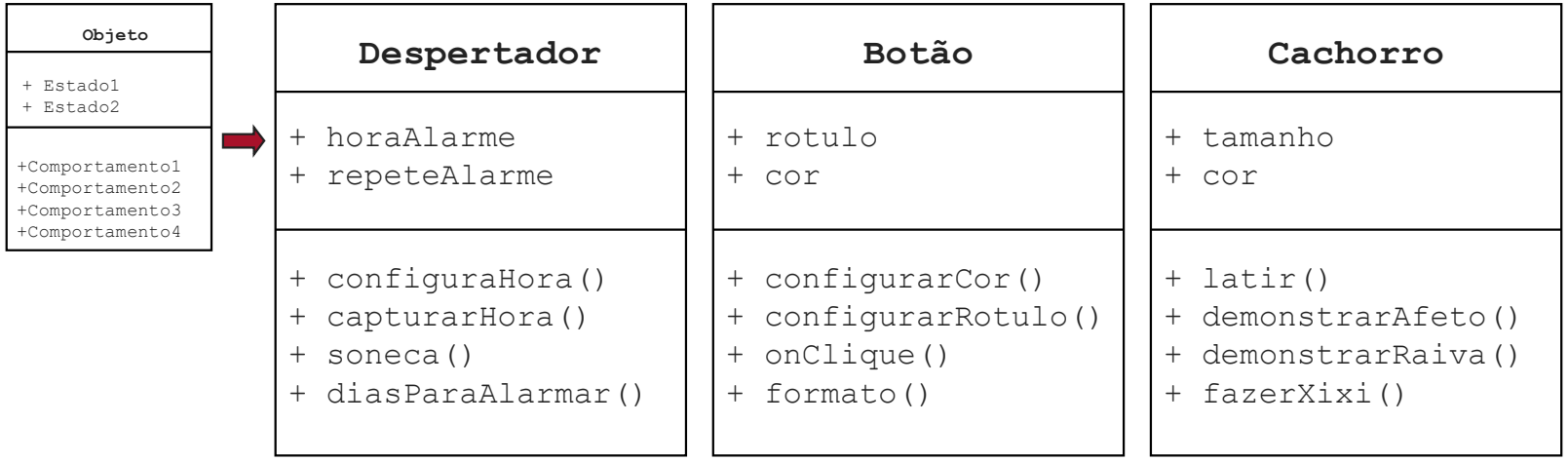
- Fundamental para a compreensão da tecnologia orientada a objetos
- Representa uma entidade que pode ser física, conceitual ou de software
- São usados frequentemente para modelar objetos do mundo real
  - Ex: cão, mesa, televisão, bicicleta etc.

2. O que é um Objeto?

- É uma abstração de algo que possui fronteira definida e significado para a aplicação
- Entidade com identificador, estados e comportamentos relacionados
- Instância de uma classe

2. O que é um Objeto?

- Exercício
  - Identificar os estados e os comportamentos de objetos d
  - o mundo real é uma ótima maneira de começar a pensar em termos de programação orientada a objeto. Identifique isso para os objetos abaixo:



Porque utilizar Objetos?

- Modularidade
  - O código-fonte de um objeto pode ser escrito e mantido independente do código-fonte de um outro objeto
- Ocultação de Informação
  - Ao interagir apenas com os métodos de um objeto, os detalhes internos de implementação permanecem escondidos do mundo exterior. Objetos como Caixa-Preta
- Reutilização de Código
  - Se um objeto já existe podemos usar esse objeto em outro programa

Porque utilizar Objetos?

- Facilidade para "*plug-and-play*"
  - Caso um objeto em particular esteja com problemas, simplesmente o removemos da aplicação e conectamos um outro objeto como seu substituto

- Facilidade para "debugging"
  - Encontrar e resolver problemas sem diretamente prejudicar o andamento do sistema como um todo. Se um parafuso quebra, basta substituí-lo, Não a máquina inteira

Vamos um gole por ver...

2. O que é uma Classe?

- Modelo ou protótipo a partir do qual os objetos são criados

- No mundo real, muitas vezes encontraremos muitos objetos individuais de uma mesma espécie.
  - Ex: Imagine a quantidade de marcas e modelos de bicicletas existentes no mundo. Cada tipo de bicicleta foi construída a partir do mesmo projeto e, portanto, contém os mesmos componentes. Em termos de orientação a objetos, dizemos que a bicicleta é uma instância da classe de objetos conhecidos como bicicletas

2. O que é uma Classe?

- Ao projeto de um objeto, isto é, a definição do objeto, damos o nome de Classe. Ao que podemos construir a partir desse projeto, damos o nome de objetos

- A palavra classe vem da taxonomia da biologia. Todos os seres vivos de uma mesma classe biológica têm uma série de atributos e comportamentos em comum, mas não são iguais, podem variar nos valores desses atributos e como realizam esses comportamentos

Como criar uma Classe?

- Definir as variáveis de instância, ou atributos, e seus respectivos tipos

```
class Conta {
   int numero;
   String nome;
   double saldo;
   double limite;
   // ..
}
```

- Definir os métodos da classe

```
class Conta {

   void saca(double quantidade) {
      double novoSaldo = this.saldo - quantidade;
      this.saldo = novoSaldo;
   }
   //...
}
```

Como criar um Objeto?

- Para criar (construir, instanciar) uma Conta, basta usar a palavra chave *new*

```
class Programa {
   public static void main(String[] args) {
      new Conta(); // Cria um objeto
   }
}
```

- O código acima cria um objeto do tipo Conta, mas como acessar esse objeto que foi criado? Criando uma **referência** a esse objeto.

```
class Programa {
   public static void main(String[] args) {
      Conta c = new Conta(); // Cria uma referência para um objeto
   }
}
```

```
class programaApp {
   public static void main(String[] args) {
      new Conta(); // Cria um objeto
```

E o “main”?

- É obrigatório!
  - toda aplicação precisa de um ponto de entrada.
- JRE (SO) que inicia a aplicação
- Na assinatura do método foi convencionado o uso de String[] args para receber argumentos de linha de comando ou terminal (CMD, Shell, Bash, etc.).
  - Também se optou por retornar nada (void) para o sistema operacional.
  - Você pode ignorar essa variável.

Tipos de Classe

- Abstrata
  - Desenvolvida para representar entidades e conceitos abstratos
  - Sempre é uma superclasse que não possui instâncias
  - Uma classe abstrata pode possuir métodos abstratos ou concretos
  - Um único método abstrato força a classe ser abstrata, necessariamente
  - Métodos abstratos definem apenas a assinatura do método e, portanto, não contém código

Tipos de Classe

- Concreta
  - Implementam todos os comportamentos das instâncias
  - Não possui métodos abstratos e, geralmente, quando utilizadas neste contexto, são classes derivadas de uma classe abstrata

Construtor e Destrutor

- Construtor
  - Método chamado quando uma nova instância do objeto é criada
  - Geralmente é responsável pela alocação de recursos necessários ao funcionamento do objeto além da definição inicial das variáveis de estado

```
public class Exemplo {

   int data;

   // Construtor
   public Exemplo() {
      data = 1;
   }
}
```

Construtor e Destrutor

- Destrutor
  - Membro de uma classe, um método, invocado quando a classe deve ser destruída
  - Utilizado, normalmente, para liberar a memória alocada dinamicamente pela classe e para eliminar as possíveis referências à classe, quando ela não mais existir
  - Não recebe parâmetros de entrada ou de saída
  - Java não possui método Destrutor, entretanto, possui o recurso Garbage Collection (coletor de lixo), que retiram da memória os objetos não referenciados

Python – POO!

- Mercado de trabalho!
  - ;)

Thanks!

DÚVIDAS?

@oanjoeducador

prof2132@iesp.edu.br

+55 81 98808 1008

angelofdiasg.tech (em dev)

https://www.linkedin.com/in/angelofdiasg/