CURSO TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS
Disciplina: Desenvolvimento de Sistemas I
Docente: Anildo Nascimento Matos
Aluno (a): Hercules Oliveira das Mercês

Conceitos Programação Orientada objetos

Salvador/ BA
2020

HERANÇA

O conceito de herança pode ser entendido como o recurso pelo qual uma determinada classe, denominada sub-classe ou classe filha, pode receber atributos e métodos presentes em outra classe, neste caso chamada de super-classe ou classe pai.
Uma das características fundamentais da orientação a objetos é a capacidade de reaproveitamento de código, a qual pode ser implementada através da adoção do conceito de herança.

PETRONI, Benedito; OLIVEIRA, Cláudio. **Fundamentos da Linguagem de Programação Java, p. 24.**

Segundo (PETRONI, p. 24; OLIVEIRA, p. 24), herança é o recurso em que uma sub-classe, recebe (herda) atributos e métodos de uma classe superior (Classe PAI), e desse modo facilita reaproveitamento de código.

CLASSE ABSTRATA

Classes abstratas são classes que servem como modelo para suas subclasses. Classes abstratas não podem ser instanciadas, apenas herdadas.

Os métodos declarados na classe abstrata devem ser implementados nas subclasses (classes concretas). Caso contrário, as subclasses também acabam se tornando classes abstratas.

SERMENHO, Rafael. Conceitos Básicos de Orientação a Objetos — Parte 2. **Medium**, 2016. Disponível em: <https://medium.com/@RafaelSermenho/conceitos-b%C3%A1sicos-de-orienta%C3%A7%C3%A3o-a-objetos-parte-2-5accfe670a6e>. Acesso em: 27 nov. de 2020.

Segundo (SERMENHO, 2016), as classes abstratas são modelos para subclasses, onde não pode instanciar, somente herdar, e implementar nas subclasses.

POLIMORFISMO

A palavra polimorfismo vem do grego poli morfos e significa muitas formas. Na orientação a objetos, isso representa uma característica que permite que classes diferentes sejam tratadas de uma mesma forma.

O polimorfismo permite escrever programas que processam objetos que compartilham a mesma superclasse (direta ou indiretamente) como se todos fossem objetos da superclasse; isso pode simplificar a programação.

> DEITEL, Harvey; DEITEL, Paul. **Java: como programar**. 8. ed. São Paulo: Pearson/ Prentice-Hall, 2010, p. 305.

> Segundo (DEITEL, 2010, p. 305; DEITEL, 2010, p. 305), polimorfismo significa muitas formas, onde um mesmo método pode ser executado de forma diferente de acordo a classe objeto acionando parâmetros.

SOBRECARGA DE MÉTODOS

Métodos de mesmo nome podem ser declarados na mesma classe, contanto que tenham diferentes conjuntos de parâmetros (determinado pelo número, tipos e ordem dos parâmetros). Isso é chamado sobrecarga de. Para que os métodos de mesmo nome possam ser distinguidos, eles devem possuir assinaturas diferentes. A assinatura (signature) de um método é composta pelo nome do método e por uma lista que indica os tipos de todos os seus argumentos. Assim, métodos com mesmo nome são considerados diferentes se recebem um diferente número de argumentos ou tipos diferentes de argumentos e têm, portanto, uma assinatura diferente. Quando um método sobrecarregado é chamado, o compilador Java seleciona o método adequado examinando o número, os tipos e a ordem dos argumentos na chamada.

DEITEL, Harvey; DEITEL, Paul. **Java: como programar**. 8. ed. São Paulo: Pearson/ Prentice-Hall, 2010, p. 174.

Segundo (DEITEL, 2010, p. 174; DEITEL, 2010, p. 174), métodos com mesmo nome podem ser declarados na mesma classe, para ser diferenciados, devem possuir argumentos, e assinaturas diferentes. E quando o método sobrecarregado, é invocado o compilador seleciona mediante analise do tipo, e ordem.

PROPRIEDADES

Uma propriedade é um membro que oferece um mecanismo flexível para ler, gravar ou calcular o valor de um campo particular. As propriedades podem ser usadas como se fossem membros de dados públicos, mas na verdade elas são métodos realmente especiais chamados acessadores. Isso permite que os dados sejam acessados facilmente e ainda ajuda a promover a segurança e a flexibilidade dos métodos

As propriedades podem ser de leitura/gravação (elas têm um acessador get e set), somente leitura (elas têm um acessador get, mas nenhum set) ou somente gravação (elas têm um acessador set, mas nenhum get). As propriedades somente gravação são raras e são mais comumente usadas para restringir o acesso a dados confidenciais.

Propriedades (Guia de Programação em C#). **Docs. Microsoft**, 2017. Disponível em: < https://docs.microsoft.com/pt-br/dotnet/csharp/programming-guide/classes-and-structs/properties>. Acesso em: 27 de nov. de 2020.

De acordo com a (DOCS.MICROSOFT, 2017), propriedade, é um mecanismo flexível para ler, gravar, calcular. Utilizando GET para leitura, e SET para gravação, e assim facilita acesso aos dados e passa segurança.

INTERFACE

Podemos definir como interface o contrato entre a classe e o mundo exterior. Quando uma classe implementa uma interface, se compromete a fornecer o comportamento publicado por esta interface.

As classes ajudam a definir um objeto e seu comportamento e as interfaces que auxiliam na definição dessas classes. As interfaces são formadas pela declaração de um ou mais métodos, os quais obrigatoriamente não possuem corpo.

> CAMARGO, Wellington. Interfaces: Programação Orientada a Objetos. **Devmedia**, 2010. Disponível em: <https://www.devmedia.com.br/interfaces-programacao-orientada-a-objetos/18695#:~:text=Podemos%20definir%20como%20interface%20o,auxiliam%20na%20defini%C3%A7%C3%A3o%20dessas%20classes>. Acesso em: 27 de nov. de 2020.
>
> De acordo com (CAMARGO, 2010), interface é como um contrato, que a classe que utiliza deve cumprir com o comportamento publicado na interface.

ENCAPSULAMENTO

O *encapsulamento* é uma das principais técnicas que define a programação orientada a objetos. Se trata de um dos elementos que adicionam segurança à aplicação em uma programação orientada a objetos pelo fato de esconder as propriedades, criando uma espécie de caixa preta.

A maior parte das linguagens orientadas a objetos implementam o encapsulamento baseado em propriedades privadas, ligadas a métodos especiais chamados *getters* e *setters*, que irão retornar e setar o valor da propriedade, respectivamente. Essa atitude evita o acesso direto a propriedade do objeto, adicionando uma outra camada de segurança à aplicação.

> GASPAROTTO, Henrique. Os 4 pilares da Programação Orientada a Objetos. **Devmedia**, 2014. Disponível em: <https://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>. Acesso em: 27 de nov. de 2020.
>
> Segundo (GASPAROTTO, 2014), encapsulamento é um elemento que adiciona segurança, escondendo as propriedades, e evitando acesso direto, mediante get e set, irão retornar o valor da propriedade.

MÉTODOS COM E SEM RETORNO

Métodos sem retorno em Java

Esse tipo de método executa apenas o código que tem dentro dele, não retornando nenhum resultado, sendo identificados com a palavra-chave void.

Métodos com retorno em Java

Esses métodos que não possuem a palavra-chave void incorporada na declaração, mas sim um tipo de dados, apresentam em seu corpo a palavra reservada return, que informa que o método terá que retornar o mesmo tipo de dados com o qual foi declarado.

> PALMEIRA, Thiago. Trabalhando com métodos em Java. **Devmedia**, 2012. Disponível em: <https://www.devmedia.com.br/trabalhando-com-metodos-em-java/25917>. Acesso em: 27 de nov. de 2020.
>
> Segundo (PALMEIRA, 2012), o método é sem retorno quando executa apenas o código, e não retornam valores, métodos sem retorno são identificados pela palavra reservada VOID, que significa vazio. E o método com retorno, como o próprio nome diz, retorna valores, não contém a palavra VOID, e sim a palavra reservada RETURN, que informa ao método um retorno com o mesmo tipo de dado declarado anteriormente.