

Concepts of object-oriented programming

Leandro Gonçalves - 07020114, Álvaro Kissula - 07020834

Acadêmicos de Sistemas de Informação
Universidade Paranaense (UNIPAR) – Cascavel, PR – Brazil
leandro.gwn@gmail.com, alvarokissula@hotmail.com

Abstract. Here is the post shows the main concepts of some items of object-oriented programming, namely constructors, Encapsulation, Polymorphism, Abstract Classes, Interfaces, Enumeration.

Resumo. Aqui será posto a mostra os conceitos principais de alguns itens da programação orientada a objetos, sendo eles Construtores, Encapsulamento, Polimorfismo, Classes abstratas, Interfaces, Enumeration.

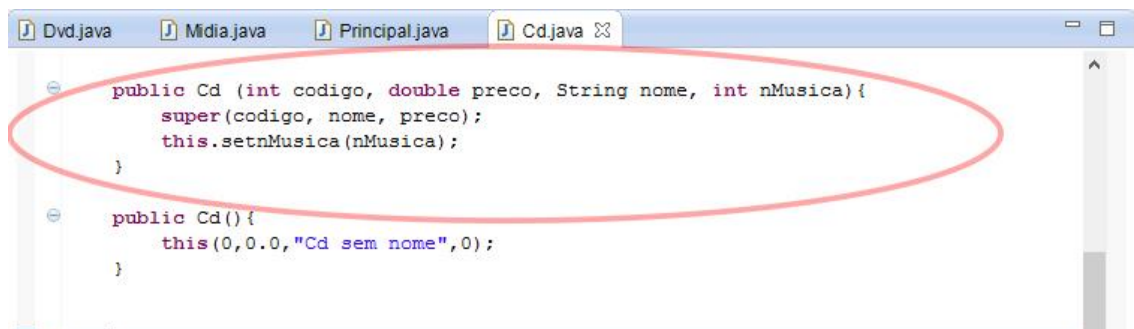
1. Constructors

A constructor method is the method which has the instructions that will be executed whenever an object is instantiated from this class, the constructor can be done by passing some value but can also be an empty constructor. This method is usually responsible for the allocation of resources necessary for the operation of the object beyond the initial definition of the state variables (attributes).

1. Construtores

Um método construtor é o método onde tem as instruções que serão executadas sempre que for instanciado um objeto desta classe, no construtor pode ser feito a passagem de algum valor como também pode ser um construtor vazio. Tal método geralmente é responsável pela alocação de recursos necessários ao funcionamento do objeto além da definição inicial das variáveis de estado (atributos).

Figure 1. Constructor is created and receives values below
Figura 1. Construtor é criado e logo abaixo recebe valores



```
public Cd (int codigo, double preco, String nome, int nMusica){  
    super(codigo, nome, preco);  
    this.setnMusica (nMusica);  
}  
  
public Cd(){  
    this(0,0.0,"Cd sem nome",0);  
}
```

2. Encapsulation

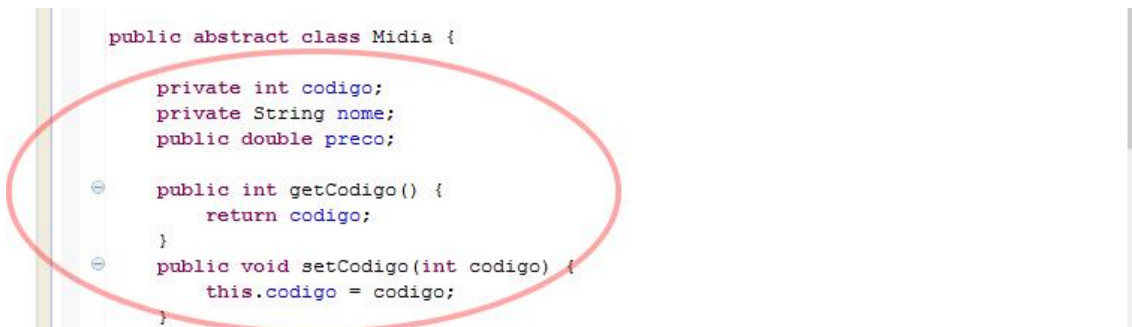
The encapsulation serves to control access to attributes and methods of a class. It is an efficient way to protect data handled within the class, and determine where this class can be manipulated. We use the most restrictive access level, private, that makes sense for a particular member. Always use private, unless we have good reason to leave you with another level of access.

2. Encapsulamento

O Encapsulamento serve para controlar o acesso aos atributos e métodos de uma classe. É uma forma eficiente de proteger os dados manipulados dentro da classe, além de determinar onde esta classe poderá ser manipulada. Usamos o nível de acesso mais restritivo, private, que faça sentido para um membro particular. Sempre usamos private, a menos que tenhamos um bom motivo para deixá-lo com outro nível de acesso.

Figure 2. Private attributes are given in their access modifiers and is created just below the methods Getters and Setters

Figura 2. Atributos recebem Private em seus modificadores de acesso e logo abaixo é criado os metodos Getters e Setters



```
public abstract class Midia {  
  
    private int codigo;  
    private String nome;  
    public double preco;  
  
    public int getCodigo() {  
        return codigo;  
    }  
  
    public void setCodigo(int codigo) {  
        this.codigo = codigo;  
    }  
}
```

The image shows a code editor with a Java class named 'Midia'. The class is an abstract class. It has three private attributes: 'codigo' (int), 'nome' (String), and 'preco' (double). Below the attributes, there are two public methods: 'getCodigo()' which returns the 'codigo' attribute, and 'setCodigo(int codigo)' which sets the 'codigo' attribute. A red oval highlights the private attributes and the public methods.

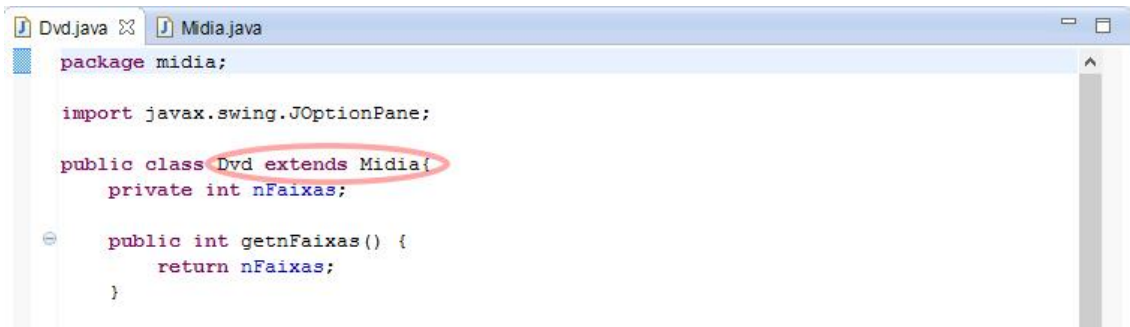
3. Polymorphism

Polymorphism means "many forms", the term is defined in object-oriented languages, that allows the developer to use the same element in different ways. Specifically in Java, polymorphism is in fact can completely modify the code of a method inherited from a different class, or overriding the parent class method.

3. Polimorfismo

Polimorfismo significa "muitas formas", é o termo definido em linguagens orientadas a objeto, que permite ao desenvolvedor usar o mesmo elemento de formas diferentes. Especificamente em Java, polimorfismo se encontra no fato de podemos modificar totalmente o código de um método herdado de uma classe diferente, ou seja, sobrescrevemos o método da classe pai.

Figure 2. Class Dvd taking shape and extending the abstract class Media
Figura 2. Classe Dvd extendendo e tomando forma da classe abstrata Midia



4. Classes abstract

Classes are abstract classes that are not instantiated, ie are classes that it will not create any objects directly.

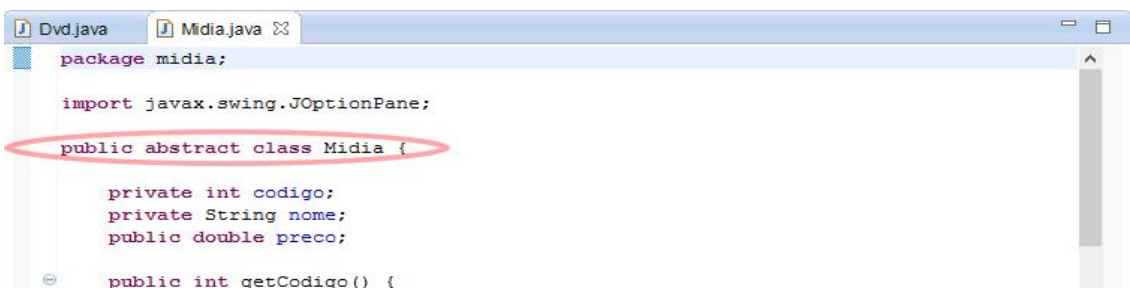
There is a feature in Java that, in an abstract class, we can write that particular method is always written by the child classes. This is an abstract method. It indicates that all subclasses (concrete that is, they are not abstract) must rewrite this method compile or not. It's like if you inherit the responsibility of having that method.

4. Classes abstratas

Classes abstratas são classes que não serão instanciadas, ou seja, são classes em que não criaremos nenhum objeto dela, diretamente.

Existe um recurso em Java que, em uma classe abstrata, podemos escrever que determinado método será sempre escrito pelas classes filhas. Isto é, um método abstrato. Ele indica que todas as classes filhas (concretas, isto é, que não forem abstratas) devem reescrever esse método ou não compilarão. É como se você herdasse a responsabilidade de ter aquele método.

Figure 2. Abstract Class Media
Figura 2. Classe Midia abstrata



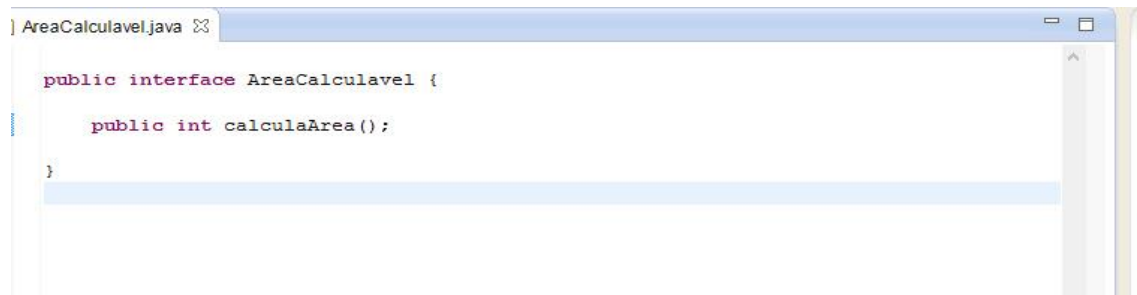
4. Interface

Interface is a feature of object-orientation in Java used to define actions that have to be executed, but that each class can perform differently. Interfaces contain constant values or method signatures which must be implemented within a class.

4. Interface

Interface é um recurso da orientação a objeto utilizado em Java que define ações que devem ser obrigatoriamente executadas, mas que cada classe pode executar de forma diferente. Interfaces contém valores constantes ou assinaturas de métodos que devem ser implementados dentro de uma classe.

Figure 2. Abstract Class Media
Figura 2. Classe Midia abstrata



```
AreaCalculavel.java
public interface AreaCalculavel {
    public int calculaArea();
}
```

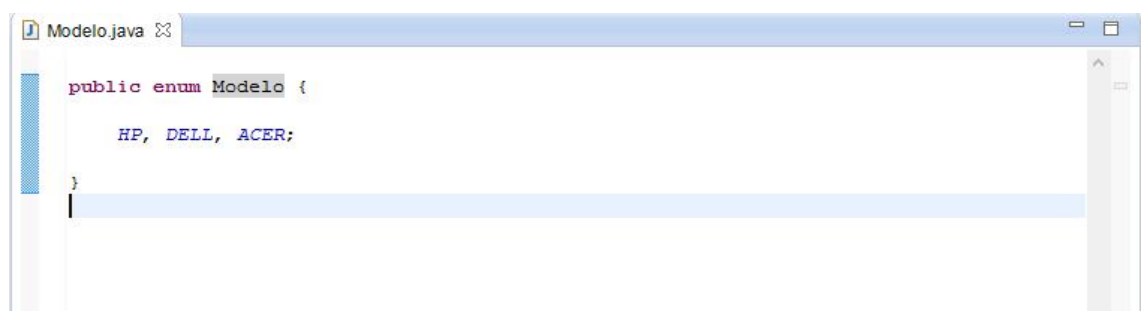
5. Enumeration

Enumeration is nothing more than a resource, very useful in object-oriented programming, done to treat, store and easily use its constants, assigning default values and made the code more readable.

5. Enumeration

Enumeration nada mais é que um recurso, muito útil na programação orientada a objetos, feito para tratar, guardar e usar com facilidade suas constantes, atribuindo valores predefinidos e tornado o código mais legível.

Figure 2. Enum class presetting values for Model
Figura 2. Classe enum predefinindo valores do para Modelo



```
Modelo.java
public enum Modelo {
    HP, DELL, ACER;
}
```

References

Support materials available during class.

Referências

Materiais de apoio disponibilizados durante as aulas.