PROGRAMAÇÃO I - CCT0827

Semana Aula: 6

Unidade 2 - Conceitos de orientação a objetos

Tema

Agregação e Particionamento

Palavras-chave

Particionamento, Agregação

Objetivos

O aluno deverá ser capaz de:

- Compreender o conceito de Agregação e Particionamento.
- Aplicar os conceitos de Agregação e Particionamento em Java.

Estrutura de Conteúdo

Particionamento

É a decomposição de classes extensas em classes menores, que podem ser melhor reaproveitadas em outras classes, além de permitir melhor controle e manutenção.

O particionamento de classes nos permite criar objetos menores e mais simples, que poderão ser reunidos em conjunto, capazes de criar novas classes, maiores e mais complexas.

Exemplo:

Um computador do tipo Desktop, que é um objeto bem complexo e possui diferentes partes. Muitas destas partes são usadas por outros objetos também, tal como Notebooks e Servidores. Um HD (HardDisk), por exemplo pode ser usado por cada um deles, assim como a placa de vídeo, a placa-mãe, o vídeo, a memória, além de outros dispositivos. Se formos criar uma classe para representar um Desktop, termos uma classe com muitos atributos, o que a faria ser grande e complexa, com muitas linhas de código e de difícil manutenção, uma vez que qualquer mudança necessária teríamos que trabalhar em uma classe altamente complexa. Outro ponto importante seria a criação das classes Notebook e Servidor, que seriam igualmente complexas, sem contar que uma alteração em um único componente que fosse, teríamos que realizar a alteração em todas as três classes (Desktop, Notebook e Servidor).

Incialmente nossa classe Desktop ficaria com os seguintes atributos:

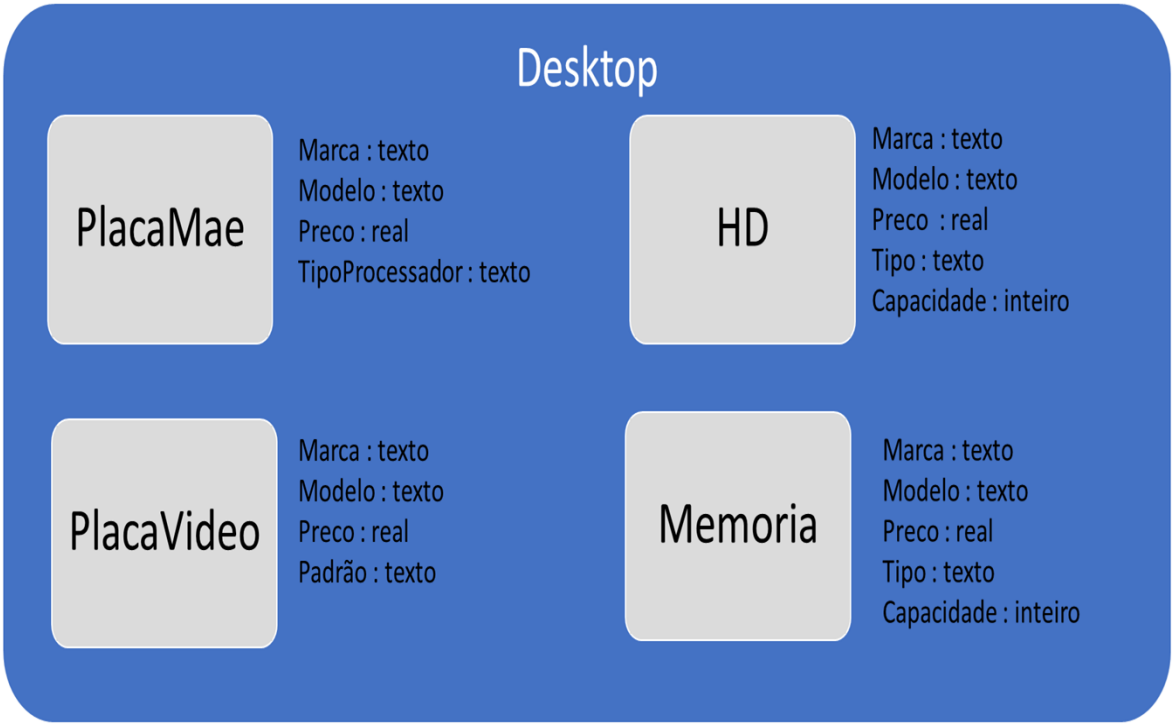
```
public class Desktop {
```

```
public String marcaPlacaMae, modeloPlacaMae;

public double precoPlacaMae;

public String tipoProcessador, marcaHD, modeloHD;

public double precoHD;

public String tipoHD;

public int capacidadeHD;

public String marcaPlacaVideo, modeloPlacaVideo;

public double precoPlacaVideo;

public String padrao, marcaMemoria, modeloMemoria;

public double precoMemoria;

public String tipoMemoria;

public int capacidadeMemoria;

}
```

Trabalhando com o particionamento, podemos então dividir a classe Desktop, grande e complexa, em classes menores e mais simples.

- Placa-mãe (PlacaMae)
- Disco Rígido (HD)
- Placa de vídeo (PlacaVideo)
- Memória (Memoria)

A classe Desktop decomposta ficará da seguinte forma, com seus atributos:



Assim, vamos criar as classes separadamente, dividindo a classe Desktop conforme proposto:

```
public class PlacaMae {
public String marca, modelo, tipoProcessador;
public double preco;
}
```

```
public class HD {
public String marca, modelo, tipo;
public double preco;
public int capacidade;
}
```

```
public class PlacaVideo {
public String marca, modelo;
```

```
public double preco;

public int capacidade;

}
```

```
public class Memoria {

public String marca, modelo, tipo;

public double preco;

public int capacidade;

}
```

Agregação

A reunião de uma ou mais classes para formar novas classes é chamado de agregação. Uma nova classe pode ser formada por um conjunto de diferentes objetos.

Seguindo o exemplo anterior, poderíamos reaproveitar as classes PlacaMae, HD, PlacaVideo, e Memoria para criar novas classe como Desktop, Notebook e Servidor, como nos exemplos a seguir:

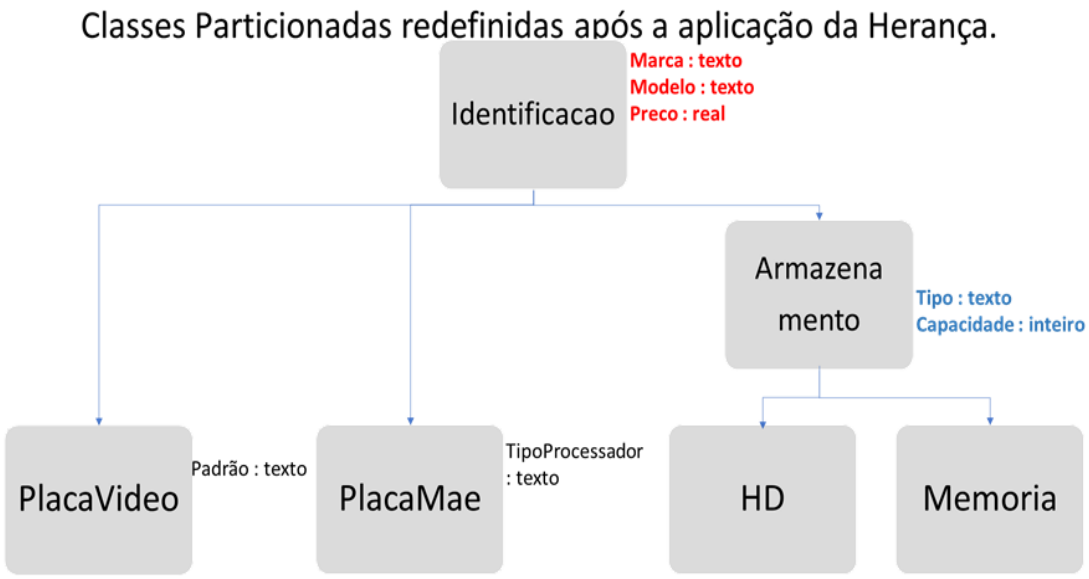
```
public class Desktop {

public String tipoCooler;

public PlacaMae pm = new PlacaMae();

public PlacaVideo pv = new PlacaVideo();

public HD hd = new HD();

public Memoria me = new Memoria();

}
```

```
public class Notebook {

public double peso;

public PlacaMae pm = new PlacaMae();
```

```
public PlacaVideo pv = new PlacaVideo();

public HD hd = new HD();

public Memoria me = new Memoria();

}
```

```
public class Servidor {

public int numeroPlacasRede;

public PlacaMae pm = new PlacaMae();

public PlacaVideo pv = new PlacaVideo();

public HD hd = new HD();

public Memoria me = new Memoria();

}
```

Visando melhorar as classes podemos aplicar o conceito de herança.



Estratégias de Aprendizagem

Implementar programas em Java, explorando os conceitos de herança e sobrescrita.

Indicação de Leitura Específica

Aplicação: articulação teoria e prática

Considerações Adicionais