PROGRAMAÇÃO I - CCT0827

Semana Aula: 5

Unidade 2 - Conceitos de orientação a objetos

Tema

Herança

Palavras-chave

Herança, Superclasse, Subclasse, Tipos de Herança

Objetivos

O aluno deverá ser capaz de:

·Compreender o conceito de herança.

· Aplicar o conceito de herança na construção e análise de programas em Java.

Estrutura de Conteúdo

Herança

Mecanismo pelo qual classes compartilham atributos e comportamentos através de um relacionamento hierárquico.

É uma forma de dizer que uma classe "é um tipo de" outra classe.

Exemplos:

Um professor é um tipo de funcionário.

Um pentágono é um tipo de polígono.

A herança é um conceito muito importante que possibilita que você ao identificar duas ou mais classes que possuam semelhanças, estas podem ser definidas através de uma hierarquia, onde os membros comuns as duas ou mais classes passam para uma classe nova classe conhecida como Superclasse ou classe ?mãe?. Já as classes originais permanecerão apenas com os membros não comuns e estas classes são denominadas Subclasses ou classes ?filhas?.

Na herança, temos sempre uma classe definida de forma genérica que, posteriormente, é

refinada em classes mais específicas.

Exemplo :

Um carro é um tipo de veículo.

Podemos ter várias classes específicas para uma mesma classe genérica:

Exemplo :

Um carro é um tipo de veículo

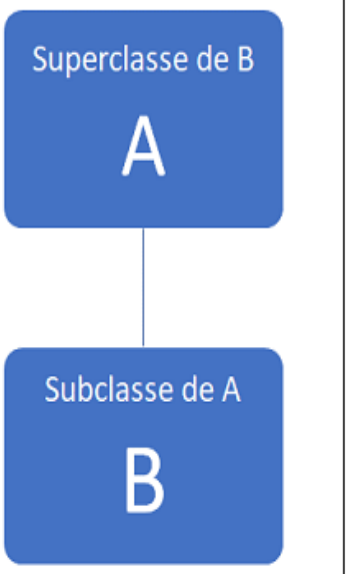
Um caminhão é um tipo de veículo

Cada classe específica possui atributos e comportamentos somente seus, mas também herda atributos e comportamentos da classe genérica.

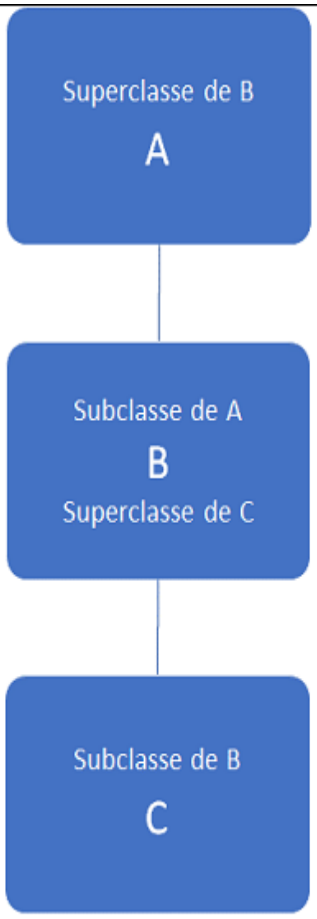
Tipos de Herança

1. Herança Simples

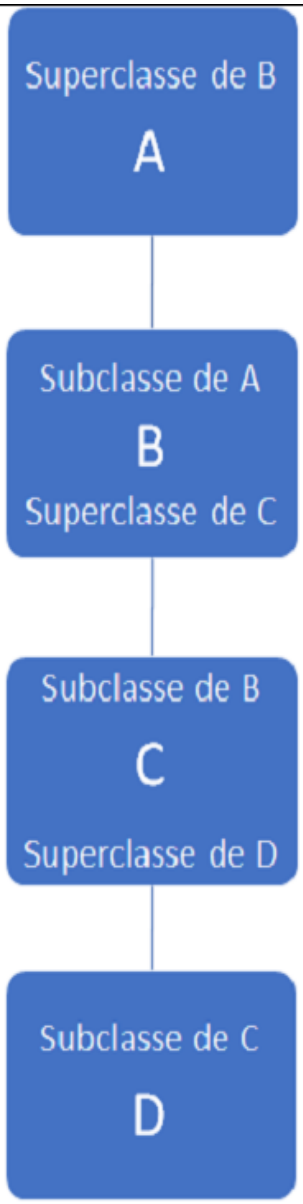
Em Java temos apenas a implementação da herança simples. A herança simples se caracteriza por cada classe herdar sempre de apenas uma outra classe por vez. Devemos observar que mesmo que tenhamos uma sequência de classes herdando, onde uma herda da outra, ainda assim, temos a herança simples



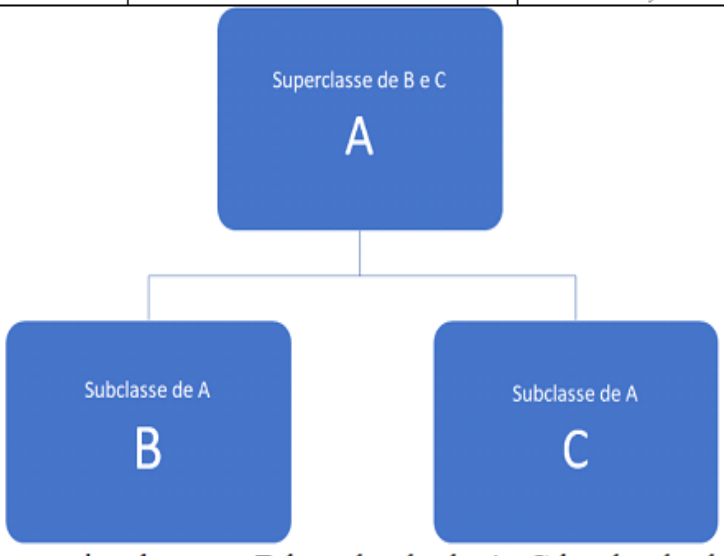
Herança simples com B heradando de A.

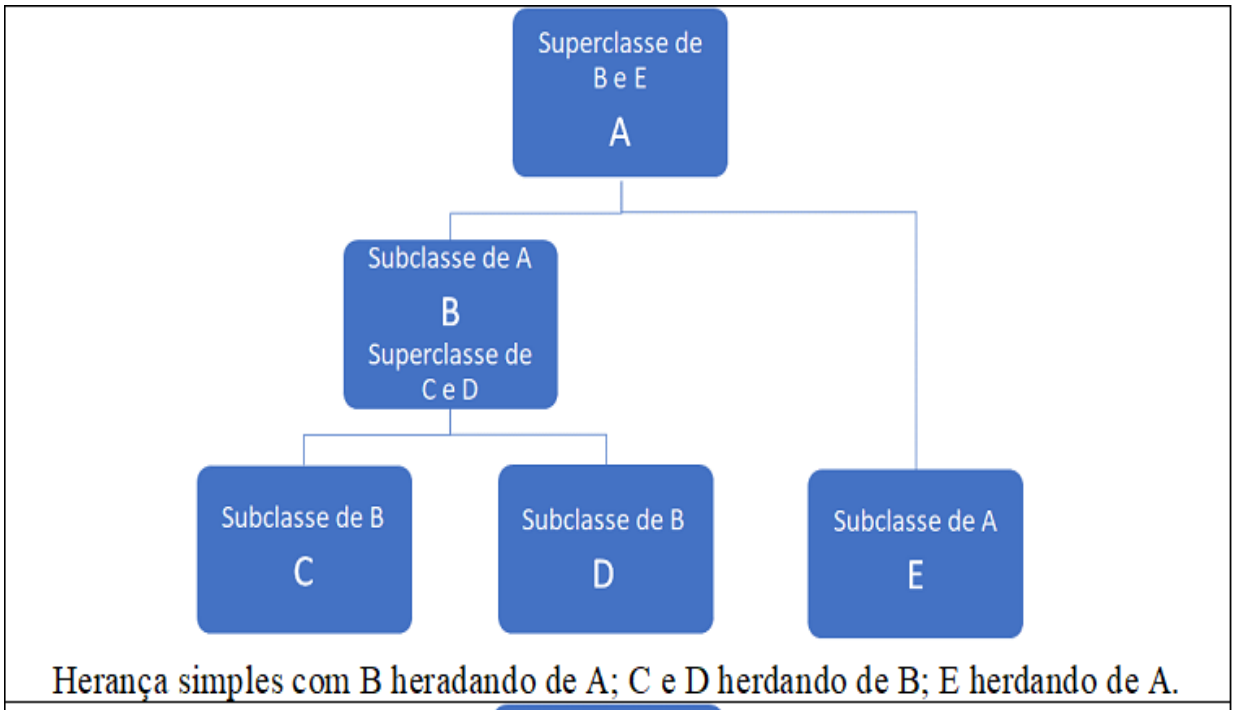


Herança simples com B heradando de A; C herdando de B.

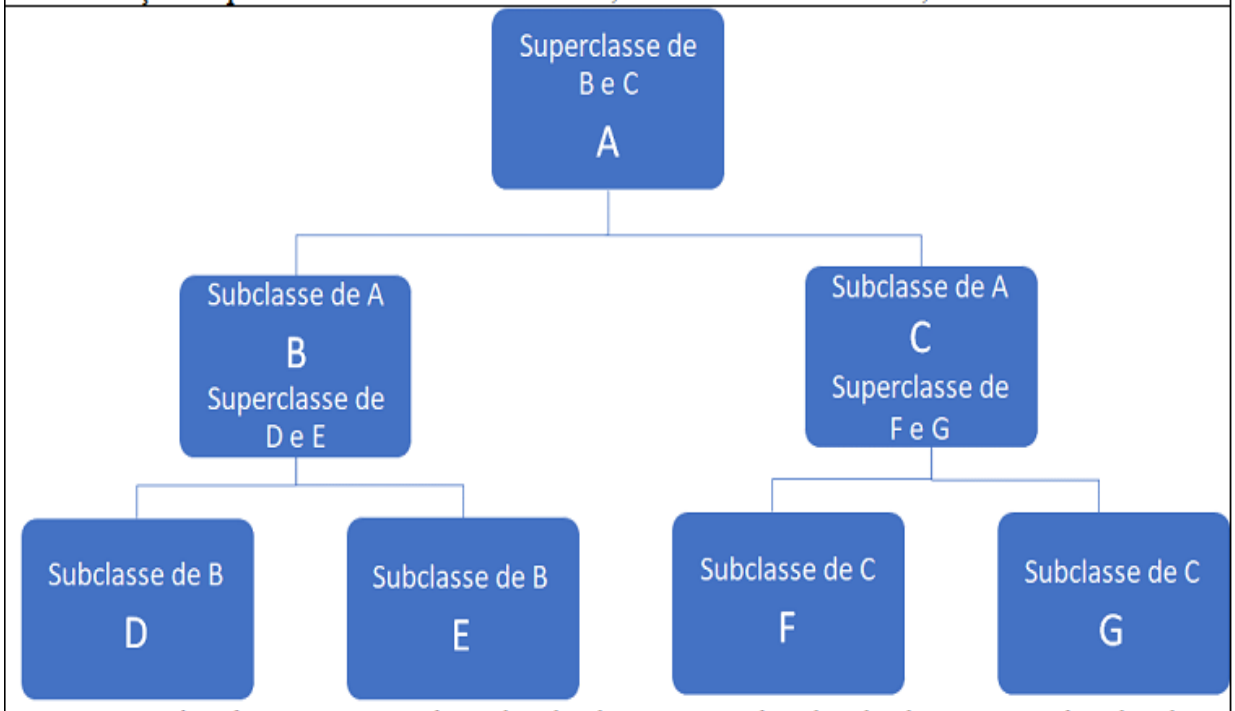


Herança simples com B heradando de A; C herdando de B; D herdando de C.





Herança simples com B heradando de A; C e D herdando de B; E herdando de A.



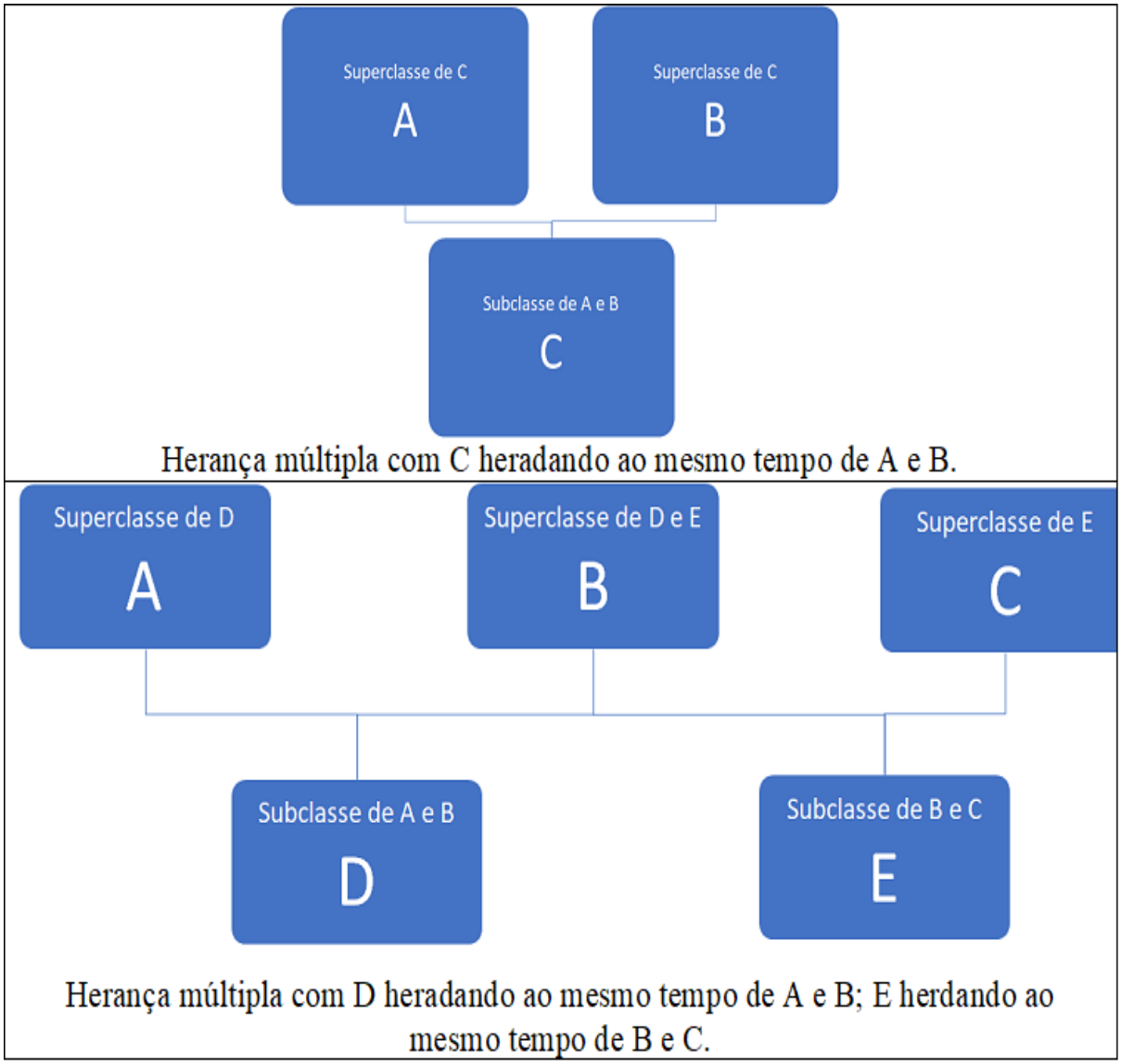
Herança simples com B e C heradando de A; D e E herdando de B; F e G herdando de C.

1.2. Herança Múltipla

A herança múltipla se caracteriza quando uma mesma classe herda de duas ou mais classes ao mesmo tempo.

Java não permite a implementação da herança múltipla, mesmo este sendo um conceito da programação orientada a objetos.

Algumas linguagens de programação não implementam este conceito. A linguagem C++ permite a implementação de herança múltipla, mas Java e C# por exemplo, não permitem esta implementação.



Herança em Java

Para definir a herança entre duas classes devemos usar a palavra reservada extends na

definição da subclasse.

Exemplo:

```
public class Carro extends Veiculo {

private int qtdPassageiros;

private int capacidadeBagagem;

//Note que Carro herda atributos e métodos de Veiculo.

}
```

Classe mãe ou superclasse : Veiculo

Classe filha ou subclasse : Carro

Sobrescrita de métodos

Métodos de uma SuperClasse podem ser sobrescritos em suas subclasses, isso implica que um método descrito na Superclasse poderá ser substituído na Subclasse. Para isso é importante observar que estes métodos devem possuir as mesmas assinaturas, caso contrário será usado o conceito de Sobrecarga e não de Sobrescrita.

Sobrecarga X Sobrescrita

- Sobrecarga ou overload : métodos da mesma classe; métodos com assinaturas diferentes e a chamada do método ocorre em tempo de compilação.
- Sobrescrita ou override : Métodos em classes diferentes em uma hierarquia de classes; métodos com a mesma assinatura; ocorre em tempo de execução ou ligação tardia.

Estratégias de Aprendizagem

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

Exemplo de Aplicação de Herança.

Classe **Pessoa** com o uso do conceito de Herança (SuperClasse):

```
import java.util.Scanner;

public class Pessoa {

String identidade, nome;

public String getIdentidade() {

return identidade;
```

```
}
public void setIdentidade( String id ) {
if(!id.isEmpty()) {
identidade = id;
}
}
public String getNome() {
return nome;
}
public void setNome( String no ) {
if(!no.isEmpty()) {
nome = no;
}
}
public Pessoa() { }
public Pessoa( String id ) {
setIdentidade ( id );
}
public Pessoa( String id, String no ) {
setIdentidade ( id );
setNome ( no );
} // Só podemos criar 3 construtores para a classe Pessoa
public void cadastrar( String id, String no ) {
setIdentidade ( id );
setNome ( no );
```

```
}

public void entradaDados( ) {

Scanner entrada = new Scanner( System.in );

System.out.println( "Identidade :" );

setIdentidade( entrada.nextLine() );

System.out.println( "Nome :" );

setNome( entrada.nextLine() );

entrada.close();

}

public void imprimir( ) {

System.out.println( "Identidade :" + getIdentidade() );

System.out.println( "Nome :" + getNome() );

}

}
```

Classe **PessoaEmpresa** com o uso do conceito de Herança (SubClasse de Pessoa e SuperClasse de Gerente e Funcionário):

```
import java.util.Scanner;

public class PessoaEmpresa extends Pessoa{

String matricula;

double salario;

public String getMatricula() {

return matricula;

}

public void setMatricula( String ma ) {

if(!ma.isEmpty()) {

matricula = ma;
```

```
}
}
public double getSalario() {
return salario;
}
public void setSalario( double sa ) {
if ( sa >=0) {
salario = sa;
}
}
public PessoaEmpresa() { }
public PessoaEmpresa( String id ) {
super ( id );
}
public PessoaEmpresa( double sa ) {
setSalario ( sa );
}
public PessoaEmpresa( String id, double sa ) {
super ( id );
setSalario ( sa );
}
public PessoaEmpresa( double sa, String id ) {
super ( id );
setSalario ( sa );
}
```

```
public PessoaEmpresa( String id, String no, String ma, double sa ) {
super ( id, no );
setMatricula ( ma );
setSalario ( sa );
}
public void cadastrar( String id, String no, String ma, double sa) {
super.cadastrar(id, no);
setMatricula ( ma );
setSalario ( sa );
}
public void entradaDados( ) {
Scanner entrada = new Scanner( System.in );
super.entradaDados();
System.out.println( "Matrídula :" );
setMatricula ( entrada.nextLine() );
System.out.println( "Salário :" );
setSalario ( Double.parseDouble( entrada.nextLine() ) );
entrada.close();
}
public void imprimir( ) {
super.imprimir();
System.out.println( "Matrídula :" + getMatricula() );
System.out.println( "Salário :" + getSalario() );
}
}
```

Indicação de Leitura Específica

Herança

DEITEL, Paul. Java: como programar (Biblioteca Virtual). 10. ed. São Paulo: Pearson, 2017. [Páginas: 284 ? 286]

Herança

FURGERI, Sérgio. Java 8 ? ensino didático: desenvolvimento e implementação de aplicações. São Paulo: Érica, 2015. [Páginas: 119 ? 124]

Aplicação: articulação teoria e prática

Exercício:

Uma loja vende 3 tipos de produto: livro, CD e software. Para todos os produtosexiste código, descrição, preço e peso. Para o CD existe o nome da banda, para o livro existe o nome do autor e para o software existe a categoria. Para entregar um produto, o cálculo do frete é feito multiplicando o peso do produto por R$ 6,50. Implemente as classes que retratam esse cenário.

Considerações Adicionais