Orientação a objetos

Herança

Herança

- Herança é uma forma de reutilização de software.
- Permite além de reutilização, manutenção simples e eficiente: alterações em cascata.
- Em Java a herança é observada através da palavra *extends*
- Java não permite herança múltipla: um artifício é utilizar uma classe, que herda de outra classe, que herda de outra classe e assim por diante.
- Exemplo:
  - classe 2 *extends* classe 1
  - classe 3 *extends* classe 2

Herança

```
3  public class Carro {
4
5      protected String nome;
6
7      public String getNome() {
8          return nome;
9      }
10
11     public void setNome(String nome) {
12         this.nome = nome;
13     }
14
15     public void exibeMsg() {
16         System.out.println("Estou na classe Carro\nO nome do carro é: "+nome);
17     }
18 }
```

Herança

- A classe Onibus herda os atributos e os métodos da classe Carro

```
3  public class Onibus extends Carro{
4      protected String modelo;
5
6      public String getModelo() {
7          return modelo;
8      }
9
10     public void setModelo(String modelo) {
11         this.modelo = modelo;
12     }
13 }
```

Herança

```
3  public class Principal {
4
5      public static void main(String[] args) {
6          Onibus o = new Onibus();
7          o.setNome("Marcopolo");
8          o.setModelo("OF-1519");
9
10         o.exibeMsg();
11
12         System.out.println("O nome do carro: "+o.getNome());
13         System.out.println("Modelo do onibus: "+o.getModelo());
14     }
15 }
```

setNome –> Classe Carro

setModelo –> Classe Onibus

Este método está na classe Carro!

getNome –> Classe Carro

getModelo –> Classe Onibus

Herança

- Jogo rápido [1]:
  - Crie uma classe Animal com os seguintes atributos: nome, idade e som. Em seguida, crie uma classe Cachorro que herda de Animal e adiciona um atributo raça. Crie um método na classe Cachorro chamado latir() que exibe o som do cachorro.

Herança

```
3  public class Animal {
4      private String nome;
5      private int idade;
6      protected String som;
7      public String getNome() {
8          return nome;
9      }
10     public void setNome(String nome) {
11         this.nome = nome;
12     }
13     public int getIdade() {
14         return idade;
15     }
16     public void setIdade(int idade) {
17         this.idade = idade;
18     }
19     public String getSom() {
20         return som;
21     }
22     public void setSom(String som) {
23         this.som = som;
24     }
25 }
```

Herança

```
3  public class Cachorro extends Animal{
4      private String raca;
5
6      public String getRaca() {
7          return raca;
8      }
9
10     public void setRaca(String raca) {
11         this.raca = raca;
12     }
13
14     public void latir() {
15         System.out.println(som);
16     }
17 }
```

Herança

```
3  public class Principal {
4
5      public static void main(String[] args) {
6          Cachorro c = new Cachorro();
7          /* atributos da classe Animal */
8          c.setNome("Amarelo");
9          c.setIdade(10);
10         c.setSom("Au-Au");
11
12         /* atributos da classe Cachorro */
13         c.setRaca("Vira-lata");
14
15         c.latir();
16     }
17 }
```

Herança

- Uma classe Pessoa, clássica

```
 3 public class Pessoa {
 4     protected String nome;
 5     protected int idade;
 6     public Pessoa(String nome, int idade) {
 7         this.nome = nome;
 8         this.idade = idade;
 9     }
10 
11     public void exibeDados() {
12         System.out.println("Nome: "+nome+"\nIdade: "+idade);
13     }
14 }
```

Herança

- Uma classe Principal clássica

```
3 public class Principal {
4
5     public static void main(String[] args) {
6         Pessoa p = new Pessoa("Ricardo", 40);
7         p.exibeDados();
8     }
9 }
```

Herança

- Jogo rápido [2]:
  - Quero criar uma classe PessoaJuridica que herde a classe Pessoa, como faço?
  - Atributos da classe PessoaJuridica:
    - String CNPJ
    - String socio
    - String dtAbertura

Herança

```
3  public class PessoaJuridica extends Pessoa {
4      protected String CNPJ;
5      protected String socio;
6      protected String dtAbertura;
7      public PessoaJuridica(String nome, int idade, String CNPJ, String socio, String dtAbertura) {
8          super(nome, idade);
9          this.CNPJ = CNPJ;
10         this.socio = socio;
11         this.dtAbertura = dtAbertura;
12     }
13 }
```

- O comando super serve para chamar o construtor da superclasse, ou seja, da classe que é herdada.

- Ele sempre é chamado, mesmo quando não está explícito no código, quando for explicitado deve ser o primeiro item dentro do construtor.

Herança

- No exemplo é chamado o super dentro do construtor onde é passado os parâmetros (nome e idade) da classe que é herdada (Pessoa).

- E abaixo são atribuído os atributos da classe PessoaJuridica.

Herança múltipla (?)

```
3  public class Desenho {
4      protected String nomeAutor;
5
6      public String getNomeAutor() {
7          return nomeAutor;
8      }
9
10     public void setNomeAutor(String nomeAutor) {
11         this.nomeAutor = nomeAutor;
12     }
13 }
```

Herança múltipla (?)

```
3  public class Desenho2D extends Desenho{
4      protected int largura;
5      protected int altura;
6
7      public Desenho2D(int largura, int altura) {
8          super();
9          this.largura = largura;
10         this.altura = altura;
11     }
12 }
```

Herança múltipla (?)

```
3  public class Quadrado extends Desenho2D{
4      protected String desc;
5      public Quadrado(int largura, int altura, String desc) {
6          super(largura, altura);
7          this.desc = desc;
8
9      }
10     public void exibeDados() {
11         System.out.println("Largura: "+this.largura);
12         System.out.println("Altura: "+this.altura);
13         System.out.println("Descrição: "+this.desc);
14         System.out.println("Autor: "+this.nomeAutor);
15     }
16 }
```

Herança múltipla (?)

```
3 public class PrincipalDesenho {
4     public static void main(String[] args) {
5         Quadrado q = new Quadrado(100, 150, "Quadrado do Ricardo");
6         q.exibeDados();
7     }
8 }
9
```

Herança múltipla (?)

- Como citado anteriormente, o Java não permite herança múltipla, mas neste exemplo anterior é apresentado a classe Desenho2D herda Desenho e a classe Quadrado herda Desenho 2D.

- Consequentemente, a classe Quadrado herda os atributos e métodos do Desenho2D que também herda os atributos e métodos da classe Desenho

Herança

- Jogo rápido [3]:
  - O que apareceu como Autor na mensagem?
  - Corrija o problema fazendo a atribuição na main.

Herança

- Jogo rápido [3]:

```
3 public class PrincipalDesenho {
4     public static void main(String[] args) {
5         Quadrado q = new Quadrado(100, 150, "Quadrado do Ricardo");
6         q.setNomeAutor("Ricardo Frohlich");
7         q.exibeDados();
8     }
9 }
```

Herança

- Jogo rápido [4]:
  - Altere o código deste exemplo e adicione um construtor na classe Desenho recebendo por parâmetro o atributo autor.

Herança

- Jogo rápido [4]:

```
3  public class Desenho {
4      protected String nomeAutor;
5
6      public Desenho(String nomeAutor) {
7          super();
8          this.nomeAutor = nomeAutor;
9      }
10
11     public String getNomeAutor() {
12         return nomeAutor;
13     }
14
15     public void setNomeAutor(String nomeAutor) {
16         this.nomeAutor = nomeAutor;
17     }
18 }
```

Herança

- Jogo rápido [4]:
  - Apareceu um erro, certo?
  - Na classe Desenho2D? -> No super(), porquê?

Herança

- Jogo rápido [4]:
  - Pois ao criarmos um construtor na classe Desenho, precisamos receber o parâmetro na chamada do construtor da super classe.

```
3  public class Desenho2D extends Desenho{
4      protected int largura;
5      protected int altura;
6
7      public Desenho2D(int largura, int altura, String nomeAutor) {
8          super(nomeAutor);
9          this.largura = largura;
10         this.altura = altura;
11     }
12 }
```

Herança

- Jogo rápido [4]:
  - Mas agora apareceu o erro na classe Quadrado, certo? E porquê?
  - Mesmo motivo, precisamos passar o parâmetro.

Herança

```
3  public class Quadrado extends Desenho2D{
4      protected String desc;
5      public Quadrado(int largura, int altura, String nomeAutor, String desc) {
6          super(largura, altura, nomeAutor);
7          this.desc = desc;
8
9      }
10     public void exibeDados() {
11         System.out.println("Largura: "+this.largura);
12         System.out.println("Altura: "+this.altura);
13         System.out.println("Descrição: "+this.desc);
14         System.out.println("Autor: "+this.nomeAutor);
15     }
16 }
```

Herança

- Jogo rápido [4]:
  - E agora o erro é na Main pelo mesmo motivo:

```
3 public class PrincipalDesenho {
4     public static void main(String[] args) {
5         Quadrado q = new Quadrado(100, 150, "Ricardo Frohlich", "Quadrado do Ricardo");
6         q.exibeDados();
7     }
8 }
```

Exercícios

- 1) Crie uma classe Figura com os seguintes atributos: cor e preenchido. Em seguida, crie uma classe Retangulo que herda de Figura e adiciona dois atributos: largura e altura. Crie um método na classe Retangulo chamado calcularArea() que retorna a área do retângulo (altura * largura).

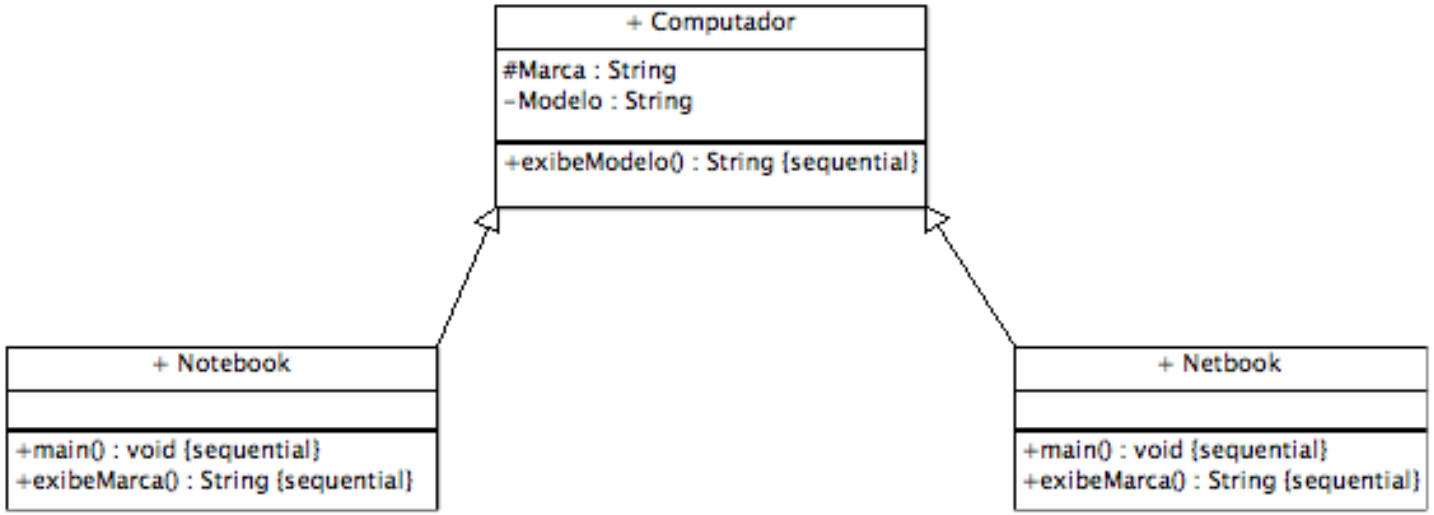
- 2) Crie uma classe "Pessoa" com um construtor que recebe um parâmetro "nome" e um método "trabalhar" que imprime "A pessoa está trabalhando". Crie uma subclasse "Funcionario" que herda da classe "Pessoa". No construtor da classe "Funcionario", utilize o comando "super" para chamar o construtor da classe "Pessoa" e passar o parâmetro "nome". Adicione um método "trabalhar" na classe "Funcionario" que imprime "O funcionário está trabalhando".

Exercícios

- 3) Crie uma classe chamada Pessoa que herda da classe SerHumano os atributos nome e idade e o método falar. A classe SerHumano também possui herança e herda o atributo tipo e o método andar da classe Animal. Desse modo, crie uma classe Principal para exibir na tela o conteúdo de todos os atributos e realizar a chamada de todos os métodos envolvidos no processo. O método falar retorna a string “Nem todos falam” e o método andar imprime na tela a string “Todos andam, mas o modo é variado”. Solicite ao usuário para informar o nome, a idade e o tipo.

Exercícios

- 4) Escreva um programa orientado a objetos baseado no diagrama de classes da UML apresentado abaixo:



- Dica: no método main das classes herdeiras, crie a opção do usuário inserir a marca do computador. Já, o modelo será sempre “Portátil”.