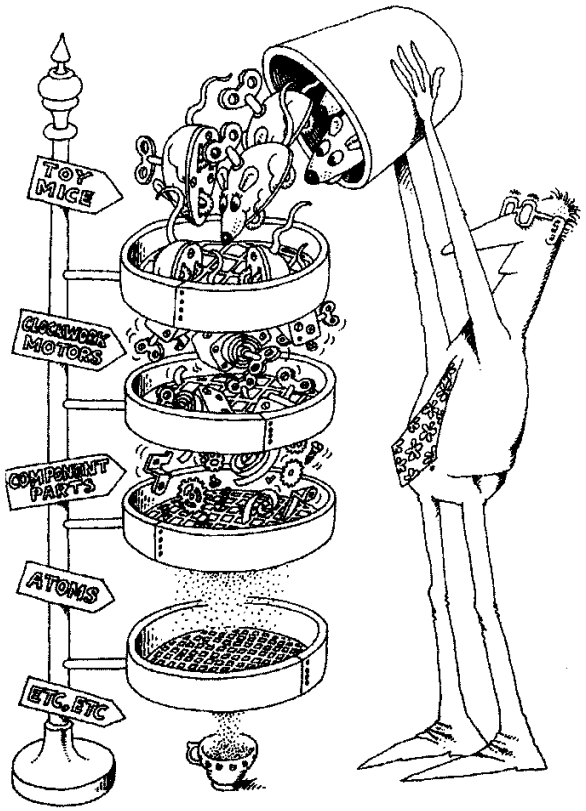
Hierarquia

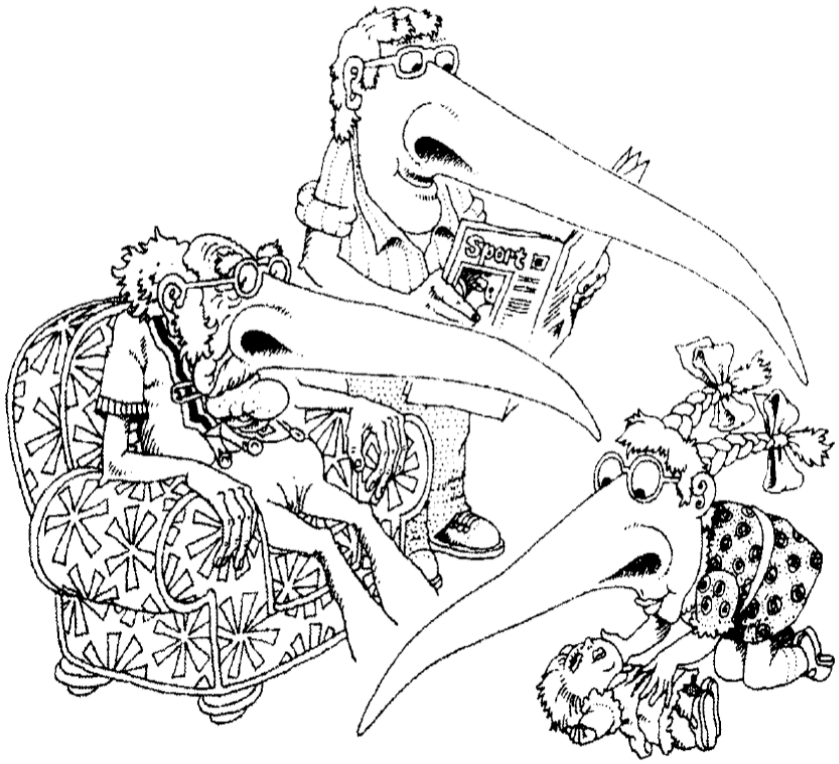
- Os objetos devem ser organizados no sistema de forma hierárquica.



Fonte: livro “Object-Oriented Analysis and Design with Applications”

Hierarquia

- Objetos herdam atributos e métodos dos seus ancestrais na hierarquia.

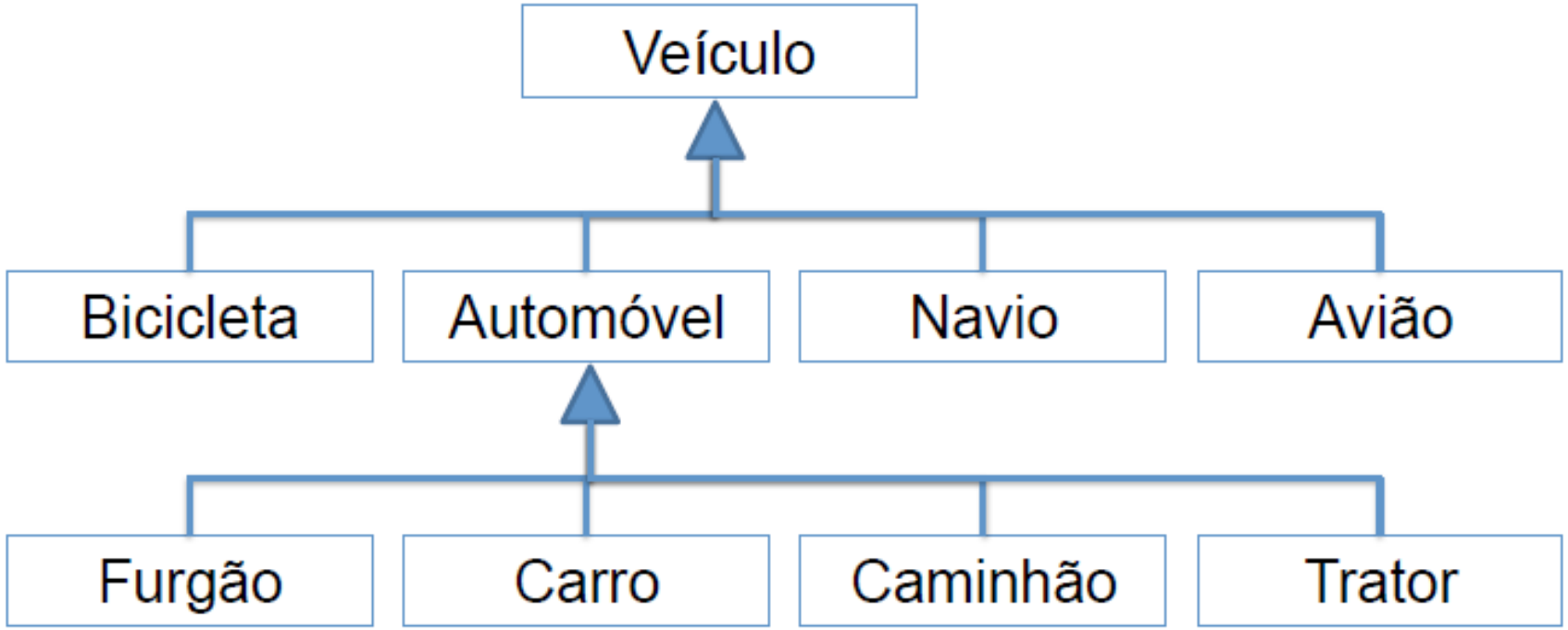


Fonte: livro “Object-Oriented Analysis and Design with Applications”

Herança

- Para viabilizar a hierarquia entre objetos, as classes são organizadas em estruturas hierárquicas.
  - A classe que forneceu os elementos herdados é chamada de superclasse;
  - A classe herdeira é chamada de subclasse;
  - A subclasse pode herdar os métodos e atributos de suas superclasses;
  - A subclasse pode definir novos atributos e métodos específicos;
  - As subclasses são mais especializadas do que as suas superclasses, mais genéricas.

Exemplo de Herança



Teste da Leitura: "*subclasse* é **um** *superclasse*"
Ex.: Carro é um Automóvel; Trator é um Veículo; ...

Exemplo de Herança

- Relembrando a classe Carro...

```
public class Carro {
 private int velocidade;
 public Carro(int velocidadeInicial) {
  velocidade = velocidadeInicial;
 }
 public void acelera() {
  velocidade++;
 }
 public void freia() {
  velocidade--;
 }
}
```

Exemplo de Herança

- Criando um carro inteligente...

```
public class CarroInteligente extends Carro {
	public CarroInteligente(int velocidadeInicial) {
		super(velocidadeInicial);
	}
	public void estaciona() {
		// código mágico para estacionar sozinho
	}
}
```

- Usando um carro inteligente...

```
CarroInteligente tiguan = new CarroInteligente(10);
for (int i = 10; i > 0; i--) {
	tiguan.freia();
}
tiguan.estaciona();
```

De onde veio isso? :o

Exemplo de Herança

- Criando um carro de corrida...

```
public class CarroCorrida extends Carro {
	public CarroCorrida(int velocidadeInicial) {
		super(velocidadeInicial);
	}
	public void acelera() {
		for(int i = 1; i <=5; i++)
			super.acelera();
	}
}
```

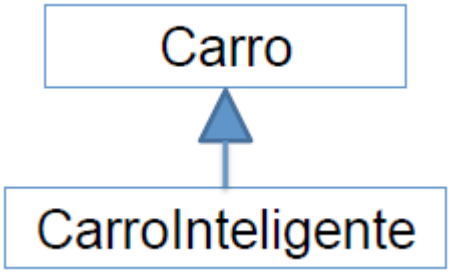
- Usando um carro de corrida...

```
CarroCorrida f1 = new CarroCorrida(10);
f1.acelera();
```

Qual a velocidade agora: :/

Compatibilidade

- Qualquer subclasse é compatível com a sua superclasse. Contudo, a reciproca não é verdadeira...




```
Carro c = new CarroInteligente(20);
c.acelera();
c.freia();
```

✓

```
CarroInteligente c = new Carro(20);
c.acelera();
c.freia();
c.estaciona();
```

✗

Herança em Java

- Uma classe só pode herdar de uma outra classe (herança simples).
- Caso não seja declarada herança, a classe herda da classe *Object*.
  - Ela define o método *toString*(), que retorna a representação em *String* do objeto.
  - Qualquer subclasse pode sobrescrever o método *toString*() para retornar o que ela deseja.
  - Veja os demais métodos da classe *Object* em https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html

Polimorfismo

- Uma subclasse pode redefinir (sobrescrever) um método herdado. Este mecanismo é chamado de **polimorfismo**.
- O polimorfismo se realiza através da recodificação de um ou mais métodos herdados por uma subclasse.
  - Em tempo de execução, o Java saberá qual implementação deve ser usada.
- Tipos de polimorfismo: sobrecarga (*overload*) e sobreposição (*override*).

Polimorfismo

- A **sobrecarga** de métodos consiste em criar métodos distintos com nomes iguais em uma mesma classe, contanto que suas listas de argumentos sejam diferentes.
  - Se o programa encontrar dois métodos com argumentos iguais ele não saberá qual chamar e haverá um erro em seu programa.
  - A sobrecarga é muito utilizada em construtores.
  - Exemplo:

```
public class Calculadora{
   public int soma(int a, int b){
      return a+b;
    }
    public double soma(double a, double b){
       return a+b;
    }
}
```

Polimorfismo

- A **sobreposição** permite que os métodos da classe pai sejam reescritos nas classes filhas, transformando métodos genéricos em específicos e implementando outras funcionalidades.
  - Os métodos que serão sobrepostos devem possuir o mesmo nome, tipo de retorno e quantidade de parâmetros do método inicial.
  - Exemplo:

```
public class Calculadora{
   public int soma(int a, int b){
      return a+b;     }
}
public class CalculadoraCientifica extends Calculadora{
    public int soma(int a, int b){
       if((a>0)&&(b>=0)))
           System.out.println("Numeros positivos");
       return a+b;     }
}
```

Exercícios

Mais exemplos... (1)

- Em um sistema de loja, há 3 tipos de usuário: gerente, funcionário e cliente. Todo usuário tem nome e senha. O cliente possui, além do nome e senha, outros dados cadastrais. O funcionário possui métodos relacionados a venda de produtos. O gerente pode fazer tudo que o funcionário pode e também fechamento do caixa. Como é a hierarquia de herança desse sistema no que se refere a controle de usuários?

Mais exemplos... (2)

- O que está definido no código abaixo?

```
import java.util.Date;
public class Pessoa {
        public String nome;
        public String cpf;
        public Date data_nascimento;

        public Pessoa(String _nome, String _cpf, Date _data) {
                this.nome = _nome;
                this.cpf = _cpf;
                this.data_nascimento = _data;
        }
}
```