LTP 1 – Java Básico

Aula 8

Tópicos

1 - Herança

1.1 - Introdução

Herança é uma das principais formas de reutilização de software na qual uma nova classe é criada, absorvendo membros de uma classe existente e aprimorada com capacidades novas ou modificadas. Assim, os programadores economizam tempo durante o desenvolvimento de programa reutilizando software de alta qualidade testado e depurado.

Em resumo, ao criar uma nova classe, em vez de declarar membros completamente novos, o programador pode designar que a nova classe deverá herdar membros de uma classe existente. Essa classe existente é chamada de **superclasse** e a nova classe é chamada **subclasse**.

Observação:

- cada subclasse pode tornar-se a superclasse para futuras subclasses.
- As superclasses tendem a ser "mais gerais" e as subclasses mais específicas.

Para utilizar a herança numa subclasse devemos utilizar extends, que é uma palavra reservada do Java, usada para definir uma relação de herança entre classes.

```
public class Quadrupede {//superclasse
   public int patas=4;
}
```

```
public class Cachorro extends Quadrupede {//subclasse
   public String voz="late";
}
```

```
import java.util.Scanner;
public class Exemplo {
   public static void main(String args[]){
      Cachorro zeca = new Cachorro();
      System.out.println("um cachorro tem "+zeca.patas+" patas");
      System.out.println("todo cachorro "+zeca.voz);
   }
}
```

Saída:

```
cachorro tem 4 patas
todo cachorro late
```

Exemplo um pouco mais complexo:

```
public class Pessoa {//superclasse
   private String nome;
   private String cpf;
   public String getNome() {
      return nome;
   }
   public void setNome(String nome) {
      this.nome = nome;
   }
   public String getCpf() {
      return cpf;
   }
   public void setCpf(String cpf) {
      this.cpf = cpf;
   }
}
```

Professores e Alunos também têem esses atributos. Portanto, nada mais prático que criar subclasses de Pessoa para representa-los. As novas classes criadas possuem suas características (atributos e métodos) próprias, mas possuem também propriedades comuns: os atributos nome e CPF.

```
public class Aluno extends Pessoa{//subclasse
   private String matricula;
   public String getMatricula() {
      return matricula;
   }
   public void setMatricula(String matricula) {
      this.matricula = matricula;
   }
}
```

```
public class Professor extends Pessoa{//subclasse
   private String titulacao;
   public String getTitulacao() {
      return titulacao;
   }
   public void setTitulacao(String titulacao) {
      this.titulacao = titulacao;
   }
}
```

```
import java.util.Scanner;
public class Exemplo {
   public static void main(String args[]){
      Scanner a = new Scanner(System.in);
```

```
        Professor var = new Professor();
        Aluno aux = new Aluno();
        System.out.println("Qual o nome do aluno?");
        aux.setNome(a.nextLine());
        System.out.println("Qual o CPF do "+aux.getNome()+"?");
        aux.setCpf(a.nextLine());
        System.out.println("Qual a matrícula do CPF: "+aux.getCpf()+"?");
        aux.setMatricula(a.nextLine());
        System.out.println("Qual o nome do professor?");
        var.setNome(a.nextLine());
        System.out.println("Qual a titulação do Professor "+var.getNome()+"?");
        var.setTitulacao(a.nextLine());
        System.out.println("O "+var.getTitulacao()+" "
                           +var.getNome()+" é professor do "+aux.getNome() );
    }
}
```

Saída:

```
Qual o nome do aluno?
Marcos Nascimento
Qual o CPF do Marcos Nascimento?
34212398923
Qual a matrícula do CPF: 34212398923?
2016123
Qual o nome do professor?
Salvador Melo
Qual a titulação do Professor Salvador Melo?
Mestre
O Mestre Salvador Melo é professor do Marcos Nascimento
```

1.2 - Superclasse direta e indireta

- Superclasse direta: quando a subclasse herda explicitamente;
- Superclasse indireta: é qualquer subclasse acima da classe direta.

Exemplo:

```
public class Mamifero {//superclasse indireta
    public String alimento="leite";
}
```

```
public class Quadrupede extends Mamifero {//superclasse direta
    public int patas=4;
}
```

```
public class Cachorro extends Quadrupede {//subclasse
    public String voz="late";
}
```

```
import java.util.Scanner;
public class Exemplo {
    public static void main(String args[]){
        Cachorro zeca = new Cachorro( );
```

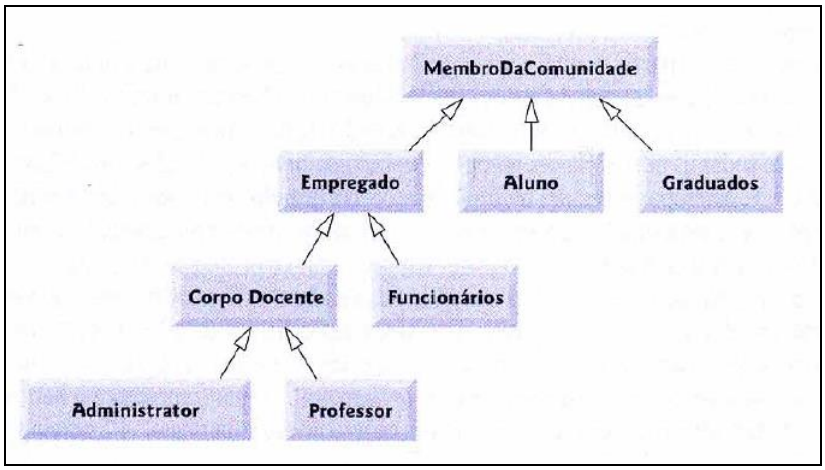
```
        System.out.println("zeca tem "+zeca.patas+
                           " patas e "+zeca.voz+
                           " quando quer "+zeca.alimento);
    }
}
```

Tipos de relacionamento

- **É um**: representa a herança, um objeto de uma subclasse também pode ser tratado como um objeto de uma superclasse. Por exemplo, carro é um veículo;
- **Tem um**: representa a composição, um objeto contém uma ou mais referências de objetos como membros. Carro tem uma direção.

Superclasse: ContaBancaria
Subclasses: ContaPoupança, ContaCorrente, ContaInvestimento

Assim, os relacionamentos de herança formam estruturas hierárquicas do tipo árvore. Por exemplo a árvore da comunidade acadêmica:



1.3 - Membros protected

- public: esses membros de uma classe são acessíveis por todos
- private: esses membros não são herdados por suas subclasses
- **protected**: oferece um acesso intermediário entre public e private. Eles só pode ser acessados pelos membros da superclasse, das suas subclasses e por membros de outras classes do mesmo pacote.

Os métodos de uma subclasse podem referir-se a membros protected herdados da superclasse simplesmente utilizando o nome do membro.

1.4 - Usando a palavra-chave super

Quando um método de uma subclasse sobrescrever um método de uma superclasse, o método da superclasse pode ser acessado a partir da subclasse precedendo o nome do método da superclasse com a palavra-chave super e um separador de ponto.

Exemplo:

```
public class Familia {
   public void mostraNome(){
      System.out.println("Junior");
```

```
    }
}
```

```
public class Amigos extends Familia{
    @Override//marca que o método da superclasse foi reescrito
    public void mostraNome(){
        System.out.println("Salvador");
        super.mostraNome();//chama o método da superclasse
    }
}
```

```
import java.util.Scanner;
public class Exemplo {
    public static void main(String args[]){
        Amigos var = new Amigos();
        var.mostraNome();
    }
}
```

2 - Exercícios

2.1 - Faculdade:

- Crie uma superclasse chamada Funcionários, com as seguintes características:
  i. Nome;
  ii. Endereço;
  iii. Telefone.
- Crie uma subclasse chamada Professor, com as seguintes características:
  i. Disciplina Ministrada (basta uma);
  ii. Valor da hora/aula.
- Crie outra subclasse chamada Administrativo, com as seguintes cartacterísticas:
  i. Cargo;
  ii. Salário.
- Crie uma classe principal que:
  i. Le o nome de um funcionário;
  ii. Usa um switch para que escolha entre professor e administrativo;
  iii. Le e mostra todos os dados de um funcionário com base na opção escolhida.

2.2 - Dados dos carros:

- Uma classe MeioDeTransporte com atributos ligado (boolean) e velocidade (inteiro) e métodos liga() e desliga (). O método ligado torna o atributo ligado true e o método desliga torna o atributo ligado false, além de tornar a velocidade zero. Crie também métodos get e set para modificar o atributo velocidade, sendo que a velocidade não pode ser negativa.
- Uma subclasse de MeioDeTransporte chamada Carro. Carro deve ter os atributos marca (String), nome (String) e quilometragem (tipo int) e os métodos necessários para ler os atributos e alterá-los (get e set). A quilometragem não pode ser negativa, nem ultrapassar o valor 999999. A velocidade do Carro não pode ser negativa, nem ultrapassar 200.
- Uma classe que contenha o método main que permita ao usuário digitar os dados de 5 carros. Adicione os objetos em um array de carro. Por fim exiba os dados dos 5 carros em uma JOptionPane.showMessageDialog.