```
package edu.univasf.poo;
public class Pacote
{
   private String nomeRemetente;
   private String enderecoRemetente;
   private String cidadeRemetente;
   private long CEPRemetente;
   private String nomeDestinatario;
   private String enderecoDestinatario;
   private String cidadeDestinatario;
   private long CEPDestinatario;
   private double peso;
   private static double custoPorQuilo = 1.7;
   public Pacote(double novoPeso, double novoCustoPorQuilo)
   {
      setPeso((novoPeso>0)?novoPeso:0.1);
      setCustoPorQuilo((novoCustoPorQuilo>0)?novoCustoPorQuilo:
getCustoPorQuilo());
   }
   public Pacote(Pacote pacote)
   {
      this(pacote.peso, pacote.custoPorQuilo);
   }
```

```
public static void setCustoPorQuilo(double novoCustoPorQuilo)
{
  custoPorQuilo = novoCustoPorQuilo;
}
public static double getCustoPorQuilo()
{
  return custoPorQuilo;
}
public void setPeso(double novoPeso)
{
        peso = novoPeso;
}
public double getPeso()
{
  return peso;
}
public double calculaCusto()
{
  return getPeso()*getCustoPorQuilo();
}
public void setNomeRemetente(String novoNomeRemetente)
{
  nomeRemetente = novoNomeRemetente;
}
```

```
public void setEnderecoRemetente(String novoEnderecoRemetente)
{
  enderecoRemetente = novoEnderecoRemetente;
}
public void setCidadeRemetente(String novaCidadeRemetente)
{
  cidadeRemetente = novaCidadeRemetente;
}
public void setCEPRemetente(long novoCEPRemetente)
{
  CEPRemetente = novoCEPRemetente;
}
public void setNomeDestinatario(String novoNomeDestinatario)
{
  nomeDestinatario = novoNomeDestinatario;
}
public void setEnderecoDestinatario(String novoEnderecoDestinatario)
{
  enderecoDestinatario = novoEnderecoDestinatario;
}
public void setCidadeDestinatario(String novaCidadeDestinatario)
{
  cidadeDestinatario = novaCidadeDestinatario;
}
```

```
public void setCEPDestinatario(long novoCEPDestinatario)
{
  CEPDestinatario = novoCEPDestinatario;
}
public String getNomeRemetente()
{
  return nomeRemetente;
}
public String getEnderecoRemetente()
{
  return enderecoRemetente;
}
public String getCidadeRemetente()
{
  return cidadeRemetente;
}
public long getCEPRemetente()
{
  return CEPRemetente;
}
public String getNomeDestinatario()
{
  return nomeDestinatario;
}
```

```
  public String getEnderecoDestinatario()
   {
     return enderecoDestinatario;
   }
   public String getCidadeDestinatario()
   {
     return cidadeDestinatario;
   }
   public long getCEPDestinatario()
   {
     return CEPDestinatario;
   }
   public String toString()
   {
          return String.format("\nDe %s\n%s\n%s\nCEP %d\nPara
%s\n%s\n%s\nCEP %d",
          nomeRemetente, enderecoRemetente, cidadeRemetente,
          CEPRemetente, nomeDestinatario, enderecoDestinatario,
          cidadeDestinatario, CEPDestinatario);
   }
}
```

```
import edu.univasf.poo.Pacote;
public class TestePacote
{
  public static void main (String args[])
  {
        if (args.length==2)
        {
          Pacote pacote = new Pacote(args[0], args[1]);
          pacote.setNomeRemetente("Marcelo Linder");
          pacote.setEnderecoRemetente("Minha Rua, 10");
          pacote.setCidadeRemetente("Juazeiro");
          pacote.setCEPRemetente(90380220);
          pacote.setNomeDestinatario("Fulana da Silva");
          pacote.setEnderecoDestinatario("Casa Dela");
          pacote.setCidadeDestinatario("Salvador");
          pacote.setCEPDestinatario(90280110);
          System.out.println(pacote);
        }
  }
}
```

```
import edu.univasf.poo.Pacote;
public class TestePacote2 {
   public static void main (String args[]) {
      Pacote pacote1 = new Pacote(12.1, 2.7);
      pacote1.setNomeRemetente("Marcelo Linder");
      pacote1.setEnderecoRemetente("Minha Rua, 10");
      pacote1.setCidadeRemetente("Juazeiro");
      pacote1.setCEPRemetente(90380220);
      pacote1.setNomeDestinatario("Fulana da Silva");
      pacote1.setEnderecoDestinatario("Casa Dela");
      pacote1.setCidadeDestinatario("Salvador");
      pacote1.setCEPDestinatario(90280110);
      System.out.println(pacote1);
      Pacote pacote2;
      if (pacote2==null) /* ERRO! pacote2 não foi inicializado. */
         System.out.println("\nInicia com o valor null.");
      else
         System.out.println("\nInicia com lixo.");
      pacote2 = pacote1; /*Copia a referência de pacote1 e não o conteúdo de seus
atributos */
      System.out.println(pacote2);
      pacote1.setNomeRemetente("Ciclano dos Santos");
      System.out.println(pacote2); //Apresenta “De Ciclano dos Santos ...”
      System.out.println(pacote1); //Apresenta “De Ciclano dos Santos ...”
   }
}
```

Observações importantes

HERANÇA

Conforme estudamos, a herança constitui uma das principais características da OO e, portanto, a linguagem Java não cercearia seus programadores de usufruir de tal propriedade.

Visando uma maior objetividade analisaremos um exemplo da implementação de herança na linguagem Java e discutiremos como este processo ocorre.

No exemplo a seguir é declara a classe Circulo que herda da classe Ponto2D declarada anteriormente e neste momento redefinida.

```
package edu.univasf.poo;
public class Ponto2D
{
   private float x;
   private float y;
   public Ponto2D(float novoX, float novoY)
   {
      setX(novoX);
      setY(novoY);
   }
   public void setX (float novoX)
   {
      x = novoX;
   }
   public float getX ()
   {
      return x;
   }
   public void setY (float novoY)
   {
      y = novoY;
   }
```

```
public float getY ()
   {
      return y;
   }
   public String toString()
   {
      return String.format("\n(%f, %f)\n", getX(), getY());
   }
}
```

```
package edu.univasf.poo;
import java.lang.Math; /* esta importação já ocorre implicitamente */
public class Circulo extends Ponto2D
{
  private float raio;
  private float distanciaAoCentro (float a, float b)
  {
    return((float)Math.sqrt(Math.pow((a-getX()),2)+Math.pow((b-
     getY()),2)));
  }
  public Circulo(float x, float y, float r)
  {
      super(x, y);
      setRaio(r);
  }
  public void setRaio(float novoRaio)
  {
    raio = novoRaio;
  }
  public float getRaio()
  {
    return raio;
  }
}
```

```
  public float area()
  {
    return((float)(Math.PI*Math.pow(getRaio(),2)));
  }
  public boolean pontoPertenceAoCirculo (Ponto2D p)
  {
    return(getRaio()>=distanciaAoCentro(p.getX(),
     p.getY())?true:false);
  }
  public String toString()
  {
    return String.format("\nCentro: (%f, %f)\nRaio:%f", x, y, getRaio());
    /* Não é possível acessar as variáveis de instância herdadas x e y
     diretamente, pois estas foram especificadas na classe base como
     privadas. Este problema pode ser solucionado utilizando-se os
     métodos set e get herdados(como a seguir) ou especificando
     estas variáveis de instância da classe base como protegidas
     (protected) */
    return String.format("\nCentro: (%f, %f)\nRaio:%f", getX(), getY(),
     getRaio());
  }
}
```

```
import edu.univasf.poo.Ponto2D;
import edu.univasf.poo.Circulo;
public class TesteCirculo
{
  public static void main(String args[])
  {
    Ponto2D p = new Ponto2D((float)3.5, (float)4.1);
    float x, y;
    System.out.print ("\n\nO ponto incialmente encontra-se em ");
    System.out.print (p);
    Circulo c = new Circulo(p.getX(), p.getY(), 5);
    System.out.println ("Dados do circulo: ");
    System.out.print (c);
    System.out.print ("\n\nA area da circunferencia eh " + c.area());
    System.out.print ("\n\nO ponto");
    if (c.pontoPertenceAoCirculo(p))
      System.out.print (" ");
    else
      System.out.print (" nao ");
    System.out.print ("pertence ao circulo.");
  }
}
```

SOBREPOSIÇÃO

Podemos na linguagem Java redefinir um método definido em uma superclasse em uma de suas subclasses.

Este mecanismo é denominado sobreposição.

Vamos agora analisar um exemplo da exploração deste conceito:

```
class Base {
  public void func(){
    System.out.println("Esta eh func() de Base");
  }
}
class Derivada1 extends Base {
  public void func(){
    System.out.println("Esta eh func() de Derivada1");
  }
} //este arquivo continua no próximo slide
```

Linguagem de Programação Java

```
class Derivada2 extends Base
{
   public void func()
   {
      System.out.println("Esta eh func() de Derivada2");
   }
}
public class TesteSobreposicao
{
   public static void main(String args[])
   {
      Base b = new Base();
      Derivada1 d1 = new Derivada1();
      Derivada2 d2 = new Derivada2();
      b.func();
      d1.func();
      d2. func();
   }
}
```

Exercício:

Com base no exercício do slide 509, considerado que o serviço de correio expresso como o SEDEX®, oferece não uma, mas várias opções de entrega, cada qual com custos específicos. Crie uma hierarquia de herança para representar vários tipos de pacote. Utilize Pacote como a classe base da hierarquia, então inclua as classes PacoteDoisDias e PacoteNoite que derivam de Pacote. A classe derivada PacoteDoisDias deve herdar as funcionalidades da classe básica Pacote, mas também incluir uma variável de classe que representa uma taxa fixa que a empresa de entrega cobra pelo serviço de entrega em dois dias. O construtor de PacoteDoisDias deve receber um valor para inicializar esta variável. PacoteDoisDias deve redefinir o método calculaCusto para que ele calcule o custo de entrega adicionando a taxa fixa ao custo baseado em peso

calculado pelo método calculaCusto da classe básica Pacote. A classe PacoteNoite deve herdar diretamente da classe Pacote e deve conter uma variável de classe adicional para representar uma taxa adicional por quilo cobrada pelo serviço de entrega noturno. PacoteNoite deve redefinir o método culculaCusto para que ele acrescente a taxa adicional por quilo ao custo padrão por quilo antes de calcular o custo de entrega. Defina uma classe de teste que instancie objetos de todos as classes definidas e teste o método calculaCusto para cada uma das mesmas.

```
package edu.univasf.poo;
import edu.univasf.poo.Pacote;
public class PacoteDoisDias extends Pacote
{
  private static double taxaExtra = 5.5;
  public PacoteDoisDias(double novoPeso, double novoCustoPorQuilo,
double novaTaxaExtra)
  {
    super(novoPeso, novoCustoPorQuilo);
    setTaxaExtra((novaTaxaExtra>0)?novaTaxaExtra:getTaxaExtra());
  }
  public static void setTaxaExtra(double novaTaxaExtra)
  {
    taxaExtra = novaTaxaExtra;
  }
  public static double getTaxaExtra()
  {
    return taxaExtra;
  }
  public double calculaCusto()
  {
    return super.calculaCusto()+getTaxaExtra();
  }
}
```

```
package edu.univasf.poo;
import edu.univasf.poo.Pacote;
public class PacoteNoite extends Pacote {
  private static double taxaExtraPorQuilo = 1.2;
  public PacoteNoite(double novoPeso, double novoCustoPorQuilo,
double novaTaxaExtraPorQuilo)
  {
        super(novoPeso, novoCustoPorQuilo);
    setTaxaExtraPorQuilo((novaTaxaExtraPorQuilo>0)?
novaTaxaExtraPorQuilo:getTaxaExtraPorQuilo());
  }
  public static void setTaxaExtraPorQuilo(double novaTaxaExtraPorQuilo)
  {
    taxaExtraPorQuilo = novaTaxaExtraPorQuilo;
  }
  public static double getTaxaExtraPorQuilo() {
    return taxaExtraPorQuilo;
  }
  public double calculaCusto() {
    return super.calculaCusto()+getPeso()*getTaxaExtraPorQuilo();
  }
}
```

```
import edu.univasf.poo.Pacote;
import edu.univasf.poo.PacoteDoisDias;
import edu.univasf.poo.PacoteNoite;
public class TestaPacotes
{
  public static void main(String args[])
  {
    Pacote p1 = new Pacote(1, -3);
    PacoteDoisDias p2 = new PacoteDoisDias(1, 0, 0);
    PacoteNoite p3 = new PacoteNoite(1, 0, 0);
    p1.setPeso(5);
    p2.setPeso(5);
    p3.setPeso(5);
    System.out.printf ("Custo do pacote normal: %f\n", p1.calculaCusto());
    System.out.printf ("Custo do pacote dois dias: %f\n",
p2.calculaCusto());
    System.out.printf  ("Custo do pacote noite: %f\n", p3.calculaCusto());
  }
}
```

POLIMORFISMO

Assim como na linguagem C++, a linguagem Java também permite tratar objetos das classes derivadas de forma genérica. Em outras palavras, Java nos permite "programar no geral" em vez de "programar no específico" através a exploração do princípio do polimorfismo.

Já nos utilizamos da sobreposição de um método, ou seja, efetuamos a redefinição de um método definido em uma superclasse em uma de suas subclasses.

Vamos agora analisar este conceito considerando que Java também viabilizará o polimorfismo em tempo de execução, pois possibilita referenciar um objeto de uma classe derivada utilizando para tal uma referência para um objeto da classe base.

UNIVASF

Linguagem de Programação Java

```
class Base {
   public void func(){
      System.out.println("Esta eh func() de Base");
   }
}
class Derivada1 extends Base {
   public void func(){
      System.out.println("Esta eh func() de Derivada1");
   }
}
class Derivada2 extends Base{
   public int teste;
   public void func(){
      System.out.println("Esta eh func() de Derivada2");
   }
}
//o arquivo continua...
```

Linguagem de Programação Java

```
//continuação do arquivo
public class TestePolimorfismo {
   public static void main(String args[]){
      Base b = new Base();
      Derivada1 d1 = new Derivada1();
      Derivada2 d2 = new Derivada2();
      b.func(); //apresentará na tela “Esta eh func() de Base”
      b = d1;
      b.func(); //apresentará na tela “Esta eh func() de Derivada1”
      d2.teste = 7;
      b = d2;
      b.func(); //apresentará na tela “Esta eh func() de Derivada2”
      b.teste = 7; /* ERRO! */
   }
}
```

Exercício: Com base no que vimos, use a hierarquia de herança Pacote criada no exercício do slide 523 para criar um programa que exibe as informações de endereço e calcula os custos de entrega de vários pacotes. O programa deve conter um vetor de referências a objetos da classe Pacote e utilizá-las para referenciar objetos das classes PacoteDoisDias e PacoteNoite; manipulando-o através das operações disponibilizadas pelo seguinte menu.

Digite:

1 – Inserir um pacote para entrega em dois dias;

2 – Inserir um pacote para entrega a noite;

3 – Imprimir os endereços para postagem e custos da postagem;

4 – Imprimir o custo total das postagens.

Para simplificar as manipulações considere que no máximo o vetor possuirá 100 referências para pacotes.