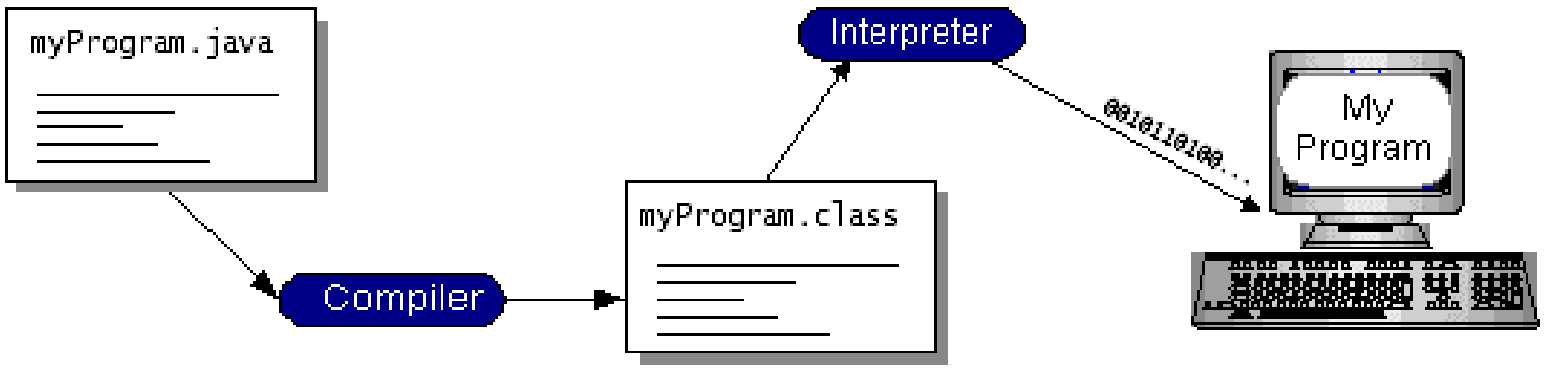
Curso básico de linguagem Java

- Java é tanto uma **linguagem** quanto uma **plataforma**.
  - **linguagem**
    - orientada a objeto, simples, familiar
    - robusta, segura
    - arquitetura neutra, portável
    - alto desempenho
    - interpretada, multiprocessada, dinâmica

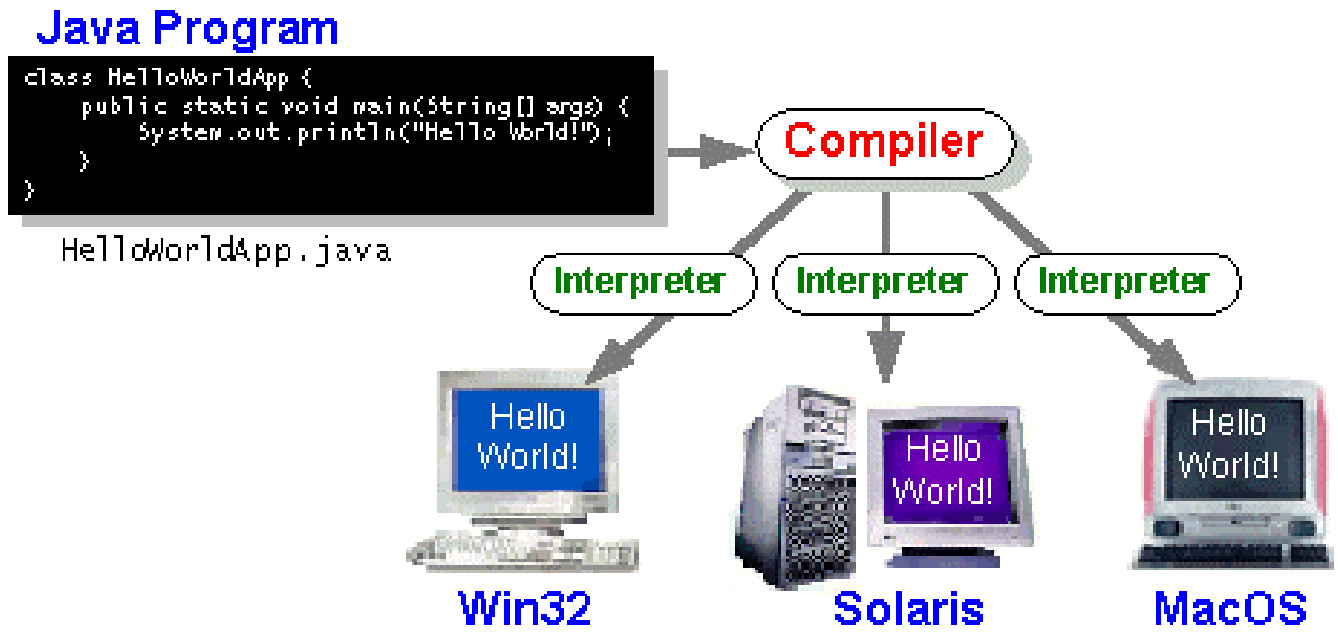
Curso básico de linguagem Java

Linguagem tanto compilada quando interpretada
- Código fonte
- compilado para *Java bytecode.*
- interpretado pela plataforma da *Java Virtual Machine (JVM)*



Curso básico de linguagem Java

Portabilidade

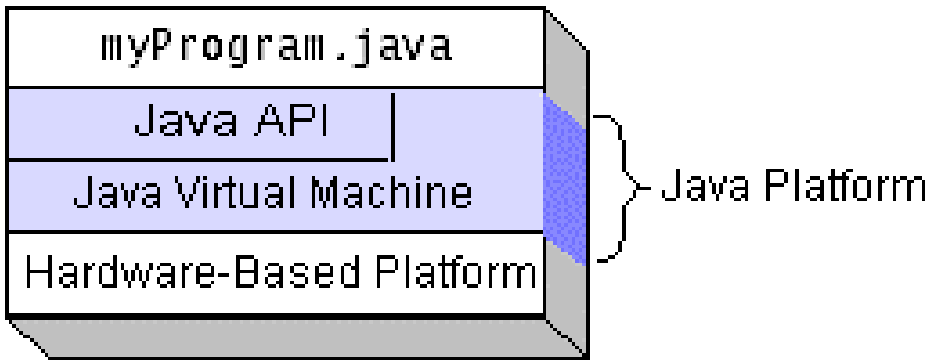


Curso básico de linguagem Java

Plataforma Java:

- *Java Virtual Machine* (Java VM)
- *Java Application Programming Interface* (Java API)

  aplicações gráficas.



Curso básico de linguagem Java

Hello World

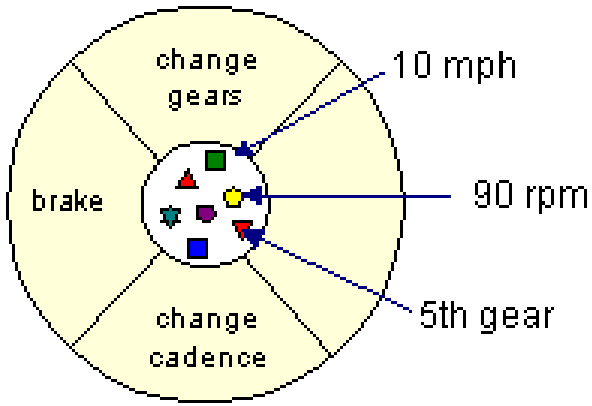
```
/**
 * A classe HelloWorldApp implementa uma aplicação que
 * simplesmente imprime "Hello World!" para a saída padrão.
 */
class HelloWorldApp{
    public static void main(String[ ] args) {
        System.out.println("Hello World!"); // Imprime o string.
    }
}
```

Curso básico de linguagem Java

objetos e classes

Objetos no mundo real: estado e comportamento

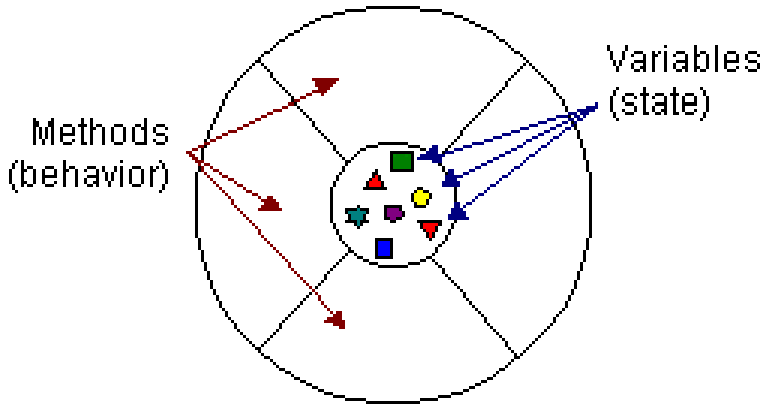
- Exemplos:
  - cães: nome, raça, cor.
    latir, abanar cauda, correr, pular.
  - bicicleta: pedais, duas rodas, marchas, número de marchas.
    acelerar, frear, troca de marchas.



Curso básico de linguagem Java

objetos e classes

Objetos como modelos do mundo real: *instâncias* representadas através de *variáreis*, que definem o estado do objeto. *Métodos*, que definem o comportamento do objeto, e permitem também alterá-lo.
As instâncias estão **encapsuladas** dentro do objeto, circundadas pelos métodos.



Curso básico de linguagem Java

objetos e classes

As instâncias e métodos de um objeto são definidas por meio de *classes*.

```
class Point extends Object {
        public double  x;     /*  instance variable  */
        public double  y;     /*  instance variable  */
    }
```

Aqui entra o conceito de **herança**. A classe Point herda as variáveis e os métodos da classe Object, sendo então uma subclasse desta.
A classe Objeto está na raiz de hierarquia de classes, sendo ela uma superclasse de todas as demais.
As variáveis de instância são declaradas como públicas. Deste modo, seu valores são acessíveis para o objeto.

Curso básico de linguagem Java

objetos e classes

```
class createPoint01 {
    public static void main(String[] args) {
       Point myPoint;
       myPoint = new Point();
       System.out.println("coordenada inicial em x: “
                          + myPoint.x);
       System.out.println("coordenada inicial em y: “
                          + myPoint.y);
 }
}
```

Curso básico de linguagem Java

Construtor: usado para inicializar as variáveis de instância, devendo ter o mesmo nome da classe.

```
class Point extends Object {
        public double  x;    /* variável de instância  */
        public double  y;    /* variável de instância */

         /* construtor para inicializar com valor zero */
        Point() {
            x = 0.0;
            y = 0.0;
        }
        /* construtor para inicializar com um valor especificado */
        Point(double x, double y) {
            this.x = x;
            this.y = y;
        }
    }
```

Curso básico de linguagem Java

objetos e classes

As variáveis de instância são declaradas como públicas. Deste modo, seu valores são acessíveis para o objeto.

```
class createPoint01 {
    public static void main(String[] args) {

        ......

        myPoint.x = 10.0;
        myPoint.y = 25.7;
        System.out.println("\ncoordenada modificada em x: “
                           + myPoint.x);
        System.out.println("coordenada modificada em y: “
                           + myPoint.y);
 }
}
```

Curso básico de linguagem Java

objetos e classes

A variável this refere-se ao objeto criado, this.x e this.y significam a variável de instância x e y, respectivamente, do objeto.

```
class createPoint01 {
    public static void main(String[] args) {

        ......

       Point  lowerLeft;
        Point  upperRight;
        lowerLeft = new Point();
        upperRight = new Point(100.0, 200.0);
        System.out.println("\ncoordenada inferior esquerda ("
                              +lowerLeft.x+","+lowerLeft.y+")");
        System.out.println("coordenada superior direita ("
                             +upperRight.x+","+upperRight.y+")");}
}
```

Curso básico de linguagem Java

Os construtores são em geral opcionais. Há casos, entretanto, onde os construtores são essenciais. Por exemplo, o construtor Rectangle() construtor é necessário para assegurar que os objetos Point são instanciados ao mesmo tempo que o objeto Rectangle o é. Caso assim não fosse, o objeto Rectangle tentaria referenciá-los sem sucesso.

```
 class Rectangle extends Object {
       public Point lowerLeft;
       public Point upperRight;

       Rectangle() {
            lowerLeft = new Point();
            upperRight = new Point();
       }
}
```

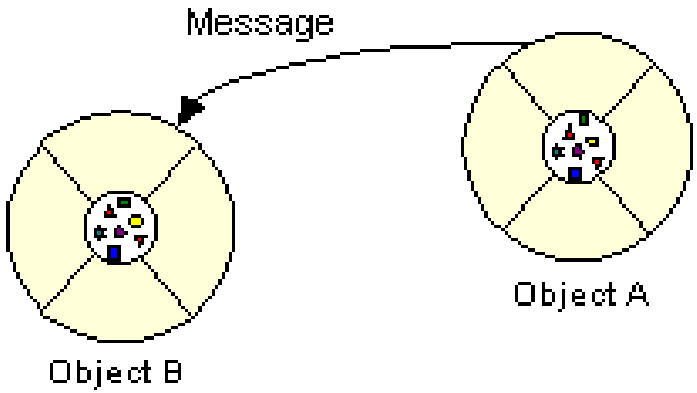
Curso básico de linguagem Java

objetos e classes

```
class createPoint01 {
    public static void main(String[] args) {

       ......

       Rectangle retangulo;
       retangulo = new Rectangle();
       System.out.println("\ncoordenada inferior esquerda ("
                          + retangulo.lowerLeft.x+","
                          + retangulo.lowerLeft.y+")");
       System.out.println("coordenada superior direita ("
                          + retangulo.upperRight.x+","
                          + retangulo.upperRight.y+")");
}
```

Curso básico de linguagem Java

métodos e mensagens



Um objeto isoladamente tem pouca utilidade. O que torna a programação orientada a objetos interessante, é a possibilidade de os objetos em um projeto se comunicarem entre si, acessando seus métodos através da passagem de **mensagem**.

Usando este paradigma de programação orientada a objeto, pode-se construir redes de objetos que passam mensagens entre eles para alterar seu estado. Esta é uma das técnicas que melhor possibilitam representar o sistemas do mundo real.

Curso básico de linguagem Java

métodos e mensagens

Quando as variáveis de instância são do tipo *private*, as mesmas só poderão ser alteradas através de métodos (*accessor methods*).

```
class Point extends Object {
        private double  x;     /* variável de instância */
        private double  y;     /* variável de instância */

         .......

        public void setX(double x) {    /*  accessor method  */
            this.x = x;
        }
        public void setY(double y) {    /*  accessor method  */
            this.y = y;
        }
        public double getX() {    /*  accessor method  */
            return x;
        }
        public double getY() {    /*  accessor method  */
            return y;
        }
    }
```

Curso básico de linguagem Java

métodos e mensagens

```
class createPoint02 {
    public static void main(String[] args) {
       // agora torna as variaveis da classe Point private
       Point myPoint;
       myPoint = new Point();
       // uso de accessor methods atraves
       // da passagem de mensagem ao objeto
       myPoint.setX(10.0); myPoint.setY(25.7);
       System.out.println("coordenada em x: “
                              + myPoint.x);
       System.out.println("coordenada em y: “
                              + myPoint.y);
    }
}
```

Curso básico de linguagem Java

métodos e mensagens

```
class createPoint02 {
    public static void main(String[] args) {
       // agora torna as variaveis da classe Point private
       Point myPoint;
       myPoint = new Point();
       // uso de accessor methods atraves
       // da passagem de mensagem ao objeto
       myPoint.setX(10.0); myPoint.setY(25.7);
       System.out.println("coordenada em x: “
                          + myPoint.getX());
       System.out.println("coordenada em x: “
                          + myPoint.getY());
    }
}
```

Curso básico de linguagem Java

métodos e mensagens

Uma mensagem é constituída de três componentes básicos:
- O objeto a que a mensagem está sendo enviada (`myPoint`);
- O nome do método a ser executado (`setX` e `setY`);
- Os parâmetros do método (`10.0` e `25.7`, respectivamente) ;

```
Point  myPoint;

myPoint = new Point();
myPoint.setX(10.0);
myPoint.setY(25.7);
```

Curso básico de linguagem Java

subclasses

Novos objetos podem ser definidos em termos de objetos já existentes.

```
class Point extends Object {
    protected double  x;    /*  instance variable  */
    protected double  y;    /*  instance variable  */

    Point() {    /*  constructor to initialize to zero  */
        x = 0.0;
        y = 0.0;
    }
}
```

Curso básico de linguagem Java

subclasses

```
class ThreePoint extends Point {
    protected double z;    /*  the z coordinate of the point  */

    ThreePoint() {      /*  default constructor  */
        x = 0.0;        /*  initialize the coordinates  */
        y = 0.0;
        z = 0.0;
    }
    ThreePoint(double x, double y, double z) {/* specific constructor */
        this.x = x;        /*  initialize the coordinates  */
        this.y = y;
        this.z = z;
    }
}
```

Para realizar esta tarefa, é criada uma subclasse da classe que instancia o objeto, que no exemplo apresentado é representada pela subclasse ThreePoint da classe Point. Aqui um novo tipo variável é usado: a variável *protected*. Neste caso, somente a subclasse tem acesso às variáveis de instância.

Nota-se que as variáveis x e y não necessitam ser definidas novamente na subclasse ThreePoint, pois as mesmas são *herdadas* da classe Point.

Curso básico de linguagem Java

subclasses

Subclasses permitem usar códigos existentes que já foram desenvolvido e, principalmente, já testados para casos mais genéricos.

```
    class createPoint03 {
    public static void main(String[] args) {
       ThreePoint myPoint;
       myPoint = new ThreePoint(3.2,1.4,5.6);
       System.out.println("coordenada em x: " + myPoint.x);
       System.out.println("coordenada em y: " + myPoint.y);
       System.out.println("coordenada em y: " + myPoint.z);
    }
}
```

Curso básico de linguagem Java

Controle de acesso

Até aqui foram vistos três níveis de acesso a variáveis:
- *public*: variáveis de instância e métodos são disponíveis a qualquer classe
- *protected*: variáveis de instância e métodos podem ser acessadas somente por subclasses da classe, e por mais nenhuma outra.
- *private*: variáveis de instância e métodos são disponíveis somente dentro da classe em que foram declarados, não podendo ser acessadas nem mesmo por suas subclasses.

Há ainda um quarto nível de acesso:
- *friendly*: indica que variáveis de instância e métodos são disponíveis a todos os objetos de um mesmo pacote (*package*), mas são inacessíveis a objetos fora deste pacote. Quando o nível de acesso não é especificado, têm-se o nível *friendly*.

Curso básico de linguagem Java

Váriáveis e métodos classe

As variáveis de instância normalmente têm uma cópia para cada objeto que é criado a partir de uma classe. Já as variáveis classe possui uma única cópia que é compartilhada por cada objeto criado a partir da classe.

```
class Rectangle extends Object {
        static  final int version = 2;
        static  final int revision = 0;
    }
```

Métodos classe são aqueles que são comuns a toda classe. Só podem ser aplicados em variáveis classe, não tendo acesso a variáveis de instância, nem podem chamar métodos de instância. Tal como variáveis classe são definidos com **`static`**.

Curso básico de linguagem Java

superclasse abstrata & classe concreta

Uma superclasse abstrata é uma classe que declara métodos sem realmente implementá-los.

```
// Definição da superclasse abstrata Shape

public abstract class Shape extends Object {
   public double area() { return 0.0; }
   public double volume() { return 0.0; }
   public abstract String getName();
}
```

Curso básico de linguagem Java

superclasse abstrata & classe concreta

```
public class Point extends Shape {

        .......

// convert the point into a String representation
   public String toString() {
          return "[" + x + ", " + y + "]";
   }
   // return the class name
   public String getName() { return "Point"; }
}
```

Curso básico de linguagem Java

superclasse abstrata & classe concreta

```
public class Circle extends Point {  // inherits from Point
   protected double radius;
   public Circle()
   {
      // implicit call to superclass constructor here
      setRadius( 0 );
   }
   // Constructor
   public Circle( double r, int a, int b )
   {
      super( a, b );  // call the superclass constructor
      setRadius( r );
   }
   // Set radius of Circle
   public void setRadius( double r )
      { radius = ( r >= 0 ? r : 0 ); }
   // Get radius of Circle
   public double getRadius() {
              return radius;
   }
   // Calculate area of Circle
   public double area() { return Math.PI * radius * radius; }
   // convert the Circle to a String
   public String toString()
      { return "Center = " + super.toString() +
                "; Radius = " + radius; }
   // return the class name
   public String getName() { return "Circle"; }
```

Curso básico de linguagem Java

superclasse abstrata & classe concreta

```
public class Cylinder extends Circle {
   protected double height;  // height of Cylinder
   public Cylinder()
   {
      // implicit call to superclass constructor here
      setHeight( 0 );
   }
   public Cylinder( double h, double r, int a, int b )
   {
      super( r, a, b );   // call superclass constructor
      setHeight( h );
   }
   // Set height of Cylinder
   public void setHeight( double h )
      { height = ( h >= 0 ? h : 0 ); }
   // Get height of Cylinder
   public double getHeight() { return height; }
   // Calculate area of Cylinder (i.e., surface area)
   public double area()
   {
      return 2 * super.area() +
             2 * Math.PI * radius * height;
   }
   // Calculate volume of Cylinder
   public double volume() { return super.area() * height; }
   // Convert a Cylinder to a String
   public String toString()
      { return super.toString() + "; Height = " + height; }
   // Return the class name
   public String getName() { return "Cylinder"; }
}
```

Curso básico de linguagem Java

superclasse abstrata & classe concreta

```
public class Test {
   public static void main( String args[] )
   {
      Point point = new Point( 7, 11 );
      Circle circle = new Circle( 3.5, 22, 8 );
      Cylinder cylinder = new Cylinder( 10, 3.3, 10, 10 );

      Shape arrayOfShapes[];

      arrayOfShapes = new Shape[ 3 ];

      // aim arrayOfShapes[0] at subclass Point object
      arrayOfShapes[ 0 ] = point;

      // aim arrayOfShapes[1] at subclass Circle object
      arrayOfShapes[ 1 ] = circle;

      // aim arrayOfShapes[2] at subclass Cylinder object
      arrayOfShapes[ 2 ] = cylinder;

                     ..........

   }
}
```

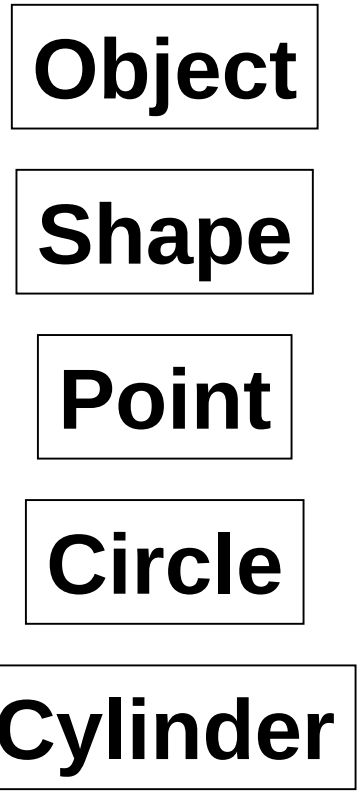
Curso básico de linguagem Java

superclasse abstrata & classe concreta

```
public class Test {
   public static void main( String args[] )
   {

                     ..........

      String output =
         point.getName() + ": " + point.toString() + "\n" +
         circle.getName() + ": " + circle.toString() + "\n" +
         cylinder.getName() + ": " + cylinder.toString();

      // Loop through arrayOfShapes and print the name,
      // area, and volume of each object.
      for ( int i = 0; i < arrayOfShapes.length; i++ ) {
         output += "\n\n" +
            arrayOfShapes[ i ].getName() + ": " +
            arrayOfShapes[ i ].toString() +
            "\nArea = " + arrayOfShapes[ i ].area() +
            "\nVolume = " + arrayOfShapes[ i ].volume();
      }

              System.out.println(output);
   }
}
```

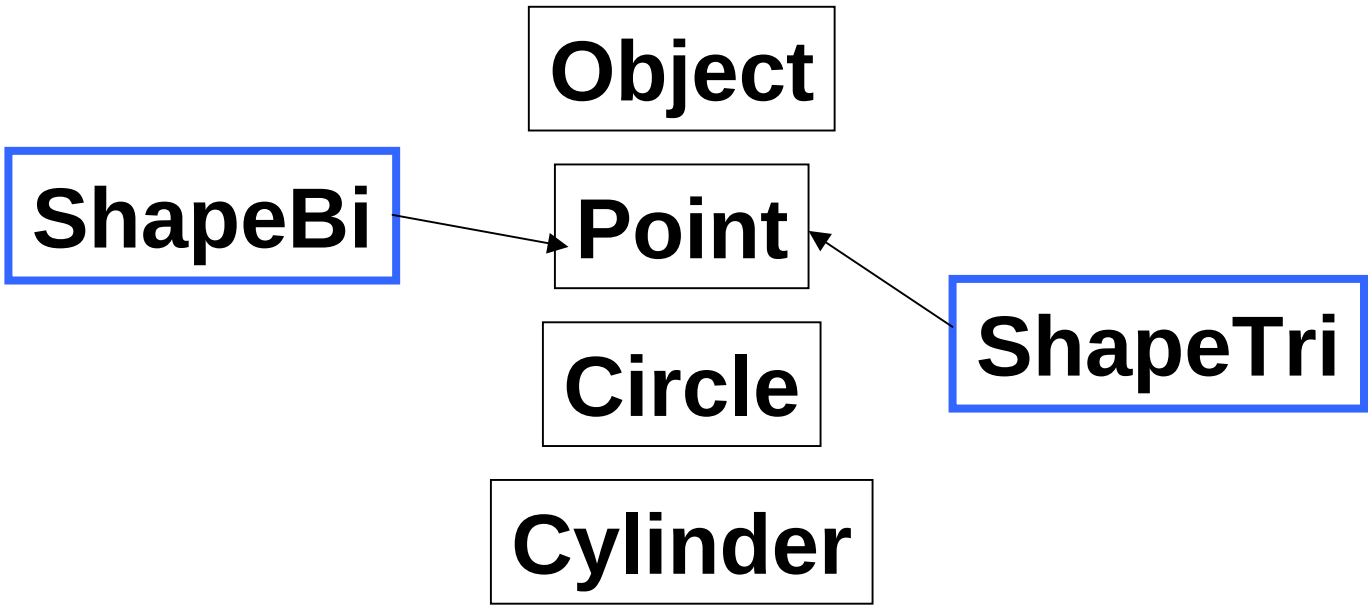
Curso básico de linguagem Java

superclasse abstrata & classe concreta



Curso básico de linguagem Java

interface




```
public interface ShapeBi {
   public abstract double area();
   public abstract String getName();
}
public interface ShapeTri {
   public abstract double volume();
}
```

Curso básico de linguagem Java

interface

```
public class Point extends Object implements ShapeBi, ShapeTri {

      .......

   // return the area
   public double area() { return 0.0; }
   // return the volume
   public double volume() { return 0.0; }
   // return the class name
   public String getName() { return "Point"; }
}
```

Curso básico de linguagem Java

interface

```
public class Test {
   public static void main( String args[] )
   {
                  ........
         output += "\n\n" +
            point.getName() + ": " +
            point.toString() +
            "\nArea = " + point.area() +
            "\nVolume = " + point.volume();
         output += "\n\n" +
            circle.getName() + ": " +
            circle.toString() +
            "\nArea = " + circle.area() +
            "\nVolume = " + circle.volume();
         output += "\n\n" +
            cylinder.getName() + ": " +
            cylinder.toString() +
            "\nArea = " + cylinder.area() +
            "\nVolume = " + cylinder.volume();
           System.out.println(output);
   }
}
```

Curso básico de linguagem Java

pacotes

```
package formas;
public interface ShapeBi {
   public abstract double area();
   public abstract String getName();
}

package formas;
public interface ShapeTri {
   public abstract double volume();
}
```

```
Nome da interface:  formas.ShapeBi
                    formas.ShapeTri
Caminho do arquivo: formas/shapebi.java
                    formas/shapetri.java
```

Curso básico de linguagem Java

pacotes

```
package formas;
public class Point extends Object implements ShapeBi, ShapeTri {
        .......
   // return the class name
   public String getName() { return "Point"; }
}

package formas;
public class Circle extends Point {
        .......
   // return the class name
   public String getName() { return "Circle"; }
}

package formas;
public class Cylinder extends Circle {
        .......
   // return the class name
   public String getName() { return "Cylinder"; }
}
```

Curso básico de linguagem Java

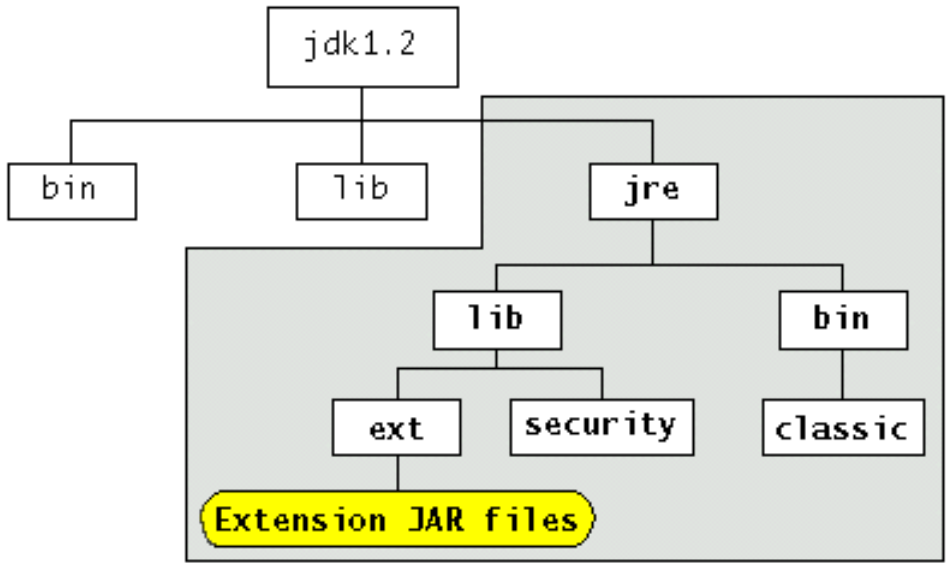
pacotes

```
import formas.*;
public class Test {
   public static void main( String args[] )
   {
                .........
   }
}
```

Curso básico de linguagem Java

pacotes

```
jar cvf formas.jar formas/

java –classpath [diretório] Test

Ou copia arquivo .jar para o
diretório lib/ext:
java Test
```



Curso básico de linguagem Java

pacotes

```
import java.text.DecimalFormat;
import formas.*;
public class Test {
   public static void main( String args[] )
   {
                .........
   }
}
```

Curso básico de linguagem Java

pacotes

```
import javax.swing.JOptionPane;
import java.text.DecimalFormat;
import formas.*;
public class Test {
   public static void main( String args[] )
   {
                .........
   }
}
```

Curso básico de linguagem Java

pacote gráfico

```
Abstract Window Toolkit (AWT) componentes
Pacote java.awt
Exemplo: Button

Swing (nome provisório)
Pacote javax.swing
Exemplo: JButton
```

Curso básico de linguagem Java

pacote gráfico

```
import javax.swing.*;

public class HelloWorldSwing {
    public static void main(String[] args) {
        JFrame frame = new JFrame("HelloWorldSwing");
        final JLabel label = new JLabel("Hello World");
        frame.getContentPane().add(label);

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }
}
```

Curso básico de linguagem Java

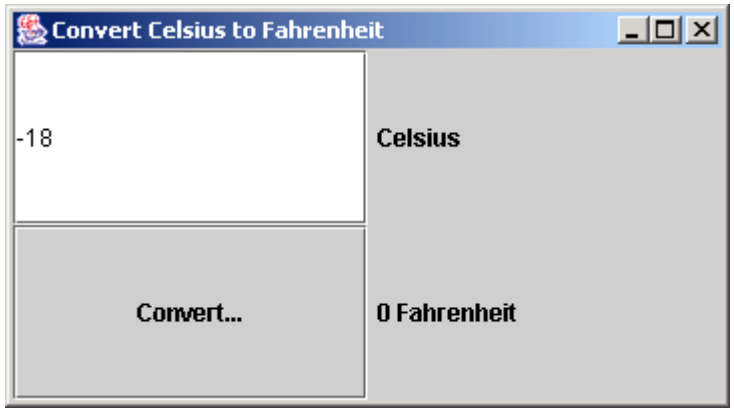
pacote gráfico

Curso básico de linguagem Java

pacote gráfico

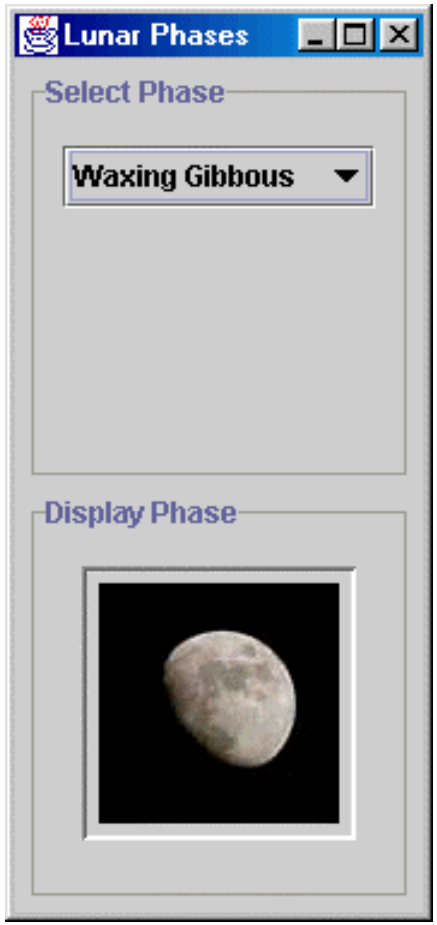
Curso básico de linguagem Java

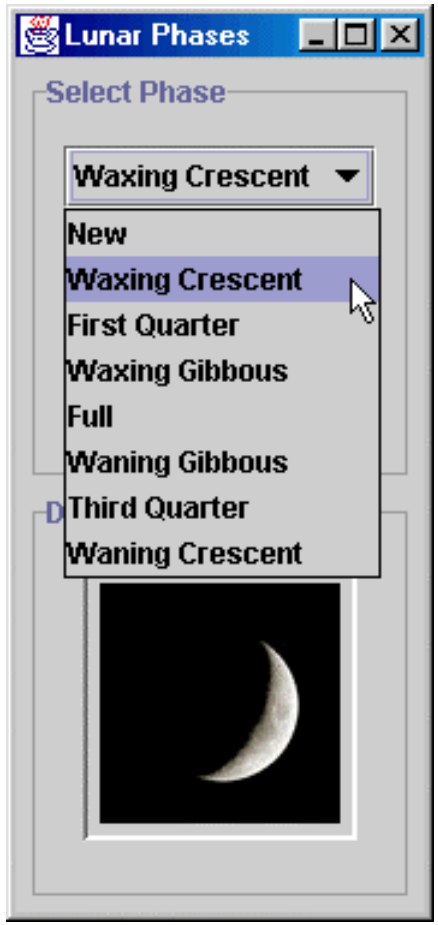
pacote gráfico



Curso básico de linguagem Java

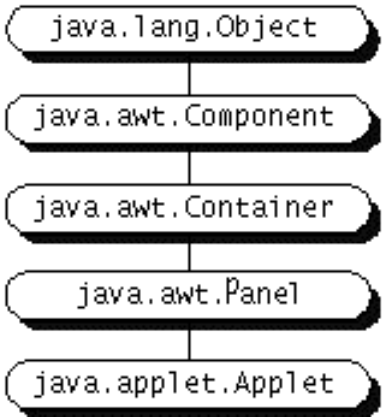
pacote gráfico





Curso básico de linguagem Java

applet




```
import java.applet.Applet;
import java.awt.Graphics;

public class HelloWorld extends Applet {
    public void paint(Graphics g) {
        g.drawString("Hello world!", 50, 25);
    }
}
```

Curso básico de linguagem Java

applet

```
<HTML>
<HEAD>
<TITLE> A Simple Program </TITLE>
</HEAD>
<BODY>
Here is the output of my program:
<APPLET CODE="HelloWorld.class" WIDTH=150 HEIGHT=25>
</APPLET>
</BODY>
</HTML>
```

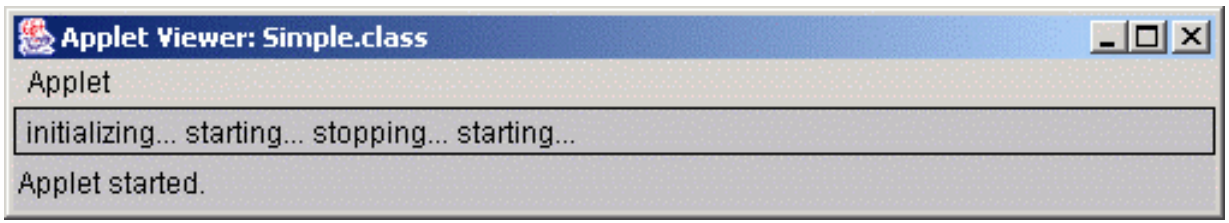
O comando appletviewer permite rodar applets fora de um browser.

```
appletviewer HelloWorld.html
```

Curso básico de linguagem Java

applet

```
<html>
<applet code="Simple.class" width=500 height=20>
</applet>
</html>
```

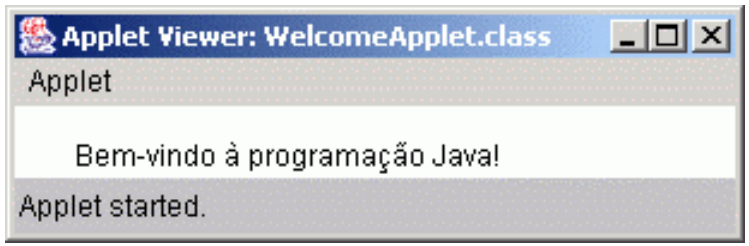


Curso básico de linguagem Java

applet

```
import javax.swing.JApplet;  // import class JApplet
import java.awt.Graphics;    // import class Graphics

public class WelcomeApplet extends JApplet {
   public void paint( Graphics g )
   {
      g.drawString( "Bem-vindo à programação Java!", 25, 25 );
   }
}
```

```
<html>
<applet code="WelcomeApplet.class" width=300 height=30>
</applet>
</html>
```



Curso básico de linguagem Java

applet

```
import javax.swing.JApplet;  // import class JApplet
import java.awt.Graphics;    // import class Graphics

public class WelcomeApplet2 extends JApplet {
   public void paint( Graphics g )
   {
      g.drawString( "Bem-vindo à", 25, 25 );
      g.drawString( "programação Java!", 25, 40 );
   }
}
```

```
<html>
<applet code="WelcomeApplet2.class" width=300 height=45>
</applet>
</html>
```

Curso básico de linguagem Java

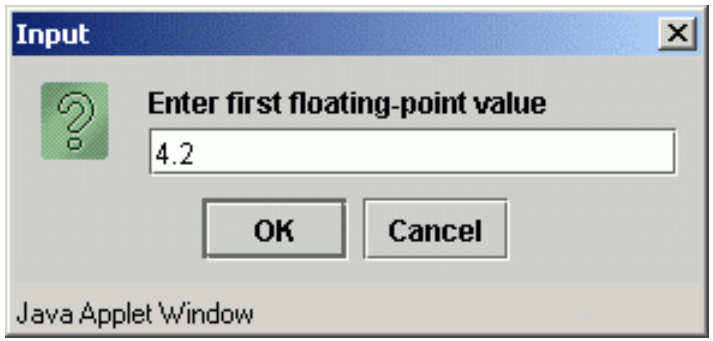
applet

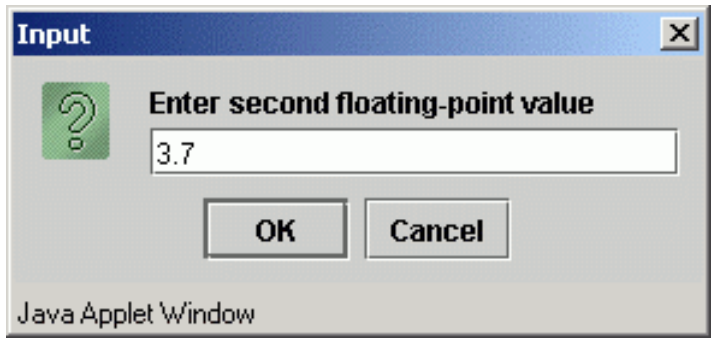
```
import javax.swing.JApplet;  // import class JApplet
import java.awt.Graphics;    // import class Graphics
public class WelcomeLines extends JApplet {
   public void paint( Graphics g )
   {
      g.drawLine( 15, 10, 210, 10 );
      g.drawLine( 15, 30, 210, 30 );
      g.drawString( "Bem-vindo à programação Java!", 25, 25 );
   }
}
```

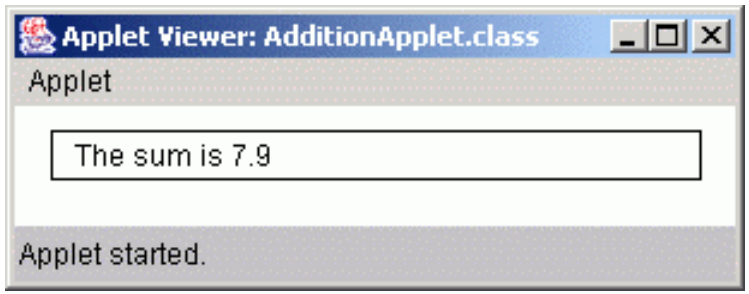
```
<html>
<applet code="WelcomeLines.class" width=300 height=40>
</applet>
</html>
```

Curso básico de linguagem Java

applet







Curso básico de linguagem Java

applet

```
<html>
<APPLET CODE = "TumbleItem.class" WIDTH = "600" HEIGHT = "95" >
<PARAM NAME = "maxwidth" VALUE ="120">
<PARAM NAME = "nimgs" VALUE ="17">
<PARAM NAME = "offset" VALUE ="-57">
<PARAM NAME = "img" VALUE ="images/tumble">
</APPLET>
</html>
```