Programação Orientada a Objetos

Dada a interface Shape e as classes Point, Circle, Cilinder e Test:

Interface Shape:

Object é uma classe do pacote *java.lang* do Java.

```java
4  public interface Shape {
5
6      // calculate area
7      public abstract double area();
8
9      // calculate volume
10     public abstract double volume();
11
12     // return shape name
13     public abstract String getName();
14  }
15
```

Programação Orientada a Objetos

Classe Point:

```java
 4 ▼ public class Point extends Object implements Shape {
 5       protected int x, y; // coordinates of the Point
 6
 7       // no-argument constructor
 8       public Point()
 9 ▼     {
10           setPoint( 0, 0 );
11       }
12
13       // constructor
14       public Point( int xCoordinate, int yCoordinate )
15 ▼     {
16           setPoint( xCoordinate, yCoordinate );
17       }
18
19       // Set x and y coordinates of Point
20       public void setPoint( int xCoordinate, int yCoordinate )
21 ▼     {
22           x = xCoordinate;
23           y = yCoordinate;
24       }
25
26       // get x coordinate
27       public int getX()
28 ▼     {
29           return x;
30       }
31
32       // get y coordinate
33       public int getY()
34 ▼     {
35           return y;
36       }
37
38       // convert point into String representation
39       public String toString()
40 ▼     {
41           return "[" + x + ", " + y + "]";
42       }
43
44       // calculate area
45       public double area()
46 ▼     {
47           return 0.0;
48       }
49
50       // calculate volume
51       public double volume()
52 ▼     {
53           return 0.0;
54       }
55
56       // return shape name
57       public String getName()
58 ▼     {
59           return "Point";
60       }
61
62   } // end class Point
63
```

profº Mauricio Conceição Mario
referências:H.M. Deitel e P.J. Deitel- Java Como Programar 4ª edição – editora Bookman -2003

Programação Orientada a Objetos

Classe Circle:

```java
4 ▼ public class Circle extends Point {  // inherits from Point
5       protected double radius;
6
7       // no-argument constructor
8       public Circle()
9 ▼     {
10          // implicit call to superclass constructor here
11          setRadius( 0 );
12      }
13
14      // constructor
15      public Circle( double circleRadius, int xCoordinate,
16          int yCoordinate )
17 ▼    {
18          // call superclass constructor
19          super( xCoordinate, yCoordinate );
20
21          setRadius( circleRadius );
22      }
23
24      // set radius of Circle
25      public void setRadius( double circleRadius )
26 ▼    {
27          radius = ( circleRadius >= 0 ? circleRadius : 0 );
28      }
29
30      // get radius of Circle
31      public double getRadius()
32 ▼    {
33          return radius;
34      }
35
36      // calculate area of Circle
37      public double area()
38 ▼    {
39          return Math.PI * radius * radius;
40      }
41
```

```java
41
42      // convert Circle to a String represention
43      public String toString()
44 ▼    {
45          return "Center = " + super.toString() +
46              "; Radius = " + radius;
47      }
48
49      // return shape name
50      public String getName()
51 ▼    {
52          return "Circle";
53      }
54
55  } // end class Circle
56
```

Programação Orientada a Objetos

Classe Cilinder:

```java
 4   public class Cylinder extends Circle {
 5       protected double height;  // height of Cylinder
 6
 7       // no-argument constructor
 8       public Cylinder()
 9       {
10           // implicit call to superclass constructor here
11           setHeight( 0 );
12       }
13
14       // constructor
15       public Cylinder( double cylinderHeight,
16           double cylinderRadius, int xCoordinate,
17           int yCoordinate )
18       {
19           // call superclass constructor
20           super( cylinderRadius, xCoordinate, yCoordinate );
21
22           setHeight( cylinderHeight );
23       }
24
25       // set height of Cylinder
26       public void setHeight( double cylinderHeight )
27       {
28           height = ( cylinderHeight >= 0 ? cylinderHeight : 0 );
29       }
30
31       // get height of Cylinder
32       public double getHeight()
33       {
34           return height;
35       }
36
37       // calculate area of Cylinder (i.e., surface area)
38       public double area()
39       {
40           return 2 * super.area() + 2 * Math.PI * radius * height;
41       }
```

```java
42
43       // calculate volume of Cylinder
44       public double volume()
45       {
46           return super.area() * height;
47       }
48
49       // convert Cylinder to a String representation
50       public String toString()
51       {
52           return super.toString() + "; Height = " + height;
53       }
54
55       // return shape name
56       public String getName()
57       {
58           return "Cylinder";
59       }
60
61   }  // end class Cylinder
62
```

Programação Orientada a Objetos

Classe Test:

```java
4   // Java core packages
5   import java.text.DecimalFormat;
6
7   // Java extension packages
8   import javax.swing.JOptionPane;
9
10  public class Test {
11
12      // test Shape hierarchy
13      public static void main( String args[] )
14      {
15          // create shapes
16          Point point = new Point( 7, 11 );
17          Circle circle = new Circle( 3.5, 22, 8 );
18          Cylinder cylinder = new Cylinder( 10, 3.3, 10, 10 );
19
20          // create Shape array
21          Shape arrayOfShapes[] = new Shape[ 3 ];
22
23          // aim arrayOfShapes[ 0 ] at subclass Point object
24          arrayOfShapes[ 0 ] = point;
25
26          // aim arrayOfShapes[ 1 ] at subclass Circle object
27          arrayOfShapes[ 1 ] = circle;
28
29          // aim arrayOfShapes[ 2 ] at subclass Cylinder object
30          arrayOfShapes[ 2 ] = cylinder;
31
32          // get name and String representation of each shape
33          String output =
34              point.getName() + ": " + point.toString() + "\n" +
35              circle.getName() + ": " + circle.toString() + "\n" +
36              cylinder.getName() + ": " + cylinder.toString();
37
38          DecimalFormat precision2 = new DecimalFormat( "0.00" );
39
40          // loop through arrayOfShapes and get name,
41          // area and volume of each shape in arrayOfShapes
42          for ( int i = 0; i < arrayOfShapes.length; i++ ) {
43              output += "\n\n" + arrayOfShapes[ i ].getName() +
44                  ": " + arrayOfShapes[ i ].toString() +
45                  "\nArea = " +
46                  precision2.format( arrayOfShapes[ i ].area() ) +
47                  "\nVolume = " +
48                  precision2.format( arrayOfShapes[ i ].volume() );
49          }
50
51          JOptionPane.showMessageDialog( null, output,
52              "Demonstrating Polymorphism",
53              JOptionPane.INFORMATION_MESSAGE );
54
55          System.exit( 0 );
56      }
57
58  } // end class Test
59
60
```

profº Mauricio Conceição Mario
referências:H.M. Deitel e P.J. Deitel- Java Como Programar 4ª edição – editora Bookman -2003

Programação Orientada a Objetos

Exercício 10: Fazer o diagrama de classes das classes que compõem a aplicação.

Exercício 11: Quais métodos da classe Shape devem obrigatoriamente ser implementados pela classe Point.

Exercício 12: Há métodos da classe Shape que devem obrigatoriamente ser implementados pelas classes Circle e Cilinder? Justificar.

Exercício 13: Que métodos da classe Point a classe Circle herda?

Exercício 14: Que métodos da classe Circle a classe Cilinder herda?

Exercício 15: Qual o resultado da execução da classe executável?