Projeto 13 – Herança e Polimorfismo IV (03/11/2014)

1.

```java
/**
 * Interface of all the shapes that can be calculate the area
 * @author FilipaG
 */
public interface IArea {

	/**
	 * Calculate the area of the shape
	 * @return
	 */
	public double calculateArea();
}
```

```java
/**
 * Interface of all the shapes that can be calculate the volume
 * @author FilipaG
 */
public interface IVolume {

	/**
	 * Calculate the volume of the shape
	 * @return
	 */
	public double calculateVolume();
}
```

```java
/**
 * Interface of all the shapes that can be printed.
 * @author FilipaG
 */
public interface IPrintableShape {

	/**
	 * print shapes's information
	 */
	public void printShape();
}
```

```java
import java.text.DecimalFormat;

public abstract class GeometricShape {
	protected String name;
	DecimalFormat decimal = new DecimalFormat("#.00");
				// para que os resultados apareçam com duas casas decimais

	/**
	 * Constructor have the following parameters
	 * @param name (name of the shape)
	 */
	public GeometricShape(String name) {
		this.name = name;
	}

	/**
	 * List all the printable Shapes on the console
	 */
	public void listShapes() {		//lista todas as formas geométricas
	}

	public void printShape() {		// imprime a informação da forma geométrica
	}
}
```

```
/**
 * Implements a container of shapes (2D and 3D Shape).
 * @author FilipaG
 */
public class GeometricShapeContainer extends GeometricShape{
	private GeometricShape[] shapes;
	private int numShapes;

	/**
	 * Constructor have the following parameters
	 * @param name  (name of the shape)
	 * @param maxShape  (maximum number of shapes that can be insert in the container)
	 */
	public GeometricShapeContainer(String name, int maxShape) {
		super(name);
		if (maxShape>0)              // a dimensão do contentor tem de ser positiva
			shapes = new GeometricShape[maxShape];
	}

	/**
	 * Adds a geometric shape to the container, until is full.
	 * @param shape
	 */
	public void add(GeometricShape shape){
		if(numShapes < shapes.length) {
		//só adiciona as formas geométricas caso o contentor ainda não estiver cheio
			shapes[numShapes] = shape;
			numShapes++;
		}
		else
			System.out.println("This container is full.");
	}

	/**
	 * List all the printable Shapes on the console
	 */
	public void listShapes(){
		System.out.println("Shapes");
		for(int idx = 0; idx < numShapes; idx++)
			shapes[idx].printShape();
				// imprime todas as formas geométricas armazenadas em Shape
	}

	/**
	 * List only the 2D Shapes on the console
	 */
	public void list2DShapes(){
		System.out.println("2D Shapes");
		for(int i = 0; i < numShapes; i++)
			if(shapes[i] instanceof TwoDimensionalShape)
				shapes[i].printShape();
					// imprime as formas geométricas 2D armazenadas em Shape

	}

	/**
	 * List only the 3D Shapes on the console
	 */
	public void list3DShapes(){
		System.out.println("3D Shapes");
		for(int i = 0; i < numShapes; i++)
			if(shapes[i] instanceof ThreeDimensionalShape)
				shapes[i].printShape();    // imprime as formas geométricas 3D
armazenadas em Shape
	}
}
```

```
public class TwoDimensionalShape extends GeometricShape implements IArea{

	/**
	 *  * Constructor have the following parameters
	 * @param name  (name of the shape)
	 */
		public TwoDimensionalShape(String name){
			super(name);
		}

		/**
		 * Calculate the area oh the shape
		 */
		@Override
		public double calculateArea() {
			return 0;
		}
}
```

```
public class Circle extends TwoDimensionalShape implements IArea, IPrintableShape{
	private int radius;

	/**
	 * The constructor has the following parameters
	 * @param name name of the shape
	 * @param rad  radius of the circle
	 */
	public Circle(String name, int rad) {
		super(name);
		this.radius = rad;
	}

	/**
	 * print the area of the circle
	 */
	@Override
	public void printShape() {
		System.out.println("The area of " + name + " (2D Shape) is: " +
decimal.format(calculateArea()));
	}

	/**
	 * Calculate the area of the circle
	 */
	@Override
	public double calculateArea() {
		return Math.PI*Math.pow(radius, 2);
	}
```

```
public class Rectangle extends TwoDimensionalShape implements IArea, IPrintableShape{

	private int width;    //largura do retângulo
	private int height;    // altura do rectângulo

	/**
	 * The constructor has the following parameters
	 * @param name  name of the shape
	 * @param w  width of the rectangle
	 * @param h  height of the rectangle
	 */
	public Rectangle(String name, int w, int h) {
		super(name);
		this.width = w;
		this.height = h;
	}
```

```
	/**
	 * print the area of the rectangle
	 */
	@Override
	public void printShape() {
		System.out.println("The volume of " + name + " (2D Shape) is: " +
calculateArea());
	}

	/**
	 * Calculate the area of the rectangle
	 */
	@Override
	public double calculateArea() {
		return width * height;
	}
}
```

```
public class Square extends TwoDimensionalShape implements IArea, IPrintableShape{

	private int side;	// lado do quadrado

	/**
	 * The constructor has the following parameters
	 * @param name  name of the shape
	 * @param side
	 */
	public Square(String name, int side) {
		super(name);
		this.side = side;
	}

	/**
	 * print the area of the square
	 */
	@Override
	public void printShape() {
		System.out.println("The volume of " + name + " (2D Shape) is: " +
calculateArea());

	}

	/**
	 * Calculate the area of the square
	 */
	@Override
	public double calculateArea() {
		return Math.pow(side, 2);
	}
}
```

```
public class Triangle extends TwoDimensionalShape implements IArea {
	private int base;	// base do triângulo
	private int height;	// altura do triângulo

	/**
	 * The constructor has the following parameters
	 * @param name  name of the shape
	 * @param b base of the triangle
	 * @param h height of the triangle
	 */
	public Triangle(String name, int b, int h) {
		super(name);
		this.base = b;
		this.height = h;
	}
```

```
	/**
	 * Calculate the area of the triangle
	 */
	@Override
	public double calculateArea() {
		return (height * base)/2;
	}
}
```

```
public class ThreeDimensionalShape extends GeometricShape implements IVolume{

	/**
	 *  * Constructor have the following parameters
	 * @param name  (name of the shape)
	 */
	public ThreeDimensionalShape(String name){
		super (name);
	}

	/**
	 * Calculate the area oh the shape
	 */
	@Override
	public double calculateVolume() {
		return 0;
	}
}
```

```
public class Cone extends ThreeDimensionalShape implements IVolume, IPrintableShape{

	private int radius;   //raio da base
	private int height;   //altura do cone

	/**
	 * The constructor has the following parameters
	 * @param name  name of the 3D shape
	 * @param r  radius of the base
	 * @param h  height oh the cone
	 */
	public Cone(String name, int r, int h) {
		super(name);
		this.radius = r;
		this.height = h;
	}

	/**
	 * print the volume of the cone
	 */
	@Override
	public void printShape() {
		System.out.println("The Volume of " + name + " (3D Shape) is: " +
decimal.format(calculateVolume()));
	}

	/**
	 * Calculate the volume of the cone
	 */

	@Override
	public double calculateVolume() {
		double baseArea = Math.PI*Math.pow(radius, 2);
		return (height * baseArea)/3;
	}
}
```

```
public class RectangularPrism extends ThreeDimensionalShape implements IVolume,
IPrintableShape{

	private int width;   //largura da base
	private int height;  // altura
	private int length;  // comprimento da base

	/**
	 * The constructor has the following parameters
	 * @param name name of the shape
	 * @param w  width of the RectangularPrism
	 * @param h  height of the RectangularPrism
	 * @param l  length of the RectangularPrism
	 */
	public RectangularPrism(String name, int w, int h, int l) {
		super(name);
		this.height = h;
		this.length = l;
		this.width = w;
	}

	/**
	 * print the volume of the RectangularPrism
	 */
	@Override
	public void printShape() {
		System.out.println("The volume of " + name + " (3D Shape) is: " +
calculateVolume());

	}

	/**
	 * Calculate the volume of the RectangularPrism
	 */
	@Override
	public double calculateVolume() {
		double baseArea = width * length;
		return baseArea * height;
	}
}
```

```
public class Sphere extends ThreeDimensionalShape implements IVolume {

	private int radius;  // raio da espera

	/**
	 * The constructor has the following parameters
	 * @param name  name of the shape
	 * @param r  sphere's radius
	 */
	public Sphere(String name, int r) {
		super(name);
		this.radius = r;
	}

	/**
	 * Calculate the volume of the sphere
	 */
	@Override
	public double calculateVolume() {
		return (4/3)*Math.PI*Math.pow(radius, 3);
	}
}
```

```
public class App {

	public static void main (String[] args){

	GeometricShape square = new Square("Square", 3);
	GeometricShape cube = new RectangularPrism("Cube", 5, 5, 5);
	GeometricShape rectangularPrism = new RectangularPrism("NotACube",3,4,5);
	GeometricShape circle = new Circle("Circle", 4);
	GeometricShape cone = new Cone("Cone", 4, 6);
	GeometricShape sphere = new Sphere("Sphere", 6);
	GeometricShape triangle = new Triangle("Triangle", 3, 6);
	GeometricShape rectangle = new Rectangle("Rectangle", 4, 5);
	GeometricShapeContainer tds = new GeometricShapeContainer("Shapes", 8);

	tds.add(square);
	tds.add(cube);
	tds.add(rectangularPrism);
	tds.add(circle);
	tds.add(cone);
	tds.add(sphere);
	tds.add(triangle);
	tds.add(rectangle);

	tds.listShapes();
	System.out.println("\n");

	tds.list2DShapes();
	System.out.println("\n");

	tds.list3DShapes();
	}
}
```

Output:

```
Shapes
The volume of Square (2D Shape) is: 9.0
The volume of Cube (3D Shape) is: 125.0
The volume of NotACube (3D Shape) is: 60.0
The area of Circle (2D Shape) is: 50,27
The Volume of Cone (3D Shape) is: 100,53
The volume of Rectangle (2D Shape) is: 20.0


2D Shapes
The volume of Square (2D Shape) is: 9.0
The area of Circle (2D Shape) is: 50,27
The volume of Rectangle (2D Shape) is: 20.0


3D Shapes
The volume of Cube (3D Shape) is: 125.0
The volume of NotACube (3D Shape) is: 60.0
The Volume of Cone (3D Shape) is: 100,53
```