Atividade 02 - Programação Modular

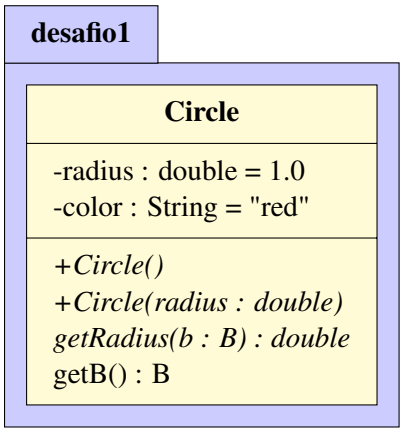
Francisco de Paula Dias Neto

10 de outubro de 2018

1 Desafio 1

O Desafio 1 coloca o código da classe Circle, bem como um Main para execução e teste. Há pequenos erros na implementação do código da classe Circle, como a não inicialização do raio diferente de 1 no construtor em que é passado o raio. Depois que a classe foi consertada os resultados dados foram corretos com as entradas.

1.1 Diagrama de classes



1.2 Códigos

1.2.1 TestCircle

```
package desafio1;

public class Main {

   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
      // Declara c1 como variável habilitada a armazenar uma
         referencia para objeto da classe Circle.
      Circle c1;
      // atribui a c1 .a referencia retornada pelo construtor padrão
         Circle ()
```

```
        c1 = new Circle();
        // Para invocar os metodos classe Circle para operar sobre a
           instância c1,
        // usa-se o operador ponto (".").
        //Em outras palavras± usa-se o ponto para enviar uma mensagem
           ao objeto c1 para que
        // ele execute um de seus métodos.
        System.out.println("O circulo tem o raio de "+ c1.getRadius()
           + "e area de "+ c1.getArea());
        // Declara e aloca uma segunda instancia da classe Circle
           chamada c2
        // com o valor do radius igual a 2.0 e color com valor padrão.
        Circle c2 = new Circle(2.0);
        // Para invocar os metodos a operar sobre a instância c2,
           usa-se o operador ponto (".")
        System.out.println("O circulo tem raio de "+ c2.getRadius() + "
           e area de "+ c2.getArea());
    }

}
```

1.2.2 Circle

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package desafio1;

/**
 *
 * @author Francisco de Paula Dias Neto
 */
public class Circle {
    // variáveis de instancia privadas, isto é, não acessíveis de
       fora desta classe.
    private double radius;
    private String color;
    private double perimeter;
    // primeiro construtor o qual atribui valores iniciais a ambos±
       radius e color.
    public Circle() {
        this(1.0);
    }
    // Segundo construtor que inicia radius com o parâmetro recebido,
       e matem color com
    //o valor padrão definido.
    public Circle(double r) {
        radius = r;
```

```
    }
    // Metodo de acesso para obter o valor armazenado em radius
    public double getRadius() {
        return radius;
    }
    // Metodo de acesso para computar a área de um circulo.
    public double getArea() {
        return radius*radius*Math.PI;
        // PI éa constante. Math éa classe onde PI édefinido.
    }
}
```

2 Desafio 2

4- É lançado uma exceção de compilação:

```
Exception in thread "main" java.lang.Error± Unresolved compilation
    problem±
   The field Circle.radius is not visible
   at desafio1.Main.main(Main.java±23)
```

2.1 Código

2.1.1 Circle

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package desafio1;

/**
 * Implementa o Circulo
 * @author Francisco de Paula Dias Neto
 */
public class Circle {
    // variáveis de instancia privadas, isto é, não acessíveis de
        fora desta classe.
    private double radius;
    private String color;
    private double perimeter;
    // primeiro construtor o qual atribui valores iniciais a ambos±
        radius e color.
    public Circle() {
        this(1.0);
    }
```

```
    // Segundo construtor que inicia radius com o parâmetro recebido,
       e mantem a cor com
    // o valor padrão definido.
    public Circle(double radius) {
       this(radius, "red");
    }

    public Circle(double radius, String color) {
       this.radius = radius;
       this.color = color;
       this.setPerimeter();
    }

    public void setColor(String color) {
       this.color = color;
    }

    public void setRadius(double radius) {
       this.radius = radius;
    }

    private void setPerimeter() {
       this.perimeter = 2*radius*Math.PI;
    }

    public double getPerimeter() {
       return this.perimeter;
    }

    @Override
    public String toString() {
       return "Circulo± raio = "+ radius + "cor = "+ color;
    }

    public String getColor() {
       return color;
    }
    // Metodo de acesso para obter o valor armazenado em radius
    public double getRadius() {
       return radius;
    }
    // Metodo de acesso para computar a área de um circulo.
    public double getArea() {
       return radius*radius*Math.PI;
       // PI éa constante. Math éa classe onde PI édefinido.
    }
}

\subsubsection{TestCircle}
\begin{lstlisting}
/*
```

```
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package desafio1;

import java.lang.Exception;

/**
 *
 * @author Francisco de Paula Dias Neto
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // Declara c1 como variável habilitada a armazenar uma
           referencia para objeto da classe Circle.
        Circle c1;
        // atribui a c1 .a referencia retornada pelo construtor padrão
           Circle ()
        c1 = new Circle();
        System.out.println(c1.toString()); // chamada explicita
        // Para invocar os metodos classe Circle para operar sobre a
           instância c1,
        // usa-se o operador ponto (".").
        //Em outras palavras± usa-se o ponto para enviar uma mensagem
           ao objeto c1 para que
        // ele execute um de seus métodos.
        System.out.println("O circulo tem o raio de "+ c1.getRadius()
           + "e area de "+ c1.getArea());
        // Declara e aloca uma segunda instancia da classe Circle
           chamada c2
        // com o valor do radius igual a 2.0 e color com valor padrão.
        Circle c2 = new Circle(2.0);
        System.out.println(c2.toString()); // chamada explicita
        System.out.println(c2); // println() chama toString()
           implicitamente
        // Para invocar os metodos a operar sobre a instância c2,
           usa-se o operador ponto (".")
        // no exemplo abaixo o operador '+' invoca c2.toString()
           implicitamente.
        System.out.println("Aqui o operador '+' tambem invoca
           toString()± "+ c2);
        System.out.println("O circulo tem raio de "+ c2.getRadius() + "
           e area de "+ c2.getArea());

        Circle c3 = new Circle(); // constroi uma instancia de Circle
        c3.setRadius(5.0); // altera radius
```

```
        c3.setColor("Red"); // altera color

    }

}
```

3 Desafio 3

3.1 Point

```
package desafio1;

/**
 * @author Francisco de Paula Dias Neto
 */
public class Point {
    private float x;
    private float y;

    public Point(float x, float y) {
        this.x = x;
        this.y = y;
    }

    void setCoordinates(final int x, final int y) {
        this.x = x;
        this.y = y;
    }

    public final float getX() {
        return x;
    }

    public final float getY() {
        return y;
    }

    @Override
    public String toString() {
        return "Coordinate x= "+ x + "Coordinate y= "+y;
    }

}
```

3.2 Circle

```
/*
```

```
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package desafio1;

/**
 * Implementa o Circulo, que herda
 * @author Francisco de Paula Dias Neto
 */
public class Circle extends Point {
   // variáveis de instancia privadas, isto é, não acessíveis de
      fora desta classe.
   private double radius;
   private String color;
   private double perimeter;
   // primeiro construtor o qual atribui valores iniciais a ambos±
      radius e color.
   public Circle() {
      this(1.0, "red", 0, 0);
   }

   // Segundo construtor que inicia radius com o parâmetro recebido,
      e mantem a cor com
   // o valor padrão definido.
   public Circle(double radius) {
      this(radius, "red", 0, 0);
   }

   public Circle(double radius, String color) {
      this(radius, color, 0, 0);
   }

   public Circle(double radius, String color, float x, float y) {
      super(x, y);
      this.radius = radius;
      this.color = color;
      this.setPerimeter();
   }

   public void setColor(String color) {
      this.color = color;
   }

   public void setRadius(double radius) {
      this.radius = radius;
   }

   private void setPerimeter() {
      this.perimeter = 2*radius*Math.PI;
   }
```

```
    public double getPerimeter() {
        return this.perimeter;
    }

    @Override
    public String toString() {
        return "Circulo± raio = "+ radius + "cor = "+ color;
    }

    public String getColor() {
        return color;
    }
    // Metodo de acesso para obter o valor armazenado em radius
    public double getRadius() {
        return radius;
    }
    // Metodo de acesso para computar a área de um circulo.
    public double getArea() {
        return radius*radius*Math.PI;
        // PI éa constante. Math éa classe onde PI édefinido.
    }
}
```

3.3 Triangle

```
package desafio1;

import java.lang.Exception;

public class Triangle {
    private double edge1;
    private double edge2;
    private double edge3;

    public Triangle(double edge1, double edge2, double edge3) throws
        Exception {
        validateEdges(edge1, edge2, edge3);
        this.edge1 = edge1;
        this.edge2 = edge2;
        this.edge3 = edge3;
    }

    public void validateEdges(double edge1, double edge2, double
        edge3) throws Exception {
        if (edge1 + edge2 <= edge3) {
            throw new Exception();
        }
        if (edge2 + edge3 <= edge1) {
            throw new Exception();
```

```
        }
        if (edge3 + edge1 <= edge2) {
            throw new Exception();
        }
    }

    public double getPerimeter() {
        return edge1 + edge2 + edge3;
    }

    public double getArea() {
        double p = getPerimeter() / 2;
        return Math.sqrt(p * ((p - edge1) * (p - edge2) * (p -
            edge3)));
    }

    public String getTriagleType() {
        int compareEdge12 = Double.compare(edge1, edge2);
        int compareEdge23 = Double.compare(edge2, edge3);
        int compareEdge13 = Double.compare(edge1, edge3);

        if (compareEdge12 == 0 && compareEdge23 == 0)
            return "Equilátero.";
        else if (compareEdge12 == 0 || compareEdge13 == 0 ||
            compareEdge23 == 0)
            return "Isosceles.";
        else
            return "Escaleno.";
    }

    public String getInternAngles() {
        double cousine3 =
            (Math.pow(edge1, 2) + Math.pow(edge2, 2) - Math.pow(edge3,
                2)) / (2*edge1*edge2);
        double angle3 = Math.toDegrees(Math.acos(cousine3));

        double cousine2 =
            (Math.pow(edge1, 2) + Math.pow(edge3, 2) - Math.pow(edge2,
                2)) / (2*edge1*edge3);
        double angle2 = Math.toDegrees(Math.acos(cousine2));

        double cousine1 =
            (Math.pow(edge2, 2) + Math.pow(edge3, 2) - Math.pow(edge1,
                2)) / (2*edge2*edge3);
        double angle1 = Math.toDegrees(Math.acos(cousine1));

        return angle1 + ", "+ angle2 + ", "+ angle3;
    }

    @Override
    public String toString() {
```

```
        return "Lados± "+ edge1 + ", "+ edge2 + ", "+ edge3 + "\n" +
           "rea± "+ getArea() + "\n" + "Perimetro± "+ getPerimeter()
              + "\n" +
           "Tipo do triagulo± "+ getTriagleType() + "\n" +
           "Angulos internos± "+ getInternAngles();
    }
}
```

3.4 Main

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package desafio1;

import java.lang.Exception;

/**
 *
 * @author Francisco de Paula Dias Neto
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // Declara c1 como variável habilitada a armazenar uma
           referencia para objeto da classe Circle.
        Circle c1;
        // atribui a c1 .a referencia retornada pelo construtor padrão
           Circle ()
        c1 = new Circle();
        System.out.println(c1.toString()); // chamada explicita
        // Para invocar os metodos classe Circle para operar sobre a
           instância c1,
        // usa-se o operador ponto (.).
        //Em outras palavras± usa-se o ponto para enviar uma mensagem
           ao objeto c1 para que
        // ele execute um de seus métodos.
        System.out.println("O circulo tem o raio de "+ c1.getRadius()
           + "e area de "+ c1.getArea());
        // Declara e aloca uma segunda instancia da classe Circle
           chamada c2
        // com o valor do radius igual a 2.0 e color com valor padrão.
        Circle c2 = new Circle(2.0);
        System.out.println(c2.toString()); // chamada explicita
```

```
        System.out.println(c2); // println() chama toString()
            implicitamente
        // Para invocar os metodos a operar sobre a instância c2,
            usa-se o operador ponto (".")
        // no exemplo abaixo o operador '+' invoca c2.toString()
            implicitamente.
        System.out.println("Aqui o operador '+' tambem invoca
            toString()± "+ c2);
        System.out.println("O circulo tem raio de "+ c2.getRadius() + "
            e area de "+ c2.getArea());

        Circle c3 = new Circle(); // constroi uma instancia de Circle
        c3.setRadius(5.0); // altera radius
        c3.setColor("Red"); // altera color

        try {
            Triangle triangle = new Triangle(3, 5, 5);
            System.out.print(triangle);
        } catch (Exception ex) {
            System.out.print("O triangulo não pode ser criado, os lados
                estão errados.");
        }
    }

}
```