

Atividade prática 02 – Classes e objetos

Objetivos

Habilitar o aluno a projetar, criar e implementar classes e instancias de objetos.

O que deve ser entregue

Apresentar um relatório sobre o trabalho observando o seguinte: ao final do **desafio 1** apresente o diagrama de classe das classes desenvolvidas, o texto fonte testado e responda às questões propostas. No **desafio 2**, para cada item, desenvolva e apresente no relatório os códigos desenvolvidos, eventuais resultados e comentários solicitados. O mesmo deve ser observado para o **desafio 3**.

Desafio 1

Neste desafio você vai trabalhar com a classe Circle, estudando sua declaração, uso de construtores, métodos modificadores e de acesso, instanciação de objetos e envio de mensagens aos objetos criados. Você deve corrigir eventuais erros identificados e executar os testes solicitados.

Construtores são rotinas presentes nas classes que permitem que um objeto da classe seja instanciado. Os construtores possuem o mesmo nome da classe e podem, ou não, possuírem parâmetros. Quando se declara uma classe ela terá, por “default”, um construtor padrão. É ele que é chamado e executado quando se usa o operador “new” seguido do nome classe e de um par de parênteses vazio. O programador pode reescrever e modificar este construtor padrão e até mesmo incluir outros com parâmetros. Os exercícios seguintes mostram como fazer isso.

O problema

Uma classe chamada Circle é projetada conforme mostrado no Diagrama de Classe abaixo.

Circle
<code>-radius:double = 1.0</code> <code>-color:String = "red"</code>
<code>+Circle()</code> <code>+Circle(radius:double)</code> <code>+getRadius():double</code> <code>+getArea():double</code>

Ela contém:

- Duas **variáveis de instância**: **radius** (do tipo double) e **color** (do tipo String), com valores iniciais de 1.0 e "red", respectivamente. O sinal “-” que as antecede indica que ambas são de acesso privado.

- Dois construtores: **Circle()**; e **Circle (radius : Double)**; O primeiro criará um objeto com as definições padrões mostradas no diagrama acima e o segundo criará um objeto iniciando o valor do raio com o valor passado como parâmetro.
- Dois métodos de acesso: **getRadius()**; e **getArea()**; Métodos de acesso são aqueles que apenas consultam e retornam os campos, ou variáveis de instância, de um objeto.
- O sinal "+" que antecede os construtores e métodos de acesso indica que são todos de acesso público.

O código fonte da classe Circle em Java é o seguinte:

```
public class Circle {           // salvar como "Circle.java"

    // variáveis de instancia privadas, isto é, não acessíveis de fora desta classe.

    private double radius;

    private String color;

    // primeiro construtor o qual atribui valores iniciais a ambos: radius e color .

    public Circle() {

        radius = 1.0;

        color = "red";

    }

    // Segundo construtor que inicia radius com o parâmetro recebido, e matem color com

    // o valor padrão definido.

    public Circle(double r) {

        this( ); // cria o objeto com o primeiro construtor:.Circle()

        color = "red";

    }

    // 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 é a constante π. Math é a classe onde PI é definido π.

    }

}
```

Edite e compile a classe Circle na plataforma de sua preferência. Você pode executar a classe Circle?

Esta classe não tem o método main(), logo não pode ser executada diretamente. Esta classe Circle pode ser considerada como uma definição de tipo para ser usada por outros programas.

Vamos então escrever um programa de teste, chamada TestCircle que usará a classe Circle:

```
public class TestCircle {    // salve como "TestCircle.java"

    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() );

    } // fim do método main()

} // fim da classe TestCircle
```

Edite, compile e execute TestCircle. Avalie os resultados.

Relatório: Neste ponto, coloque no relatório um diagrama de classes mostrando as classes desenvolvidas, os códigos desenvolvidos até aqui e suas respostas e avaliações sobre os testes realizados.

Desafio 2

Resolva o itens a seguir e responda no relatório.

1. **Construtor:** Modifique a classe **Circle** para incluir um terceiro construtor capaz de criar uma instancia de Circle com os valores de raio e cor fornecidos através de parâmetros.
2. Modifique o programa teste **TestCircle** para criar um nova instancia usando este novo construtor. O programa deverá ler os valores do raio e da cor.

```
// Construtor capaz de criar instancia de Circle com valores dados para o raio e cor.  
public Circle (double r, String c) {  
    -----  
}
```

3. **método Get:** Adicione um novo método Get para obter o valor da variável color de uma instancia da classe Circle. Modifique o programa para testar este método.

```
// metodo Get para a variável de instancia color  
public String getColor() {.....}
```

4. **public versus private:** Experimente no programa TestCircle, fazer acesso diretamente à variável de instancia radius (por exemplo, System.out.println(c1.radius)); ou atribuir um novo valor a (por exemplo, c1.radius=5.0). O que acontece? Teste , explique e justifique as mensagens recebidas.
5. **Método Set:** É razoável supor que seja necessário alterar os valores do raio e da cor de uma instancia de Circle, após ela ter sido criada. Métodos que alteram valores de variáveis de instância (ou campos) são chamados métodos modificadores. Adicione dois métodos para alterar estes valores conforme o modelo abaixo.

```
// metodo Set para a variável de instancia radius  
public void setRadius(double radius) {  
    -----  
}  
  
// metodo Set para a variável de instancia color  
public void setColor(String color) {  
    -----  
}
```

Modifique o programa TestCircle para testar estes métodos

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

6. **Palavra chave "this":** Em vez de usar nomes de variáveis como r (para radius) e c (para color) na lista de parâmetros dos métodos, é melhor usar nomes de variáveis como radius (para radius) e color (para color) e usar a palavra chave "this" para resolver o conflito entre nomes de variáveis de instancia e de parâmetros dos métodos. Veja exemplo a seguir.

```
// Variavel de instancia
private double radius;
// Método Set para o raio
public void setRadius(double radius) {
    this.radius = radius;    // "this.radius" refere à variável de instancia
                             // "radius" refere ao parametro.
}
```

Modifique todos os construtores e métodos da classe Circle para usar a palavra chave "this"

7. Acrescente um campo na classe para armazenar o comprimento do círculo. Crie um método Private para iniciar esse campo e modifique somente construtor Padrão para que esse campo seja iniciado no momento da instanciação do objeto "circle", independente do método usado. Crie também um método de acesso para retornar o valor desse campo.
8. **Method toString():** Uma classe bem projetada em JAVA deve conter um método public chamado toString() que retorna uma breve descrição da instancia (isto é, retorna um tipo String). O método toString() pode ser chamado explicitamente (via *variávelDeInstancia.toString()*) exatamente como qualquer outro método, ou implicitamente através do método println(). Se uma instancia é passada ao método println (ou seja, println(*variávelDeInstancia*)) o método toString() desta instancia é chamado implicitamente. Por exemplo, inclua o seguinte método toString() na classe Circle.
- 9.

```
public String toString() {
    return "Circulo: raio = " + radius + " cor = " + color;
}
```

Realize as tentativas indicadas a seguir.

- a) Tente chamar o método toString () explicitamente, como qualquer outro método.

```
Circle c1 = new Circle(5.0);

System.out.println(c1.toString());    // chamada explicita
```

- b) toString() é chamado implicitamente quando uma instancia é passada como parâmetro para o método println (), tente o exemplo abaixo.,

```
Circle c2 = new Circle(1.2);

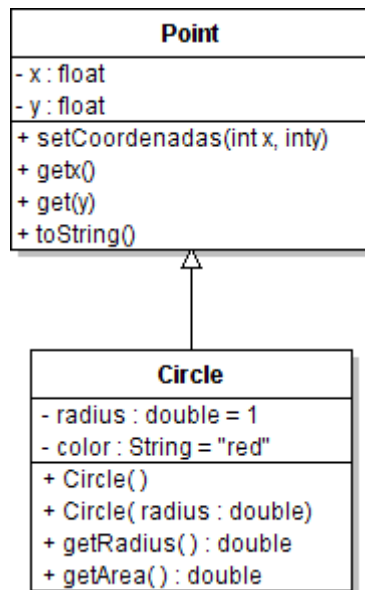
System.out.println(c2.toString());    // chamada explicita

System.out.println(c2); // println() chama toString() implicitamente
// no exemplo abaixo o operador '+' invoca c2.toString() implicitamente.

System.out.println("Aqui o operador '+' também invoca toString(): " + c2);
```

Desafio 3

1. Acrescente uma superclasse Point conforme mostrado no diagrama abaixo.



Implemente a nova classe e faça as alterações necessárias na subclasse Circle para atribuição, acesso e exibição das coordenadas do centro do círculo.

2. Acrescente uma classe ao programa acima para instanciar objetos do tipo triângulo, sendo que na criação dos triângulos deverão ser dados apenas os comprimentos dos três lados. Pode acontecer que dados os três lados, eles não formem um triângulo! Qual a sua sugestão para esse problema?

A classe deverá ter métodos que retornem o perímetro do triângulo, a área do triângulo e o tipo do triângulo (isósceles, escaleno, equilátero, retângulo, etc.). O método `toString` deverá imprimir todas as propriedades do triângulo (lados, área, perímetro, tipo do triângulo e ângulos internos).

O programa deverá ser feito observando **rigorosamente** o conceito de orientação a objetos, usando um método para cada “função” de acesso, modificação ou calculo de alguma propriedade.