

Questão 1 – Classe FormaGeometrica

...50

Crie uma classe para representar uma **FormaGeometrica**. Uma **FormaGeometrica** possui os atributos `posX` e `posY` indicando a posição da forma na tela.

Esta classe possui os métodos:

- `getPosString`: retorna uma **String** contendo a posição da forma geométrica na tela conforme o exemplo abaixo (para `posX = 32` e `posY = 87`):

```
| posição (32, 87)
```








- `getArea()`: método abstrato, não possui implementação.
- `getPerimetro()`: método abstrato, não possui implementação.

Note que como a classe possui métodos abstratos, ela também deverá ser declarada como abstrata. Note também que como a classe é abstrata, você não terá como gerar instâncias dela (objetos) para testar o método `getPosString`.

O prazo de entrega do trabalho terminou. Portanto, o botão abaixo está desabilitado.

Enviar "FormaGeometrica.java"

Solução correta!







 FormaGeometrica
 <code>posX: int</code>  <code>posY: int</code>
 <code>FormaGeometrica(int posX, int posY)</code>  <code>getArea(): double</code>  <code>getPerimetro(): double</code>  <code>getPosString(): String</code>

Questão 2 – Classe Circulo

2.00 / 2.00 

Crie uma classe para representar um **Circulo**. Um **Circulo** é uma subclasse da classe **FormaGeometrica** que, além dos atributos e métodos herdados, possui o atributo `raio` e os métodos:

- `getArea()`: implementação do método abstrato que veio da superclasse. Retorna a área do círculo. **Importante:** use a constante `PI` do java para realizar o cálculo (veja dicas abaixo).
- `getPerimetro()`: implementação do método abstrato que veio da superclasse. Retorna o perímetro do círculo. **Importante:** use a constante `PI` do java para realizar o cálculo (veja dicas abaixo).
- `toString`: sobrepõe o método `toString` da classe **Object**. Deve retornar uma descrição

 Circulo
 <code>raio: double</code>
 <code>Circulo(int posX, int posY, double raio)</code>  <code>getArea(): double</code>  <code>getPerimetro(): double</code>  <code>toString(): String</code>

do círculo atual conforme exemplo abaixo. **Importante:** como este método sobrepõe outro, este precisa ter o mesmo modificador de acesso (ou um mais permissível) que o original. Neste caso, o método precisa ser `public` (note o círculo verde na descrição do método no diagrama de classes ao lado).

```
Círculo na posição (32, 87) com raio de 6.0cm  
(área=113.09733552923255cm2, perímetro=37.69911184307752cm)
```

Dicas:

- A área de um círculo é dada pela fórmula: $A = \pi r^2$
- O perímetro de um círculo é dado pela fórmula: $C = 2\pi r$
- Use a constante `PI` presente na classe `Math` do Java para pegar o valor de π . Exemplo:
`double pi = Math.PI;`

O prazo de entrega do trabalho terminou. Portanto, o botão abaixo está desabilitado.

Enviar "Circulo.java"

Solução correta!

Questão 3 – Classe Retangulo

3.00 / 3.00

Crie uma classe para representar um `Retangulo`. Um `Retangulo` é uma subclasse da classe `FormaGeometrica` que, além dos atributos e métodos herdados, possui os atributos `largura` e `altura` e os métodos:

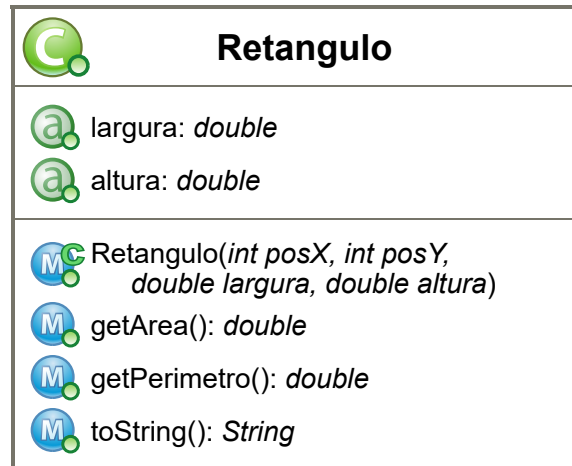
- `getArea()`: implementação do método abstrato que veio da superclasse. Retorna a área do retângulo.
- `getPerimetro()`: implementação do método abstrato que veio da superclasse. Retorna o perímetro do retângulo.
- `toString`: sobrepõe o método `toString` da classe `Object`. Deve retornar uma descrição do retângulo atual conforme exemplo abaixo.

```
Retângulo na posição (12, 65) com largura de 2.0cm e altura de 7.0cm  
(área=14.0cm2, perímetro=18.0cm)
```

Dicas:


- A área de um retângulo é dada pela fórmula: $A = largura * altura$
- O perímetro de um retângulo é dado pela fórmula: $C = 2(largura + altura)$

O prazo de entrega do trabalho terminou. Portanto, o botão abaixo está desabilitado.






Solução correta!

Questão 4 – Classe Quadrado

3.00 / 3.00 

Crie uma classe para representar um **Quadrado**. Um **Quadrado** é uma subclasse da classe **Retangulo**. Note que ele não possui atributos adicionais (irá usar os atributos largura e altura da superclasse). Os métodos `getArea` e `getPerimetro` serão herdados da superclasse. Será necessário apenas sobrepor o método `toString`.

	Quadrado
<hr/>	
	<code>Quadrado(int posX, int posY, double lado)</code>
	<code>toString(): String</code>

- `toString`: sobrepõe o método `toString` da classe **Object**. Deve retornar uma descrição do quadrado atual conforme exemplo abaixo.

```
Quadrado na posição (45, 39) com lado de 6.0cm (área=36.0cm2,
perímetro=24.0cm)
```

O prazo de entrega do trabalho terminou. Portanto, o botão abaixo está desabilitado.

Solução correta!

Questão 5 – Classe FormasMain

0.50 / 0.50 

Por fim, para exercitar o conceito de *polimorfismo* crie uma classe chamada **FormasMain** que terá o método `main`. Neste método, crie um vetor de objetos da classe **FormaGeometrica**. Crie e insira no vetor um ou mais objetos das classes **Circulo**, **Retangulo** e **Quadrado**. Em seguida, faça um **for** para iterar entre todos os elementos e mande imprimir cada um dos objetos. Como você sobrepôs o método `toString`, você pode mandar imprimir diretamente. Exemplo:
`System.out.println(circulo1);`

O prazo de entrega do trabalho terminou. Portanto, o botão abaixo está desabilitado.

Solução correta!