Observação:

Para cada questão resolvida, crie um pacote com o nome questaoXX, onde XX representa o número da questão (01, 02, 03, etc).

1) Estruturas de controles e laços

Crie um programa em Java que imprima os seguintes padrões (dica: perceba que esse padrão requer que cada linha comece com um número apropriado de espaços em branco):

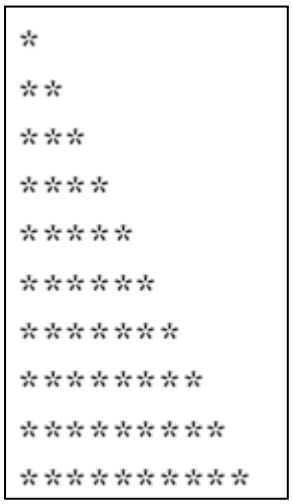


Figura 2. Padrão 1.

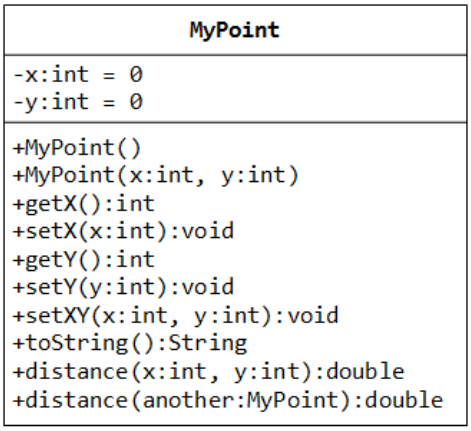
Figura 4. Padrão 2.

Figura 3. Padrão 3.

Figura 1. Padrão 4.

2) Manipulação de objetos: equals e toString

Crie uma classe chamada MyPoint para representar um ponto e se apresenta na seguinte forma (em UML):



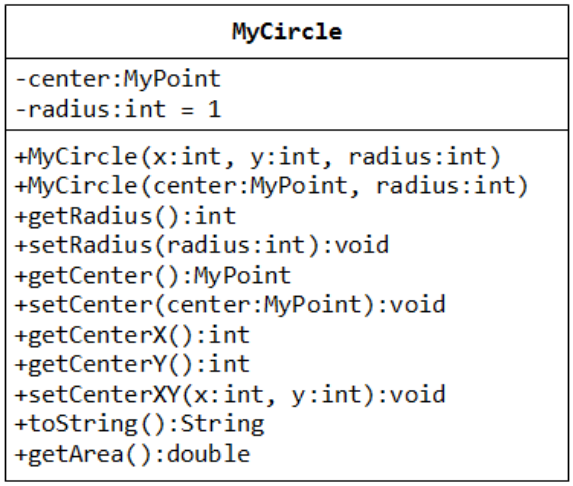
Obs.: o método toString retorna o formato do “MyPoint (x, y)”

a) Defina o método equals da classe MyPoint que utilize os valores de x e y na comparação.

b) Defina uma classe TestMyPoint que contenha um método main e teste todos os métodos e construtores da classe definida. Não precisa receber valores do teclado. Nestes testes, crie duas instâncias de MyPoint com coordenadas iguais e imprima se os pontos são iguais usando o método equals. Então, altere uma das coordenadas de um dos pontos e imprima novamente se os pontos são iguais.

c) **(DESAFIO)** Escreva um programa que declare e inicialize 10 pontos usando um laço e um array de MyPoint. Exemplo (1,1) (2,2) ... (10,10). Depois imprima cada um desses pontos.

Crie uma classe chamada MyCircle seguindo a descrição em UML e reusando a definição da classe MyPoint:

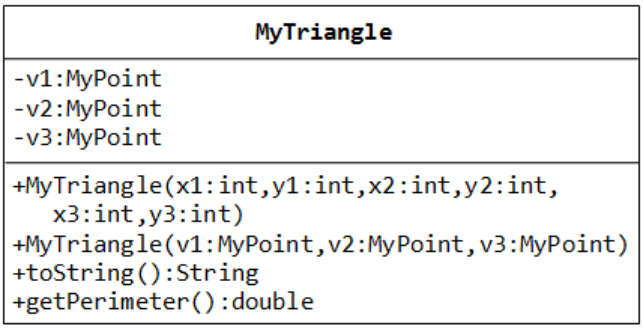


Obs.: o método toString retorna o formato do "Circle @ (*x*, *y*) radius=*r; area=a*"

d) Defina o método equals da classe MyCircle que utilize os valores do raio e as coordenadas do centro. Para comparar dois pontos, você precisará invocar o equals de MyPoint também.

e) Defina uma classe TestMyCircle que contenha um método main e teste todos os métodos da classe MyCircle definida. Não precisa receber valores do teclado. Nestes testes, crie duas instâncias de MyCircle com coordenadas de centro e raios iguais e imprima se os círculos são iguais usando o método equals. Então, altere uma das coordenadas de um dos centros e imprima novamente se os círculos são iguais.

Crie uma classe chamada MyTriangle para representar um ponto e se apresenta na seguinte forma (em UML):



Obs.: o método toString retorna o formato do "`MyTriangle TYPE @ (x1, y1), (x2, y2), (x3, y3)`", onde **TYPE informará um tipo do triângulo: isósceles, equilátero ou escaleno.**

f) Defina o método equals da classe MyTriangle que utilize os valores das coordenadas dos três vértices. Para comparar dois pontos, você precisará invocar o equals de MyPoint também.

g) Defina uma classe TestMyTriangle que contenha um método main e teste todos os métodos da classe MyTriangle definida. Não precisa receber valores do teclado. Nestes testes, crie duas instâncias de MyTriangle com coordenadas dos vértices iguais e imprima se os triângulos são iguais usando o método equals. Então, altere uma das coordenadas dos vértices e imprima novamente se os círculos são iguais.

3) Entendendo novas APIs: utilizando pacote java.time

Aprender uma nova API consiste em entender como os métodos devem ser chamados, em que ordem e para que propósito. Para isso, é necessário utilizar a documentação oficial Java (JavaDOC) e documentos de apoio que demonstrem o uso da API. Esse exercício deve fazer vocês entenderem como funciona a representação de datas em Java e como a API pode ser usada em um programa.

Para realização deste exercício, você deve utilizar as classes do pacote `java.time.*` para executar o seguinte procedimento:

- Peça que o usuário digite o dia, mês e ano de seu nascimento e transforme em um objeto do tipo apropriado que será usado em outras partes do seu programa.
- Imprima a quantidade total de anos, meses, dias e horas que esse usuário já viveu, isto é, a diferença da data de nascimento para a data de hoje.
- Some 1 ano à data de nascimento informada e imprima em qual dia da semana caiu o primeiro aniversário daquele usuário.
- Some mais 17 anos ao mesmo objeto que representa a data e imprima em qual dia da semana caiu o aniversário de 18 anos daquele usuário.

DESAFIO

- Crie uma classe auxiliar chamada DateUtils que contenha um método que recebe duas datas do tipo java.time.LocalDate e retorne a quantidade de dias úteis (excluindo-se sábados e domingos) contidos entre essas duas datas.
- Para testar o método do item anterior, durante a execução do seu programa, peça que o usuário forneça duas datas usando o teclado (você pode usar o mesmo procedimento criado antes) e então invoque o método criado no item anterior para calcular a quantidade de dias úteis entre essas datas. Repita esse procedimento três vezes, dando como entrada seis datas diferentes, duas datas para cada chamada do método, permitindo assim a avaliação da precisão do mesmo.