Terceira Lista de Exercícios

1. Crie uma interface chamada `Mensuravel` com um método `double getMedia()` para medir um objeto de alguma forma. Crie a classe `Funcionario` com atributos `nome` e `salário` e a faça implementar a interface `Mensuravel`. Finalmente, forneça o método `double media(Measuravel[] objs)` que calcula a média de objetos mensuráveis. No programa principal, use o método `média` para calcular o salário médio de um array de funcionários. (FEITO)
2. Continuando o exercício anterior, forneça o método `Mensuravel maior(Measuravel[] objs)`. Use-o para encontrar o nome do empregado com o maior salário. Você precisou fazer algum casting, onde, por que?. (FEITO)
3. Quais são os supertipos da classe `String` e `Scanner`? Lembre-se que em java uma classe ou interface sem supertipo declarado tem como supertipo `Object`.
   R: Ambas herdam de Object. A classe String implementa Serializable, Comparable<String>, CharSequence. Já a classe Scanner implementa Iterator<String>, Closeable.

4. Implemente o método estático `of()` da interface `IntSequence` que retorna um objeto que reproduz a sequência de inteiros recebida como argumento. Note que o número de parâmetros não é fixo. Por exemplo, `IntSequence.of(3, 1, 4, 1, 5, 9)` retorna um objeto que reproduz uma sequência com seis valores, `IntSequence.of(3, 1, 4, 1, 5, 9, 7, 6)` retorna um objeto que reproduz uma sequência com oito valores. (FEITO)

5. Refaça o exercício 4 implementando o método `of()` para retornar uma instância de uma classe anônima. (FEITO)

6. Adicione à interface `IntSequence` um método estático com o nome `constante()` que produz uma sequência constante infinita. Por exemplo, `IntSequence.constant(1)` produz valores 1 1 1..., ad infinitum. (FEITO)

7. Implemente o exercício 6 usando uma expressão lambda. (FEITO)

8. Implemente o método `void ordenaNaSorte(ArrayList<String> strings, Comparator<String> comp)` que continua chamando `Collections.shuffle()` para embaralhar o ArrayList de strings até que os elementos estejam em ordem crescente, tal como determinado pelo comparador. (FEITO)

9. Escreva um método que receba um array de instâncias de classes que implementem a interface `Runnable` e retorne um objeto `Runnable` cujo método `run()` executa os objetos `Runnable` recebidos como parâmetro em ordem. Faça o retorno usando uma expressão lambda. (FEITO)

10. Faça uma chamada para `Arrays.sort()` para ordenar um array de objetos funcionários, tal como definidos no exercício um, com base no salário, em caso de salários iguais a ordenação deve ser por nome, em ordem alfabética. (FEITO)

11. Defina a classe `Ponto` para representar um ponto com coordenadas x e y. Forneça um construtor para inicializar as coordenadas e métodos de acesso `getX()` e `getY()`. Defina a subclasse `PontoRotulado` com um construtor `PointoRotulado(String rotulo, double x, double y)` e um método de acesso `getRotulo()`. (FEITO)

12. Defina os métodos `toString()`, `equals()` e `hashCode()` para as classes do exercício 11. (FEITO)

13. Explique as diferenças entre visibilidade de atributos `public`, `private` e `protected` em java.
    R: Public pode ser acessado/vista por qualquer classe que instancie um objeto da classe à qual o atributo público foi declarado.
    Private só pode ser acessado/visto na classe em que foi declarada através de seus métodos.
    Protected pode ser acessado/visto somente pelas classes onde foi declarada e por classes que herdaram a classe onde a variável foi declarada.

14. Defina uma classe abstrata chamada `Shape` com uma variável de instância da classe `Point` (que modela um ponto tal como o do exercício 11), um construtor, um método concreto público `void moveBy(double dx, double dy)` que move o ponto por uma dada distância nas direções x e y, e um método abstrato público `Point getCenter()`. Defina subclasses concretas `Circle`, `Rectangle`, `Line` com construtores `public Circle(Point center, double radius)`, `public Rectangle(Point topLeft, double width, double height)`, e `public Line(Point from, Point to)`. (FEITO)

15. Defina métodos clone para as classes dos exercícios 11 e 14. (FEITO)