

Aluno(a): _____ Data: _____

Códigos desnecessários e que reduzam o desempenho do sistema serão penalizados.
Utilize as boas práticas de programação, sempre que possível. Vale lembrar que você deve declarar atributos de instância como privados.
LEIA AS QUESTÕES ATÉ O FINAL ANTES DE COMECAR.

Questão 1 (5,6) – Desenvolva o código para os itens abaixo. Nessa questão, sempre que um método for void, imprima no console alguma mensagem seguindo o padrão de aula (ex: para o método **voando()** de uma classe X, imprima no console “X voando”. **NÃO** utilize generics nessa questão.

- ❖ Crie a interface Treinamento com um método public void treinar().
- ❖ Crie uma classe concreta chamada Ninja que implementa Treinamento.
 - Crie os atributos cpf, nome e idade.
 - Crie apenas dois construtores nessa classe. O primeiro construtor inicializa os 3 atributos acima. O segundo construtor inicializa apenas o cpf.
- ❖ Crie 3 subclasses de Ninja: Genin, Chunin e Jonin.
 - A classe Genin deve sobrescrever o método treinar(), imprimindo no console a string “Genin XXX iniciando treinamento simples”, em que XXX é o valor do atributo **nome**, definido na classe Ninja.
 - A classe Chunin deve sobrescrever o método treinar(), imprimindo no console a string “Chunin XXX iniciando treinamento avançado”, em que XXX é o valor do atributo **nome**, definido na classe Ninja.
 - A classe Jonin deve sobrescrever o método treinar(), imprimindo no console a string “Jonin XXX iniciando treinamento de meditação”, em que XXX é o valor do atributo **nome**, definido na classe Ninja.
- ❖ Crie a classe concreta chamada Academia.
 - Crie o atributo nome, não crie os getters e setters.
 - Crie apenas um construtor, que permita inicializar o atributo nome.
 - Crie um atributo denominado *listaNinjas* do tipo List.
 - Crie apenas o getListaNinjas()
 - Para adicionar um Ninja a *listaNinjas*, crie o método *matriculaNinja* que permita inserir um Ninja em *listaNinjas*, ou seja, na lista de ninjas matriculados nessa academia.
 - Crie, ainda, um método *imprimeDadosNinjaSeExistir* que recebe como parâmetro um Ninja e, caso o Ninja exista em *listaNinjas*, imprime no console o nome, cpf e idade desse Ninja. **NÃO É PERMITIDO USAR QUALQUER MÉTODO PARA PERCORRER A LISTA, OU SEJA, NÃO USE WHILE, FOR, ITERATOR, ETC.** PS: Dois objetos que sejam instâncias de Ninja são considerados iguais se possuem o mesmo cpf. Além disso, esse método retorna true se o Ninja existe na lista e false se não existe.
 - Sobrescreva o método que irá permitir que, quando uma referencia a uma academia a for impressa no console (syso(a)), seja exibido o nome da academia concatenado com a quantidade de alunos matriculados (Ex: “AcadeNinja - 15”, “NinjaMia - 13”).
- ❖ Crie uma classe SistemaPrincipal.
 - Crie uma academia.
 - Para essa academia, receba do console o número N de ninjas que serão cadastrados.
 - Solicite, também no console, a informação de se o Ninja é um Genin (1), Chunin (2) e Jonin (3). Ou seja, se o usuário digitar 1, será criado um Genin, e assim por diante.
 - Cadastre os dados dos N ninjas inserindo os dados de nome, cpf e idade no console.
 - Em seguida, insira esses ninjas na academia.
 - Imprima no console “Cadastro dos ninjas realizado com sucesso!”

- Em seguida, faça um loop (while) que, enquanto não for digitado "0" (zero), será solicitado um cpf e o sistema tentará encontrar na lista dos ninjas da academia, o ninja associado a esse cpf (utilize o método *imprimeDadosNinjaSeExistir*). Dessa maneira, se o ninja for encontrado, os dados desse ninja devem ser exibidos. Se o Ninja não existir na academia, uma mensagem deve informar que não existe ninja com esse cpf.
- Ao final, imprima no console a referencia à academia (ex: syso(a)), o que deve exibir no console o nome da academia e o número de alunos matriculados. Para isso, você vai precisar alterar a classe Academia.
- Por fim, imprima os nomes dos ninjas, ordenados em ordem crescente por idade, ou seja, primeiro aparecerão os mais novos. Para isso, utilize o método de Collections que permite ordenar e modifique a classe Ninja, se for necessário.
- ❖ Crie uma classe Utils com apenas um método estático: Map retornaMapa (List listaNinjas)
 - Considerando que listaNinjas contém apenas objetos do tipo ninja, o método acima recebe listaNinjas e retorna um novo mapa em que a chave é composta dos cpfs dos ninjas em listaNinjas e o valor é composto dos objetos Ninja em listaNinjas.

Questão 3 (0,4) (Tribunal de Justiça do Rio Grande do Norte - RN (TJ/RN/RN) 2020 - Cargo: Analista de Sistemas

Sobre a linguagem de programação Java, analise as afirmativas abaixo.

I É possível instanciar uma classe abstrata.

II A palavra chave inherits define a herança de uma classe para outra.

III A palavra chave super é utilizada para fazer referência à classe pai herdada.

IV A desalocação de memória é feita de forma automática pelo garbage collector.

Estão corretas as afirmações

- (a) I e II
- (b) I e IV
- (c) III e IV
- (d) II e III

Questão 4 (1,5) Observe as classes abaixo e diga o que sai no console.

```
public class Panela {
    private String cor;
    private int quantidade;
    private double preco;
    public String getCor() {
        return cor;
    }
    public void setCor(String co) {
        cor = co;
    }
    public int getQuantidade() {
        return quantidade;
    }
    public void setQuantidade(int quantidade) {
        quantidade = quantidade;
    }
    public double getPreco() {
        return preco;
    }
    public void setPreco(double prec) {
        preco = prec;
    }
}
```

```

public class Programa2Noite {
    public static void main(String[] args) {
        Panela p = new Panela();
        p.setCor("amarelo");
        Panela p2 = new Panela();
        p2.setCor("preto");
        p2.setQuantidade(101);
        metodoA(p);
        System.out.println(p.getCor());
        System.out.println(p.getQuantidade());
        metodoB(p2, p);
        System.out.println(p.getCor());
        System.out.println(p2.getCor());
        int i = 190;
        qqq(i);
        System.out.println(i);
        int j = 150;
        qqq3(i, j);
        System.out.println(i);
        System.out.println(j);
    }
    public static void metodoA (Panela p2) {
        new Panela();
        p2.setCor("azul");
        p2.setQuantidade(10);
    }
    public static void metodoB (Panela p9, Panela p10) {
        Panela p11 = p9;
        p11.setQuantidade(3);
        p11.setCor("rosa");
        p9 = p10;
        p10 = p11;
        p9.setCor("preto");
    }
    public static int qqq (int i) {
        i = i + 665;
        i = 15;
        return i;
    }
    public static void qqq3 (int j, int i) {
        int k = j;
        j = i;
        i = k;
    }
}

```

Questão 1 – (2,5) – Considere que existe uma classe denominada Utilidades. Nessa classe existe um método estático com a seguinte assinatura: `getPalavroes():Set`. Esse método retorna uma lista de palavrões existentes para a língua portuguesa. Não crie essa classe nem esse método, eles já existem para você utilizar.

Crie um programa que receba do console (entrada do teclado) uma frase. Em seguida, esse programa deve chamar o método `Utils.verifica(String texto)`. Esse método é responsável por verificar se o texto (variável “texto”) tem menos de 10 caracteres e, caso tenha, deve lançar uma exceção **checked** denominada `TextoCurtoException`. Crie também a classe dessa exceção. Quando essa exceção for lançada, deve ser passada uma mensagem para o construtor dela, informando quantos caracteres possui o “texto” inserido pelo usuário. Ex: “texto inserido com 4 caracteres.”

Em seguida, crie o método `Utils.moderar(String texto): String`. Esse método só deve ser chamado se o método `verifica(String texto)` não lançar exceção. O `Utils.moderar(String texto): String` deve substituir todos os palavrões encontrados em “texto” por “#####” -> Cada caractere do palavrão deve ser substituído por um caractere #. Ou seja, se o palavrão tiver 6 caracteres, o palavrão no texto deverá ser substituído por #####.

PS: É importante deixar claro que nessa questão os métodos devem considerar que se um palavrão for escrito em letra minúscula ou letra maiúscula, este deve ser substituído.

PS2: Considere que os palavrões são palavras simples, não compostas, ou seja, não existem espaços.

Exemplo de entrada e saída: A casa palavra1 é minha. -> A casa ##### é minha.

ADICIONAIS

Questão 1 (1) – Escreva um método *quantasVezes* em uma classe denominada `StringUtils` que receba duas strings como argumentos e retorne o número de vezes que a segunda string aparece na primeira. Por exemplo, se a string “recrearem” e “re” forem passadas como argumentos, o método deverá retornar 3 (“recreremere”).

Questão 2 (1) – Uma string é dita palíndroma se puder ser lida da esquerda para a direita ou da direita para a esquerda da mesma forma. As strings “radar”, “asa”, “ovo” são palíndromas. Escreva um método na classe `StringUtils` que retorne `true` se uma string passada como argumento for palíndroma e `false` caso não seja. Cuidado com as letras maiúsculas e minúsculas.

BOA SORTE!