Aula 03 - Revisão de conceitos da orientação a objetos e linguagem java.

Lista de atividades.

Considerando a aula de revisão de programação orientada a objetos e características e funcionalidades da linguagem java, comente as questões a seguir:

1. O que é um framework? Qual a diferença de um framework para a API de programação da linguagem java?

2. Crie uma annotation que permite marcar um atributo de uma classe como chave primária. O nome da annotation deve ser @PrimaryKey.

3. Implemente uma classe Candidato com dois atributos: String conceito e Integer anoNascimento. Implemente a interface Comparable para garantir que a ordenação de candidatos seja primeiramente feita pelo conceito (A, B, C, D) e em caso de empate (dois candidatos com o mesmo conceito) usar o ano de nascimento como critério de desempate. Desse modo quando chamar o metodo Collections.sort e tendo na nossa "base de dados" o candidato1("B", 1990) e o candidato2("B", 1989) deve retornar em primeiro o candidato2 em primeiro lugar.

```
public static void main(String[] args) {
    List<Candidato> candidatos = new ArrayList<>();
    candidatos.add(new Candidato("C", 2010));
    candidatos.add(new Candidato("B", 2004));
    candidatos.add(new Candidato("A", 2001));
    candidatos.add(new Candidato("B", 1999));
    candidatos.add(new Candidato("A", 1999));

    System.out.println("Lista original:");
    candidatos.forEach(System.out::println);

    System.out.println("Lista ordenada:");
    Collections.sort(candidatos);
    candidatos.forEach(System.out::println);
}
```

A saída do programa deve ser:

```
Lista original:
Candidato:1 | Conceito:C | Ano: 2010
Candidato:2 | Conceito:B | Ano: 2004
Candidato:3 | Conceito:A | Ano: 2001
Candidato:4 | Conceito:B | Ano: 1999
Candidato:5 | Conceito:A | Ano: 1999
Lista ordenada:
Candidato:5 | Conceito:A | Ano: 1999
Candidato:3 | Conceito:A | Ano: 2001
Candidato:4 | Conceito:B | Ano: 1999
Candidato:2 | Conceito:B | Ano: 2004
Candidato:1 | Conceito:C | Ano: 2010
```

4. Pesquise sobre a interface Comparator. Esta interface é semelhante a interface Comparable porém permite criar *diferentes tipos de ordenação*. Por exemplo, considerando que a classe Candidato tem um campo Float rendaMensal, vamos criar duas possibilidade de ordenação: (1) ordena por ano de nascimento e rendaMensal; (2) ordena por conceito e data de nascimento.

5. Usando filter() crie uma expressão lambda que filtra filmes de “sci-fi” com nota maior>=9. Use a classe Database.getMovies() para obter a lista de filmes como exemplo para executar o filter.

6. Escreva uma expressão lambda que remove todos os usuários que possuem pontos <= 200 (ver classe Usuario.java). [.removeIf()]

7. Escreva uma função lambda que seta o usuário como moderador = true, se este possui pontos maior que 300.