Java Avançado

Exercícios Propostos

Manipulando coleções com a Stream API

1 Exercício

Considere a lista abaixo que representa ângulos em graus:

```
List<Integer> angulosGraus = Arrays.asList(90, 30, 60, 45, 180);
```

Crie uma aplicação que, usando a Stream API, gere como resultado um objeto do tipo `List<Double>` contendo os mesmos ângulos em radianos.

Dica: A classe `Math` possui o método estático `toRadians()` que faz a conversão.

2 Exercício

Considere a lista de cores abaixo:

```
List<String> cores = Arrays.asList(
  "Azul", "Branco", "Preto", "Preto", "Amarelo", "Azul");
```

Usando a Stream API, gere uma nova lista com elementos do tipo `Papel`. Veja abaixo os detalhes desta classe:

```
public class Papel {
    public enum Cor {
        Branco, Preto, Azul, Amarelo
    }

    private int id;
    private Cor cor;

    public Papel(int id, Cor cor) {
        this.id = id;
        this.cor = cor;
    }

    // Métodos getters e setters...
}
```

Perceba que objetos da classe `Papel` devem possuir um ID. Este ID deve ser um número único gerado no momento da criação do objeto. Ele deve iniciar em 1 e ser incrementado em 1 a cada novo objeto.

Dica: Para converter um objeto `String` para um elemento do enum `Cor`, use o método `Papel.Cor.valueOf()`.

3 Exercício

Considere a seguinte lista de números, representados como objetos `String`:

```
List<String> numeros = new ArrayList<>();
numeros.add("5");
numeros.add("31");
numeros.add("22");
numeros.add("14");
numeros.add("9");
numeros.add("30");
numeros.add("18");
```

Usando a Stream API, calcule a soma dos números da lista.

Resolva o exercício duas vezes, em cada uma utilizando um dos métodos de mapeamento: `map()` e `mapToInt()`.