Aula 11

Trabalho Autónomo

Enumerados, Interface e Comparadores

1. Dado o seguinte enumerado:

   ```
   public enum Options{SAVE; LOAD; EXIT;}
   ```

   Assuma que está corretamente inicializado e diga o que falta para completar a condição seguinte de modo a correr o corpo da condição caso a variável contenha EXIT.

   ```
   Options op = …
   if (op == ____) {
           ....
   }
   ```

2. Quais dos seguintes são métodos já disponiveis para os enumerados em Java (selecione todas as respostas corretas):
   a. `name()`
   b. `getValue()`
   c. `read()`
   d. `value()`

3. Crie um enumerado com os dias da semana (DOMINGO, SEGUNDA, ..., SÁBADO). Este enumerado deve ter dois método instanciados: `boolean isWeekDAY()` e `boolean isWeekend()` - este último método retorna o oposto do primeiro método. Escreva um programa que demonstra como este enum poderia ser usado, sabendo que tem um método que toma como argumento um dia da semana e imprime uma mensagem dependendo se o dia da semana é ou não fim-de-semana.
   a. Sugestão - O método main percorre todos os valores do enumerado e envia os seus valores com argumento para o método.
   b. Utilizando as exceções que conhece, complete o exercício anterior.

4. Defina um enumerado Suit para representar os quatro naipes das cartas de jogar (espadas, paus, copas, e ouros).
    a. Defina um enumerado Rank para representar os treze valores possíveis para uma carta de jogar (Às, 7 (Manilha), Rei, Valete, Dama, 10, 9, 8, 6, 5, 4, 3, 2), assumindo que estas vão ser utilizadas para o jogo da Sueca (embora as cartas 10, 9, e 8 não sejam utilizadas). Desta forma, as cartas têm uma ordem de valor, pelo que a ordem pela qual as constantes do enumerado são definidas poderá ser relevante (para determinar qual o valor mais alto entre duas cartas). Inclua informação relativa aos pontos que cada carta vale no jogo da Sueca (Ás=11, Manilha=10, Rei=4, Valete=3, Dama=2, outras=0).
    b. Defina uma função de teste que dado uma lista de Rank devolve o somatório de pontos.
    c. Utilizando as exceções que conhece, complete o exercício anterior.

5. Ordenação de números racionais (Interface Comparable)

Implemente a interface Comparable (completando adequadamente o método compareTo) para a classe Rational (que representa números racionais), tal como definida abaixo. Não se esqueça que só pode comparar o numerador de duas frações depois de terem o mesmo denominador. A igualdade deve estar definida de modo coerente com a comparação. Defina também a igualdade (método equals) para números racionais. Depois de completar este exercício altere o main para criar um vector de racionais e use o método Arrays.sort() para ordenar essa vector.

```
public class Main {

	public static void main(String[] args) {
		Rational r1 = new Rational (2, 3);
		Rational r2 = new Rational (3, 4);

		if (r1.compareTo(r2) > 0)
			System.out.println(r1 + " is bigger than " + r2);
		else
			System.out.println(r1 + " is not bigger than " + r2);
	}
}
```

```java
public class Rational implements Comparable<Rational> {
	private int numerator;
	private int denominator;

	public Rational(int numerator, int denominator) {
		this.numerator = numerator;
		this.denominator = denominator;
	}

	public int getNumerator() {
		return numerator;
	}

	public int getDenominator() {
		return denominator;
	}

	@Override
	public String toString() {
		return numerator + "/" + denominator;
	}

	@Override
	public int compareTo(Rational other) {
		// TODO Para completar...
		return 0;
	}
}
```

6. Ordenação de números racionais (Interface Comparator)

Crie uma classe Aluno e implemente dois comparadores diferentes para essa classe: um que compara por número de aluno; outro que compara por nome do aluno. Experimente ordenar uma lista de alunos usando primeiro um comparador.

7. Ordenação de números racionais (Interface Comparable)

Desenvolva uma classe que permita representar uma pessoa, de acordo com as seguintes indicações:

- A classe deve denominar-se Contacto;
- Deve incluir apenas um construtor, onde são passados o nome e o telefone da pessoa em questão;
- Deve disponibilizar, através de dois inspectores, a consulta do nome e telefone;
- Deve disponibilizar dois modificadores:
  - public void modificaTelefone(final int telefone), que muda o telefone da pessoa;
  - public void modificaNome(final String nome), que modifica o nome da pessoa.
- Deve redefinir o método equals e implementar o interface Comparable, sendo comparação por ordem alfabética de nome;
- Inclua a possibilidade de pessoas com o mesmo nome serem ordenadas pelo número de telefone.