Lista de Exercícios 2 de Programação Orientada a Objetos

Exercícios de OO

1. Faça um programa para representar a árvore genealógica de uma família. Para tal, crie uma classe Pessoa que permita indicar, além de nome e idade, o pai e a mãe. Tenha em mente que pai e mãe também são do tipo Pessoa.

2. Faça um programa que calcule a área de uma figura geométrica. Aceite quatro tipos de figura geométrica: quadrado, retângulo, triângulo e círculo. Use herança e polimorfismo.

3. Faça um programa de agenda que permita guardar dois tipos de contato: pessoa física e pessoa jurídica. Para pessoa física, guarde o nome, o CPF, o endereço e a data de aniversário. Para pessoa jurídica, guarde a razão social, o CNPJ, o endereço e o faturamento. Permita tanto listar todos os contatos da agenda quanto buscar um contato específico pelo seu CPF, se for pessoa física, ou pelo CNPJ, se for pessoa jurídica. Use herança e polimorfismo.

4. Qual a diferença entre abstração, encapsulamento e modularidade?

Exercícios de Tratamento de Exceções

5. Refaça os exercícios 1, 2 e 3 da lista 1 para protegê-los de qualquer tipo de exceção que pode ocorrer durante a interação com o usuário.

6. Qual a diferença entre uma exceção (subclasses de Exception) e um erro (subclasses de Error)?

Exercícios de Coleções

7. Refaça os exercícios 13 e 14 da lista 1 usando List no lugar de vetor.

8. Refaça o exercício 17 da lista 1 usando Map para guardar a tradução dos números decimais para romanos.

9. Refaça o exercício 1 desta lista usando Set para guardar o conjunto de filhos de uma pessoa no lugar dos seus pais.

Exercícios de Threads

10. Faça um programa que leia um número “n” informado pelo usuário e diga quantos números primos há entre 0 e “n”. Esse seu programa deve rodar em duas threads, de forma que o esforço computacional seja dividido entre as threads.

11. Faça um programa que forneça a série de Fibonacci para um número *n* informado pelo usuário. Esse programa deve rodar em duas threads, de forma que o esforço computacional seja dividido entre as threads.

12. Qual a diferença do efeito produzido pelos programas abaixo? Qual deles é mais eficiente, assumindo que o computador tem mais de um processador?

Programa A

```
Thread[] threads = new Thread[10];
for (int i = 0; i < threads.length; i++) {
    threads[i] = new Thread(new MeuRunnable());
    threads[i].start();
}

for (int i = 0; i < threads.length; i++) {
    threads[i].join();
}
```

Programa B

```
Thread[] threads = new Thread[10];
for (int i = 0; i < threads.length; i++) {
    threads[i] = new Thread(new MeuRunnable());
    threads[i].start();
    threads[i].join();
}
```

13. Para que serve o modificador synchronized? Em que situações ele deve ser usado? Por que não usar em todos os métodos do programa?

14. Qual a diferença entre o método sleep() e o método join() da classe Thread?