Computação Orientada a Objetos
Prova 2 - Turma 04
Prof. Flávio Coutinho

Questão 1. Apesar da diversidade de pardões de projetos existentes e da diversidade de cenários nos quais cada um deles pode ser aplicado, é possível notar que algumas características são comuns a mais de um padrão. Vários padrões, por exemplo, s ugerem reestruturação no código de modo a diminuir o grau de acoplamento entre as diversas classes que fazem parte de um projeto (ou seja, propõe novas estruturas de classes que "enfraquecem as ligações" existentes entre as classes do projeto). Explique como esta redução de acoplamento se manifesta nos seguintes padrões:

a) (1 ponto) Facade

b) (1 ponto) Strategy

c) (1 ponto) Iterator

Questão 2. Embora o padrão Singleton seja controverso e sua real necessidade de aplicação costume ser questionada, ele é um padrão que, de certa forma, tira das "classes usuários" (isto é, aquelas que usam a classe que implementa o padrão) uma determinada responsabilidade. Com base nisto, responda:

a) (1 ponto) Qual é esta responsabilidade que é tirada das "classes usuárias"?

b) (1 ponto) Como esta remoção de responsabilidade é implementada pelo padrão?

Questão 3. (3 pontos) Aplique o padrão Strategy ao código **TesteOrdenador.java** (cuja listagem encontra-se no final da prova). Algumas dicas: (i) pense em como o código atual (em especial a classe **Ordenador**) evoluiria caso diversos novos critérios de comparação fossem adicionados (por exemplo: crescente por valor absoluto, decrescente por valor absoluto, distância crescente em relação a um valor referência, etc) para entender as limitações da estrutura atual do código; (ii) a correta aplicação do padrão deve resultar em um código que não possua longas cadeias de ifs/elses (novamente, pense em como o código atual evoluiria caso novos critérios fossem implementados); (iii) a correta aplicação do padrão também deve resultar em um código sem qualquer redundância.

Questão 4. (2 pontos) Escreva o método estático **junta_arquivos** que concatena o conteúdo de **n** arquivos binários em um único arquivo destino. O método deve receber como parâmetros: o **prefixo** (String) do nome dos arquivos a serem concatenados; o valor de **n** (int); e nome do arquivo **destino** (String). Assuma que os nomes dos arquivos de entrada são definidos da seguinte forma: *prefixo* + "." + *i* onde i >= *1 e i* <= *n,* e que a concatenação deve-se dar em ordem crescente de numeração das partes (por exemplo, dado prefixo = "parte" e **n** = 3, os arquivos devem ser concatenados para formar o arquivo destino são "parte.1", "parte.2" e "parte.3", nesta ordem).

```java
//////// TesteOrdenador.java ////////

class Ordenador {

    public static final int CRESCENTE = 0;
    public static final int DECRESCENTE = 1;
    private int criterio;

    public Ordenador(int c) {
        if (c != CRESCENTE && c != DECRESCENTE) throw new RuntimeException("Critério inválido");
        this.criterio = c;
    }

    public void ordena(int[] a) {
        for (int i = 0; i < a.length; i++) {

            int index = i;

            for (int j = i + 1; j < a.length; j++) {

                if (criterio == CRESCENTE) {
                    if (a[j] < a[index]) index = j;
                } else if (criterio == DECRESCENTE) {
                    if (a[j] > a[index]) index = j;
                }
            }

            // trocando elementos das posições i e index
            int temp = a[i];
            a[i] = a[index];
            a[index] = temp;
        }
    }
}

public class TesteOrdenador {
    public static void print(int[] a) { /* imprime o conteúdo do array */ }

    public static void main(String[] args) {
        int[] array = {6, 4, 5, 1, 3, 7, 2};
        Ordenador o = new Ordenador(Ordenador.CRESCENTE);
        // Ordenador o = new Ordenador(Ordenador.DECRESCENTE);

        print(array);
        o.ordena(array);
        print(array);
    }
}
```