Segunda Prova de MAB 240 — Computação II

Fabio Mascarenhas

2 de Dezembro de 2016

A prova é individual e sem consulta. Responda as questões na folha de respostas, a lápis ou a caneta. Se tiver qualquer dúvida consulte o professor.

Nome: _____

DRE: _____

Questão:	1	2	3	4	5	6	Total
Pontos:	1	2	2	1	2	2	10
Nota:							

O padrão *observador* generaliza o conceito de tratamento de eventos, em que um ou mais objetos (os *observadores*) podem estar interessados em receber notificações de um objeto *sujeito*. Observadores e sujeitos implementam respectivamente o seguinte par de interfaces:

```
public interface Observador<T> {
  void mudou(T sujeito);
}
```

```
public interface Sujeito<T> {
  void adicionar(Observador<T> observador);
  void esquecer(Observador<T> observador);
  void notificar();
}
```

1. (1 ponto) Os tipos dos parâmetros de `adicionar` e `esquecer` podem ser tornados mais genéricos com o uso de coringas. Indique qual seria o tipo mais genérico para esses parâmetros.

2. (2 pontos) Crie uma classe *abstrata* e *parametrizada* `SujeitoAbstrato` que implementa `Sujeito`, e usa um `HashSet` para gerenciar o conjunto de observadores. Forneça implementações dos três métodos de `Sujeito`. A classe parametrizada `HashSet` contém métodos `add` e `remove` para adicionar ou remover um objeto do conjunto, e pode ser percorrida usando um *for* do mesmo jeito que uma lista.

3. (2 pontos) Crie uma classe `Contador` que implementa `Observador<Integer>` e cujas instâncias contêm um sujeito, um contador que começa com valor 0 e um número limite. Uma instância de `Contador` começa a observar o seu sujeito no momento em que é construída, adiciona o valor recebido pela notificação ao contador a cada notificação de mudança que recebe, e deixa de observar quando o contador passa o limite.

A interface `AcaoIO` dada abaixo representa ações que podem fazer entrada/saída usando as classes de entrada e saída da biblioteca padrão de Java, e por consequência podem lançar exceções do tipo `IOException`:

```
public interface AcaoIO {
  void executa() throws IOException;
}
```

4. (1 ponto) As exceções do tipo `IOException` são checadas ou não checadas? Justifique sua resposta.

5. (2 pontos) Crie uma classe chamada `ImprimeArquivo` implementa `AcaoIO` e recebe uma instância de `BufferedReader` representando um arquivo aberto, imprime todo o seu conteúdo na saída padrão, lendo cada linha do arquivo usando o método `readLine()` de `BufferedReader`. Esse método retorna a linha (uma `String`) ou `null` caso já tenha chegado ao final do arquivo. Garanta que o arquivo será fechado usaodo o método `close()` de `BufferedReader` mesmo que erros aconteçam.

6. (2 pontos) Implemente uma função chamada `public static IOException executa(AcaoIO acao)` que executa a ação recebida e retorna o erro em caso de erros de IO, ou `null` se não acontecerem erros. Qualquer outro erro deve ser passado adiante.

BOA SORTE!