Primeira Prova de MAB 240 2012.2 — Computação II

Fabio Mascarenhas

21 de Dezembro de 2012

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	Total
Pontos:	5	5	10
Nota:			

1. Um jogo pode estar em diferentes *estados*, e em cada estado ele se comporta de uma maneira específica. Os jogos mais simples possuem dois estados: o estado normal do jogo e o estado de jogo terminado ("Game Over"), no qual normalmente o jogo não responde mais a entrada do usuário, e as animações podem ser interrompidas. Jogos mais complexos podem ter outros estados, correspondendo a diferentes fases do jogo.

   A maneira como foi implementado o estado de jogo terminado no Space Invaders feito em sala e no Frogger feito nos laboratórios foi usando um *if* em cada método que controla o funcionamento do jogo, mas usando a interface `Jogo` pode-se fazer um esquema mais elegante de gerenciamento desses estados.

   ```
   public interface Jogo {
     void tecla(String tecla);
     void tique(Set<String> teclas, double dt);
     void desenhar(Tela tela);
   }
   ```

   A ideia é representar cada estado como uma implementação de `Jogo`, e fazer a implementação principal de `Jogo` coordenar esses estados, além de conter uma lista com todos os objetos do jogo.

   ```
   public class MeuJogo implements Jogo {
     private Jogo[] estados;
     private int estadoAtual;
     private List<ObjetoJogo> objetos; // objetos do jogo

     ... // outros campos, construtor, outos métodos
   ```

```
  public void tecla(String tecla) {
    // implementação
  }

  public tique(Set<String> teclas, double dt) {
    // implementação
  }

  public void desenhar(Tela tela) {
    // implementação
  }

  public void mudaEstado(int estado) {
    // implementação
  }
}
```

(a) (2 pontos) Implemente os métodos `tecla`, `tique` e `desenhar` da classe `MeuJogo`.

(b) (1 ponto) Implemente o método `mudaEstado`.

(c) (2 pontos) Suponha que a interface `ObjetoJogo` contém um método `void desenhar(Tela t)`. Implemente a classe `GameOver`, que implementa `Jogo` e apenas desenha todos os objetos no campo `objetos` de sua instância de `MeuJogo`, junto com uma mensagem "GAME OVER". A classe `GameOver` não faz nada nos tiques do relógio ou nas teclas. Para mostrar a mensagem, use o método `texto` de `Tela`:

```
public void texto(String texto, int x, int y, int tamanho, Cor cor);
```

Para a classe `GameOver` funcionar podem ser necessárias adições na classe `MeuJogo`. Nesse caso diga também o que foi acrescentado ou alterado em `MeuJogo`.

2. 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:

```
interface Observador {
  void evento(Sujeito fonte);
}
```

```
interface Sujeito {
  void observar(Observador o);
  void esquecer(Observador o);
  void notificar();
}
```

(a) (2 pontos) Crie uma classe `ObservaImprimeN` que implementa `Observador`, cujas instâncias contêm um sujeito e um contador, começam a observar o seu sujeito no momento em que são construídos, e imprimem "EVENTO RECEBIDO" (usando `System.out.println`) a cada notificação, decrementando o contador e deixando de observar o sujeito quando o contador chega a 0.

(b) (3 pontos) Modifique a classe `MeuJogo` da questão 1 para implementar também a interface `Sujeito`. Seus observadores devem receber uma notificação a cada mudança de estado. Não precisa reescrever toda a classe, escreva na resposta apenas as linhas que são acrescentadas ou modificadas. Dica: use um `HashSet<Observador>` para gerenciar o conjunto de observadores, essa classe 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.

BOA SORTE E BOAS FESTAS!