Segunda Prova de MAB 240 — Computação II

Fabio Mascarenhas

27 de Junho 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:	6	4	10
Nota:			

1. Um programa editor gráfico usa a seguinte interface para representar formas geométricas, que dá as ações que o usuário pode fazer para manipular as formas no editor:

```
interface Forma {
  // Mover a forma, dado o deslocamento
  void mover(int dx, int dy);
  // Redimensiona a forma por um fator de escala
  void redimensionar(float escala);
  // Rotaciona a forma em tantos graus
  void rotacionr(int graus);
}
```

Um dos requisitos do editor é oferecer *undo* (desfazer) e *redo* (refazer) de vários níveis para o usuário, onde ele pode desfazer as últimas ações tomadas, ou refazer ações desfeitas. Um jeito de fazer isso é representar as ações do usuário como objetos que implementam a seguinte interface:

```
interface Acao {
  void fazer();
  void desfazer();
}
```

(a) (3 pontos) Defina classes para cada uma das três ações que o usuário pode fazer com uma forma: mover, redimensionar, rotacionar.

(b) (3 pontos) Complete o fragmento classe `Editor` abaixo (implementando os métodos marcados com `TODO`), que implementa parte do recurso desfazer/refazer do editor gráfico:

```
class Editor {
  List<Acao> acoes;
  // ultima ação feita
  int cursor;

  void fazer(Acao a) {
    // TODO: faz uma ação nova
  }

  void desfazer() {
    // TODO: desfaz última ação,
    // caso possível
  }

  void refazer() {
    // TODO: refaz última ação,
    // caso possível
  }
}
```

Por exemplo, se o usuário faz uma ação de mover, uma de rotacionar, uma de redimensionar, desfaz duas vezes, faz uma ação de mover e depois refaz uma vez a lista de ações deve ser mover, mover, rotacionar, redimensionar, e o cursor deve estar em 2, pois a última ação feita foi a rotação (no refazer). Dica: o método `void add(int i, Acao a)` de `List<Acao>` insere uma ação na lista na posição `i`.

2. (4 pontos) Seja a classe abaixo que representa uma lista de inteiros (a representação interna da lista, assim como a implementação dos seus métodos, foi omitida, mas assuma que ela existe e está correta):

```
class Lista {
  ...
  public Lista() {
    ...
  }
  public void adiciona(int num) {
    ...
  }
  public boolean contem(int num) {
    ...
  }
  public void remove(int num) {
    ...
  }
}
```

Faça duas implementações da classe `Conjunto`, que representa conjuntos de inteiros (lembre que um conjunto não possui elementos repetidos). Tentar adicionar um número que já existe a um conjunto deve causar uma `RuntimeException` com a mensagem *"elemento já existe"*. A primeira implementação deve ser derivada de `Lista`, e a segunda implementação não deve derivar de nenhuma classe, mas deve usar uma instância de `Lista` como representação interna.