Programação Orientada pelos Objectos

Exame (época de recurso)
2007/07/12

Atenção: A fraude numa prova de avaliação, mesmo quando detectada após a prova, é punida com a reprovação na cadeira.

Cada uma das questões do primeiro grupo vale 5 pontos. Cada uma das questões 1 a 7 vale 10 pontos. As questões 8 e 9 valem 15 pontos cada. A questão 10 vale 50 pontos.

Primeiro grupo

Por favor, responda imediatamente ao questionário anexo. Note bem, o questionário será recolhido 15 minutos após o início da prova.

Segundo grupo

Os concursos de programação são geridos por um sofisticado sistema de software que se baseia na classe Competition, a qual tem três campos privados: um para registar as equipas (nome da equipa e número de cada aluno membro da equipa), outro para registar os concorrentes (número de aluno concorrente e nome da equipa a que pertence) e um terceiro para registar os pontos de cada equipa (nome da equipa e número de pontos já obtidos pela equipa). A classe tem funções para registar uma nova equipa no sistema, para registar um novo concorrente, para verificar se já existe uma equipa com o nome indicado, para verificar se já existe um concorrente com o número indicado, para obter os pontos de um concorrente, dado o número, para acrescentar um certo número de pontos a uma equipa indicada e para imprimir uma classificação ordenada dos concorrentes, pelo número de pontos (descendentemente), desempatando pelo número do concorrente, um concorrente por linha. Note que os pontos de cada concorrente são os pontos da equipa a que pertence. Eis a classe, em esquema:

```
public class Competition
{
 private HashMap<String, ArrayList<Integer>>
                teams;
 private HashMap<Integer, String> contestants;
 private HashMap<String, Integer> points;

 public Competition()
 {
  teams = new HashMap<...>(...);
  contestants = new HashMap<...>(...);
  points = new HashMap<...>(...);
 }

 public void registerTeam(String teamName)
 {
  teams.put
     (teamName, new ArrayList<Integer>());
  points.put(teamName, 0);
 }

 public boolean existsContestant(int number)
 {
  ...
 }

 public boolean existsTeam(String teamName)
 {
  ...
 }

 public void registerConstestant
    (int number, String teamName)
 {
  assert !existsContestant(number);
  assert existsTeam(teamName);
  ...
 }

 public void addPoints(String teamName, int x)
 {
  points.put(teamName, points.get(teamName)+x);
 }

 public int getPoints(int number)
 {
  ...
 }

 public void printRanking()
 {
  ...
 }

}
```

1. Por favor, complete a função registerContestant.
2. Por favor, complete a função getPoints.
3. Por favor, complete a função printRanking.
4. Por favor, programe a função testAddPoints, para realizar o teste unitário da função addPoints. Teste três casos interessantes.

Terceiro grupo

Um dos muitos estudos que estão a ser feitos por causa do novo aeroporto é sobre as bagagens despachadas. Há passageiros que só levam bagagem de mão e há outros que despacham uma ou várias malas, que viajarão no porão.

Pois bem: queremos calcular o *peso médio de carga despachada, por passageiro, para cada voo*. Por exemplo, suponha que há dois passageiros num certo voo: o primeiro despacha duas malas, uma com 10 quilos e outra com 6 quilos; o segundo despacha uma mala, com 19 quilos. Então, nesse voo, o peso médio de carga despachada por passageiro é 17.5 quilos.

Para calcular este indicador precisamos de duas classes: a classe `Passenger`, que a representa a carga despachada por um passageiro num voo, com informações sobre o número de voo (uma cadeia de caracteres), o nome do passageiro, o destino final do passageiro e um ArrayList com o peso de cada uma das malas despachadas e a classe `Luggage`, que processa os registos de tipo `Passenger` para calcular o indicador.

5. Por favor, programe em esquema a classe `Passenger`, indicando apenas os campos.
6. Por favor, programe um método estático `parsePassenger` que dada uma cadeia representando os valores dos campos de um objecto de tipo `Passenger` devolve um objecto novo, devidamente inicializado. Na cadeia, os valores estão separados por vírgulas. Por exemplo, “tp374,silva manuel,londres,10.4,12.2” representa o Sr. Manuel Silva, que viaja para Londres no voo tp374 com duas malas, uma que pesa 10.4 quilos e outra que pesa 12.2 quilos.
7. Por favor, programe a classe `Luggage` com um campo de tipo `ArrayList<Passenger>` e com um método para ler para esse vector um ficheiro de dados passado num argumento de tipo `Scanner`. Cada linha do ficheiro vem no formato explicado na questão 6.
8. Por favor, programe na classe `Luggage` um método para calcular o peso médio despachado pelos passageiros de um voo dado, cujo número de voo passa em argumento. Presuma que todos os *getters* da classe `Passenger` estão disponíveis. Requisito técnico: use um ciclo for-each.
9. Por favor, programe um método para calcular os nomes dos passageiros que viajam sem bagagem, com uma mala, com duas malas e com três ou mais malas. O resultado deve ser um ArrayList de ArrayLists. Na posição zero do resultado virá a lista de nomes dos passageiros sem bagagem, na posição 1, a lista dos que levam uma mala, na posição 2, a lista dos que levam duas malas e na posição 3, os restantes. Todas as quatro listas estarão por ordem alfabética.

Quarto grupo

Infelizmente o mar vai levando a areia na nossa Costa da Caparica e há cada vez menos espaço para a exploração balnear. Por exemplo, na praia da sereia, já só se consegue plantar uma fila de toldos. Alguns clientes gostam de escolher o toldo arbitrariamente, o que podem fazer se o toldo não estiver já ocupado por algum outro cliente que chegou mais cedo, mas outros, mais ciosos da sua privacidade, preferem o toldo mais isolado, isto é, um toldo que esteja livre mas que esteja o mais afastado que for possível de outros toldos ocupados. Claro que isto não garante nada, pois logo a seguir pode vir outro cliente que escolhe o toldo logo ao lado. Seja como for, o banheiro da praia da sereia instalou um sofisticado serviço informático e pretende agora um programa para apoiar os clientes na selecção do toldo mais isolado.

Os dados são o número de toldos e os números dos toldos já ocupados. Os toldos são numerados de norte para sul, começando em 1. O resultado é o número do toldo mais isolado.

A primeira linha do ficheiro de dados indica o número de toldos, *N*. Em cada uma das linhas seguintes, em número indeterminado mas maior ou igual a zero e menor ou igual a *N*, vem um número inteiro entre 1 e *N* (inclusive), sem repetições. O ficheiro de resultados contém apenas uma linha, na qual virá o número do toldo mais isolado. Em caso de empate, prefere-se o toldo de número mais baixo. Se a praia estiver cheia e não houver toldos livres, o programa escreve “0”, convencionalmente.

Exemplo de ficheiro de dados:

```
12
2
9
4
12
```

Respectivo ficheiro de resultados:

```
6
```

10. Por favor, escreva um programa para resolver o problema do toldo mais isolado.