LEI MCC IFG

Disciplina: Programação II (2003/04) Prova: Recurso (2004.07.12)

Esta prova tem a duração de **2 h e 30 m** e é **sem consulta**.

1. Pretende-se elaborar uma hierarquia de classes para estruturar a organização e funcionamento duma clínica de saúde. A clínica tem utentes, médicos e faz marcação de consultas. Mesmo sabendo que não pode definir todas as entidades do problema, vamos desenvolver algumas, considerando já definidas outras.

   (a) Estruture (mas não implemente!) um esquema de classes para representar *Utente* e *Medico*. Considere que o utente, além das características comuns de pessoa (nome, morada, etc), tem um número de beneficiário correspondente a um tipo de assistência (Caixa, ADSE, SAMS, etc), e o médico uma especialidade e o número da cédula profissional.

   (b) Considere definida a classe *Data*, com a API da figura 1. Pretende-se definir a classe *Horario*, para que além duma data possa também referir uma hora, entre um valor inicial, por exemplo 9, e um valor final, por exemplo 20.

      i. Defina o cabeçalho e as variáveis de instância e/ou de classe, da classe *Horario*. Esta classe implementa as mesmas interfaces que a superclasse.

      ii. Defina construtores para esta classe;

      iii. Defina o método *seguinte* que devolve o horário seguinte, considerando como seguinte um horário cuja hora é próxima hora desde que validada pelas horas iniciais e finais de *Horario* e excluindo uma hora de almoço. A hora de almoço deve também estar definida na classe. Exemplos: Um horário seguinte às 11 horas é um horário no mesmo dia às 12 horas. Um horário seguinte às 12 horas é um horário no mesmo dia às 14, se o almoço for às 13. Um horário seguinte às 19 é às 9 do dia seguinte se a última hora do dia é 19, e a primeira é 9.

      iv. Os métodos das interfaces;

   (c) Defina a classe *Consulta*, considerando que cada consulta é caracterizada pelo utente, pelo médico e o horário consulta. Considere previamente definidas as classes *Medico* e *Utente*:

      i. o cabeçalho, as variáveis de instância e/ou de classe.

      ii. um construtor (ou mais, caso considere absolutamente necessário);

      iii. métodos selectores e modificadores para:

         A. o médico que dá a consulta;

         B. o doente ;

         C. o horário da consulta;

   (d) Considere que para organizar a clínica são usadas 3 listas, uma de utentes, uma de médicos e uma de consultas. Defina a classe *Clinica* definindo:

      i. o cabeçalho, as variáveis de instância e/ou de classe, da classe.

      ii. Implemente os seguintes métodos:

         A. `livre(Medico m, Horario h)`, que devolve true, se o médico m, não tem consultas na hora h;

         B. `marcacao(Utente u, Medico m, Data d)`, faz uma marcação de consulta para o médico x, do utente u, para o dia d (caso seja possível) ;

         C. `consultas(Medico m, Data d)`, devolve uma Enumeration com as consultas do médico m, para o dia d;

         D. `listaConsultas(Medico m, Data d)`, faz uma listagem das consultas do médico m no dia d .

```
Public class Data {
  public Data(); // uma data correspondente à data corrente
  public Data(int d, int m, int a);// uma data correspondente a dd/mm/aa
  public int dia(); // o dia da data
  public int mes(); // o mes da data
  public int ano(); // o ano da data
  public boolean diaUtil(); // devolve true se é dia útil
  public void  addDias(int d); adiciona d dias à data
  public String toString(); // a string ''dd/mm/aa''
  public Object clone(); // o usual
  public int compareTo(Object o); // o usual
  public boolean equals(Object o); // o usual
}
```

Figure 1: API da classe Data

2. Considere que pretende implementar a(s) classe(s) necessárias para representar e jogar o conhecido jogo da "Forca". No jogo da forca a finalidade é adivinhar uma palavra secreta, antes de se ser "enforcado". Da palavra secreta conhecem-se as letras que já se adivinharam e a sua posição na palavra. Em cada jogada o adversário tenta adivinhar uma letra da palavra. Se a letra existe é indicada na palavra a sua posição, caso contrário o adversário aumenta a sua forca com mais uma peça. As peças que constituem a forca são cabeça, pescoço, braço direito, braço esquerdo, mãos direita e esquerda, tronco, pernas direita e esquerda e pés direito e esquerdo, sendo portanto 10 as tentativas de erro possíveis para adivinhar a palavra. Implemente a classe *Forca*, definindo:

   (a) o cabeçalho e as variáveis de instância/classe necessárias para implementar o jogo;

   (b) um construtor para a classe que permita iniciar um jogo cuja palavra secreta é a string passada por parâmetro;

   (c) Implemente para a classe os métodos:

      i. `palavra()` que devolve uma string correspondente à palavra adivinhada até ao momento, substituindo as letras desconhecidas por "X";

      ii. `jogada(char x)`, que efectuada a jogada correspondente à letra x.

      iii. `estaTudo()`, que devolve true se todas as letras da palavra foram descobertas;

      iv. `penalizacao()`, que escreve no output a penalização do jogador na jogada, tipo "perdeu a cabeça", "perdeu o braço esquerdo", etc;

      v. `jogar()`, que permite jogar um jogo da forca, até o jogador perder ou adivinhar a palavra. Escreva " P E R D E U !!!!!!!" quando o jogador perde e " G A N H O O O O O U U U !!!!!!!" quando o jogar ganha. Leia iterativamente um caracter do standard input e analise a jogada.

```
String(c) // construtor cujo parâmetro é um array de caracteres
length() // devolve o comprimento da string
charAt(i) // devolve o carater na posição i
toCharArray() // devolve um array de caracteres correspondente à String
```

Figure 2: Alguma da API da classe String que possa necessitar