Trabalho 03

Seu professor de programação orientada a objetos teve uma ideia para melhorar o desempenho dos alunos na disciplina. Ela consiste na elaboração de exercícios baseados em perguntas e respostas (estilo *quiz*), permitindo a criação de diferentes sessões de treinamento. Para tal, ele pediu que você desenvolva um programa para gerenciar estas sessões (QuizApp).

O programa deve considerar três variedades de pergunta:

- VF - Verdadeiro/Falso
- ME - Múltipla escolha (número variável de escolhas)
- RC - Resposta curta (todas as respostas são uma palavra simples)

O programa será para um único jogador (usuário) e, inicialmente, perguntará ao jogador para que ele entre com seu primeiro e último nome, seguido pelo nome do arquivo que armazena as questões e respostas da sua sessão de treinamento. Então, seu programa deverá perguntar ao jogador quantas questões ele gostaria de praticar. Apresente como valor default, o número total de questões definidas no arquivo indicado. Implemente um tratamento adequado quando o jogador escolher uma quantidade inválida e/ou quando o arquivo possuir alguma informação inválida (estrutura fora do padrão definido). Após a escolha, o programa deverá mostrar uma sequência de perguntas (limitada ao número de questões escolhido). As perguntas devem ser mostradas aleatoriamente (não devem se repetir) e não na ordem sequencial que elas estão no arquivo. Depois da apresentação da pergunta, o jogador terá duas opções:

a) Informar uma resposta válida.
b) Digitar PASSAR. Neste caso, nenhuma pergunta de resposta curta deverá ter como resposta a palavra PASSAR.

Caso o jogador passe a questão, ele não receberá nem perderá pontos. Caso ele responda à questão corretamente, ele receberá o número de pontos definido pela questão. Caso a resposta seja incorreta, ele perderá o mesmo número de pontos.

Você deve manter informações sobre o histórico de sessões de treinamento realizadas pelo jogador em um arquivo específico para aquele jogador. O programa deve mostrar todas as sessões de treinamento executadas, indicando número total de perguntas selecionadas, número de perguntas respondidas, número de respostas corretas e incorretas, bem como a pontuação obtida em cada sessão e no geral. Considere que inicialmente, um jogador iniciará com 0 pontos. O sistema não deve permitir que um jogador jogue duas vezes uma mesma sessão.

Formato do arquivo de questões

A primeira linha contém o nome da sessão de treinamento (que indica o tema das questões). A segunda linha contém um único número inteiro N que representa o número de perguntas no arquivo.

Cada uma das N perguntas possui uma sequência de linhas com informações específicas.

A primeira linha de cada pergunta contém uma string que indica o tipo de pergunta seguido por um número inteiro positivo que indica o valor da questão. O tipo da questão deve ser obrigatoriamente uma das três variedades definidas: "VF", "ME" ou "RC".

Trabalho 03

A segunda linha da pergunta contém uma string que representa a questão. Como a questão é uma frase, o programa deve considerar que existem espaços entre as palavras.

Se a questão é do tipo Verdadeiro/Falso, a terceira linha contém uma string com o valor "V" ou "F", dependendo da resposta para a pergunta.

Se a pergunta é do tipo resposta curta, então a terceira linha do contém uma única sequência de caracteres sem espaços, indicando a resposta para a pergunta.

Se a questão é do tipo múltipla escolha, a terceira linha contém um único número inteiro positivo, K (menor do que 10), indicando o número de opções para a pergunta. As linhas seguintes conterão cada uma das respostas possíveis. Estas respostas são frases (portanto, podem ter espaços) e correspondem às escolhas A, B, C, etc. A última linha de uma questão de múltipla escolha irá conter uma string com a resposta correta. Esta string será uma letra maiúscula correspondente à opção de resposta correta ("A", "B", "C", "D", "E", "F", "G", "H", ou "I", dependendo da quantidade de escolhas).

Exemplo para o arquivo de entrada: Exemplo.txt

```
POO - Definições básicas
3
VF 5
Um objeto é a instância de uma classe. (V/F)
V
ME 10
Dizer que a classe A especializa a classe B é o mesmo que dizer que:
5
a classe B é subclasse de A
a classe A é superclasse de B
a classe A é subclasse de B
a classe B é derivada de A
a classe B é abstrata
C
RC 20
Uma classe que não pode instanciar objetos é uma classe?
Abstrata
```

Exemplo de execução do programa (respostas do jogador estão em negrito)

```
Qual seu primeiro nome?
Marcello
Qual seu último nome?
Thiry
Informe o nome do arquivo com a sessão de treinamento?
Exemplo.txt
Quantas questões você gostaria de responder (1 a 3)?
4
Desculpe, mas você deve escolher entre 1 e 3 perguntas.
Quantas questões você gostaria de responder (1 a 3)?
Duas
Desculpe, mas você deve informar um número válido (1 a 3).
Quantas questões você gostaria de responder (1 a 3)?
3
```

Trabalho 03

```
Pontos: 10
Questão: Dizer que a classe A especializa a classe B é o mesmo que dizer
que:
A) a classe B é subclasse de A
B) a classe A é superclasse de B
C) a classe A é subclasse de B
D) a classe B é derivada de A
E) a classe B é abstrata
C

Correto! Você ganhou 10 pontos.


Pontos: 20
Questão: Uma classe que não pode instanciar objetos é uma classe:
Concreta

Incorreto. A resposta correta é Abstrata. Você perdeu 20 pontos.


Pontos: 5
Questão: Um objeto é a instância de uma classe. (V/F)
PASSAR

Você optou por pular esta questão.

Fim de jogo!
Sua pontuação final foi -10 pontos.
```

Orientações

1. Este trabalho não é apenas de implementação. Você deve utilizar algum tempo para projetar as classes que você precisará e quais operações serão necessárias. Não tente escrever o código diretamente. Você deve entregar também a modelagem UML completa da solução. A modelagem deve ser feita na ferramenta Bouml.
2. Depois de decidir quais classes você precisa, implemente uma de cada vez (considerando as classes simples primeiro). Realize testes para assegurar que ela está funcionando antes de prosseguir para a próxima classe. Eventualmente, você perceberá a necessidade de ajustes no seu design e não será necessário modicar todo o código. Siga a ideia: projete um pouco, codifique um pouco, teste um pouco. Repita.
3. Escreva um código legível e utilize comentários (javadoc) para facilitar o entendimento. Lembre-se que o código deve ser de todos. A qualidade e pertinência dos comentários será considerada como critério de avaliação.
4. Sua solução deve obrigatoriamente utilizar generalização (herança) ou interface (caso a superclasse seja abstrata com todas as operações abstratas).
5. Utilize a classe Scanner para ler dados do teclado. Veja a documentação disponibilizada no material didático. Além disso, você deve criar uma classe que utilize a classe Scanner com métodos específicos de entrada de dados para facilitar a sua programação.
6. O programa deve obrigatoriamente utilizar exceções. A forma adequada deste uso deve seguir as boas práticas discutidas na disciplina.
7. As perguntas devem ser carregadas em uma estrutura de classes adequada (os objetos devem ser criados no momento da leitura do arquivo de perguntas). Este é um ponto relevante do trabalho, pois não será considerada uma implementação que apenas lê as

Trabalho 03

strings do arquivo de perguntas e as apresenta para o jogador. Você deve ter classes específicas para cada tipo de pergunta.

8. Observe os princípios da orientação a objetos. Classes e operações com baixa coesão, ausência de comentários javadoc, métodos muitos longos ou com muitos parâmetros afetarão negativamente a avaliação do seu trabalho.

Referência

http://www.cs.ucf.edu/~dmarino/ucf/transparency/cop3330/asgn/