MATA55 - PROGRAMAÇÃO ORIENTADA A OBJETOS

Aula 9: Objetos, construtores, sobrecarga, encapsulamento, membros estáticos

Prof. Felipe Fernandes

14 Setembro de 2022

1. O Professor Felipe precisa de um sistema para automatizar o diário das turmas do Instituto de Computação da UFBA. O sistema precisa registrar:

   - Os alunos: cada aluno tem um nome, matrícula e data de nascimento.
   - As turmas: apresentam um código e o nome de uma disciplina.
   - Os dias-aula: a data e o horário da aula, turma a que está associada, assim como o conteúdo ministrado.
   - Os registros de frequência: correspondem a uma associação entre um aluno específico e um dia-aula específico. Por exemplo, a tupla (Marina, 14/09/2022 18h30-20h20) identifica o registro da aluna Marina na aula do 14/09/2022 18h30-20h20. Cada registro possui um campo informando se o aluno estava presente ou ausente nessa aula.
   - Registro de notas: Cada aluno A se associa a uma turma T. Essa associação precisa registrar a nota do aluno A na turma T.

   O sistema precisa apresentar um menu com as seguintes opções.

   (a) Registrar um aluno.
   (b) Registrar uma turma.
   (c) Registrar um dia-aula.
   (d) Registrar uma nota.
   (e) Registrar uma frequência.
   (f) Dado um dia aula e uma turma, identificar os alunos presentes.
   (g) Dado um aluno e uma turma, mostrar a quantidade de presenças e de faltas.
   (h) Dado um aluno e uma turma, mostrar sua nota.

(i) Listar os alunos inscritos em uma turma.

(j) Listar as turmas.

(k) Listar o total de faltas e presença por aluno, para uma dada turma.

(l) Listar notas por aluno para uma dada turma.

(m) Sair do sistema.

O usuário escolhe uma opção do menu. Em seguida, entra com os dados necessários para a opção escolhida. O sistema executa, exibe o resultado e volta a exibir o menu. O processo continua até que o usuário escolha sair do sistema. Sua tarefa consiste em modelar e implementar este simples sistema utilizando Orientação a Objeto em Java.

Atenção: Utilizar conceitos de encapsulamento. Não é requisito deste exercício trabalhar com nenhum tipo de persistência de dados, ou escrita de arquivos.

2. Implemente um sistema de correção de provas de múltiplas escolhas. Seu sistema deve ter duas classes. A classe **Gabarito** representa o gabarito da avaliação. Esta classe possui um método chamado **respostaQuestao** que recebe o número da questão e retorna a alternativa correta. A classe **Prova** representa uma prova feita por um aluno. Esta prova possui 15 questões de múltipla escolha (letras A a E). As 10 primeiras questões valem 0,5 ponto e as 5 últimas questões valem 1 ponto. A classe **Prova** deverá controlar as questões respondidas pelo aluno. Para isto, ela deve implementar os métodos:

   - **construtor**: recebe como parâmetro um objeto da classe Gabarito contendo o gabarito da prova
   - **respostaAluno**: recebe como parâmetro a resposta dada pelo aluno a uma questão; este método não recebe entre os parâmetros o número da questão, ele mesmo deve estabelecer um controle interno de modo que as questões sejam inseridas sequencialmente, ou seja, a primeira vez que o método é chamado, insere a primeira questão, a segunda, insere a segunda questão, e assim por diante.
   - **acertos**: retorna a quantidade de questões que o aluno acertou.
   - **nota**: retorna a nota que o aluno tirou na prova.
   - **maior** recebe como parâmetro um outro objeto da classe Prova e retorna a nota do aluno que acertou mais questões; se houver empate, retorna a maior nota; se houver empate novamente, retorna a nota do aluno representado no objeto corrente

   Atenção: Utilizar conceitos de encapsulamento.

3. A linguagem Java dispõe de um suporte nativo a vetores, que exige a definição de seu tamanho no momento da instanciação. Depois de instanciado, o tamanho do vetor não pode ser modificado. Escreva uma classe

chamada **Vetor** cujos objetos simulem vetores de tamanho variável. A classe define os seguintes métodos:

- **construtor**: recebe como parâmetro o tamanho inicial do vetor.
- **insert**: recebe como parâmetro uma string e a coloca na próxima posição disponível do vetor; note que o vetor cresce automaticamente, portanto, se a inserção ultrapassar o tamanho inicial estabelecido na criação, por exemplo, o vetor deve aumentar seu tamanho automaticamente.
- **get**: recebe como parâmetro uma posição do vetor e retorna a string que estiver naquela posição; se a posição não estiver ocupada ou ultrapassar o tamanho do vetor, este método retorna nulo.
- **size** retorna o número de elementos inseridos no vetor (independente do tamanho do mesmo).

Atenção: O java dispõe de classes – tal como a Vector – que realizam a tarefa solicitada nesta questão. Tais classes não devem ser utilizadas. É possível resolver esta questão apenas usando o sistema de vetores preexistente do Java procedural, sem nenhum comando especial extra. Atenção: Utilizar conceitos de encapsulamento.

4. Crie uma classe **Produto** para representar um produto do mundo real. Sua classe deverá conter os seguintes atributos e métodos:

   - Um campo de dados privado do tipo String chamado **nome**, que representará o nome do produto.
   - Um campo de dados privado do tipo double chamado **precoCusto**, que guardará o preço de custo do produto.
   - Um campo de dados privado do tipo double chamado **precoVenda**, que guardará o preço de venda do produto.
   - Um campo de dados privado do tipo double chamado **margemLucro**, que guardará a margem de lucro do produto.
   - Métodos públicos ***get()*** e ***set()*** para os atributos acima. Modifique o método **setPrecoVenda()** para que o preço de venda não seja inferior ao preço de compra. Caso isso aconteça, exiba uma mensagem alertando o usuário.
   - Crie um método chamado **calcularMargemLucro()** que calculará a margem de lucro do produto.
   - Crie um método chamado **getMargemLucroPorcentagem()** que retornará a margem de lucro como percentual.

   Para finalizar, no método **main()** da classe de teste, crie um novo objeto da classe **Produto**, peça para o usuário informar os preços de custo e de venda e exiba a margem de lucro em moeda e em percentual.

5. Implemente uma classe **Ponto**. Essa classe deve ter dois atributos, x e y, que representa suas coordenadas. Implemente três construtores para a classe:

   (a) Construtor sem parâmetros, que cria um ponto nas coordenadas (0,0);

   (b) Construtor que recebe dois parâmetros de coordenadas x e y;

   (c) Construtor que inicializa o ponto através das coordenadas de um outro Ponto recebido como argumento.

   Escreva um método **desloca**, que recebe uma coordenada *(dx,dy)* desloca o **Ponto** *(x,y)* para a nova coordenada *(x+dx, y+dy)*. Escreva duas versões do método **desloca**: a primeira recebe dois inteiros representando as respectivas coordenadas; a segunda recebe um objeto da classe **Ponto**. Escreva também um método *toString* que imprime o ponto. Implemente todos os métodos necessários para o encapsulamento. Crie dois métodos **setPonto**: um recebe a coordenada *(x,y)* e o outro recebe um objeto da classe **Ponto**.

6. Implemente uma classe chamada **Singleton**. Essa classe só pode ser instanciada uma única vez. Quando alguém necessita de uma instância de **Singleton**, precisa invocar o método estático **getInstance()**, o qual retorna a (única) instância desta classe. Esta instância deve ser criada na primeira vez que **getInstance()** é invocado. A figura 1 ilustra esta classe.

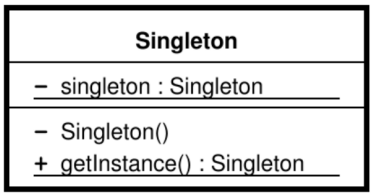

Figure 1

7. Escreva uma classe **SerieLimitada**, que encapsula um valor inteiro sequencial, com no máximo 5 dígitos, como os usados em notas e séries de produtos. Essa classe deve permitir que um programa crie um número limitado de instâncias dela, cada uma numerada com um valor sequencial. O número total de instâncias é controlado pelo campo **maximoDeInstancias**, declarado como *static final*, e o de instâncias já criadas é controlado pelo campo **contador** declarado como *static*. Escreva também uma aplicação que crie algumas instâncias da classe para demonstrar seu funcionamento. Dica: para gerar o número de série aleatório, utilize a classe **Random**, definida no pacote **java.util**.

8. O objetivo desse exercício é criar uma enumeração de objetos. Uma enumeração pode ser definida como uma classe **C**, onde cada campo é

declarado como *static final* e consiste em uma instância de **C**. A classe **C** não pode ser instanciada por outra classe. Se uma outra classe **X** precisar de uma instância de **C**, então **X** deve invocar um dos campos enumerados na classe **C**. Instâncias da classe **C** apresentam, pelo menos, um atributo chamado *descrição*, do tipo **String**, o qual armazena uma descrição para aquela instância. Implemente uma enumeração para os meses do ano. Em seguida, crie uma pequena agenda de compromissos, onde cada compromisso tem o mês e a descrição do evento.