Programação Orientada a Objetos

Tarefa 3

Prof. Dirson

Questão 01

a) **Classes concretas, abstratas e interface do Diagrama de Classe:** As classes concretas presentes no diagrama são ProfessorDaEducacaoBasica e ProfessorUniversitario, que estendem a classe abstrata Professor. A interface FuncionarioAssalariado e a interface Estudante são os elementos abstratos no diagrama.

b) **Partes do Diagrama de Classe da interface FuncionarioAssalariado:** A interface FuncionarioAssalariado contém a definição de um método chamado receberSalario(int nTurmas). Este método não possui implementação, indicando que qualquer classe que implemente essa interface deve fornecer sua própria implementação para garantir um padrão comum no recebimento de salários.

c) **Partes do Diagrama de Classe da classe Aluno:** A classe Aluno possui atributos como nome (do tipo String) e nMatricula (do tipo int). Ela inclui um construtor que aceita esses atributos e métodos para realizar operações, como fazer prova (fazerProva(Prova prova): void), obter e definir o nome (getNome() e setNome(String nome)), além de obter e definir o número de matrícula (getnMatricula() e setnMatricula(int nMatricula)), e um método para estudar (estudar(): void).

d) **Código do dialeto XML da Classe Prova e sua localização no Diagrama:** O código da classe Prova está definido no diagrama e inclui atributos como questoes (um array de String), respostas (também um array de String), e nota (inicializada com 0 do tipo double). A classe contém um construtor que recebe um array de questões e métodos para obter e definir questões e respostas (getQuestoes(), setQuestoes(String[] questoes), getRespostas(), setRespostas(String[] respostas)), além de métodos para obter e definir a nota (getNota(), setNota(double nota)).

e) **Código do dialeto XML da seta que une uma classe a interface e a seta que determina uma relação de herança entre as classes:** As setas que unem classes a interfaces são representadas no XML por lt=<., enquanto as setas que determinam uma relação de herança entre classes são representadas por lt=<. Esses elementos mostram como as diferentes entidades do diagrama se inter-relacionam e estabelecem dependências dentro do sistema.

Questão 2

a) **Nome do método e parâmetros de entrada e saída:** O nome do método na interface Estudante é estudar(). Este método não possui parâmetros de entrada e não retorna um valor, o que é indicado pela ausência de um tipo de retorno (void).

b) **Por que o método da interface termina necessariamente com um ponto-e-vírgula (;)?** O método na interface termina com um ponto-e-vírgula porque, em Java, os métodos definidos em interfaces não têm implementação. O ponto-e-vírgula é utilizado para indicar o fim da declaração do método, uma vez que não há um corpo de método (bloco de código) associado a ele.

c) **Quem deve implementar o método desta interface?** O método estudar() da interface Estudante deve ser implementado por qualquer classe que declare que implementa essa interface. Isso significa que classes que representam entidades de estudantes, como Aluno, precisam fornecer uma implementação específica para o método estudar(), garantindo que cada classe que implementa a interface possa definir o comportamento do método de acordo com suas necessidades.

Questão 03

a) **Tipo de classe e requisitos:**  
A Classe Professor é uma classe concreta. Para que uma classe seja considerada concreta, ela deve conter a implementação de todos os seus métodos e não pode ser declarada como abstrata. Isso significa que a classe pode ser instanciada e utilizada diretamente no código.

b) **Ponto-e-vírgula no método:**  
O método da interface que a Classe Professor implementa termina com um ponto-e-vírgula porque, em Java, os métodos definidos em interfaces não têm corpo. O ponto-e-vírgula indica o fim da declaração do método, já que não existe uma implementação associada a ele.

c) **Implementação do método:**  
O método da interface deve ser implementado pela Classe Professor, que é responsável por fornecer a funcionalidade específica desse método. Assim, qualquer classe que implementa a interface deve oferecer uma implementação para o método definido.

d) **Anotação @Override:**  
A anotação @Override serve para indicar que um método na classe está sendo sobrescrito a partir de uma superclasse ou interface. Isso ajuda a identificar que a implementação é uma versão modificada de um método existente. Além disso, se houver um erro na assinatura do método (como um erro de digitação), o compilador irá gerar um aviso, ajudando na identificação de problemas.

Questão 04

a) **Método com o mesmo nome da classe:**  
O método com o mesmo nome da classe é um construtor. Esse tipo de método é chamado sempre que uma instância da classe é criada e é responsável por inicializar os atributos da nova instância. Cada linha do código-fonte desse método geralmente serve para atribuir valores iniciais aos atributos da classe, garantindo que a instância comece com um estado válido.

b) **Tipo de método e implementação de elaborarProvas():**  
O método elaborarProvas() é um método concreto que possui uma implementação específica. A implementação deste método pode variar entre as classes que o possuem, dependendo das necessidades de cada classe. Cada linha do código-fonte deste método define o comportamento específico de como a prova deve ser elaborada, como a definição de perguntas, formatação e critérios de avaliação.

Questão 05

a) **Objetos instanciados e classes de origem:**  
No código da classe TestaSolucao, os objetos instanciados são geralmente do tipo Professor, Aluno ou ProfessorDaEducacaoBasica, dependendo de como o código está estruturado. As classes de origem correspondem às classes que foram definidas anteriormente e que são utilizadas para criar os objetos.

b) **Métodos usados e sua implementação:**  
Os métodos usados na classe TestaSolucao geralmente incluem métodos como main, que é o ponto de entrada da aplicação, e outros métodos que podem ter sido definidos nas classes de origem, como elaborarProvas(). A implementação desses métodos ocorre nas respectivas classes, onde são definidos para realizar ações específicas relacionadas à lógica do programa.