Universidade Federal de Goiás
Instituto de Informática
POO – Laboratório Classe Abstrata e Interface (parte 2)

Aula de Exercícios (24/09/2024) – Individual ou Dupla

Envie os códigos-fontes em um único arquivo compactado (zipado) indicados e o *output* quando solicitado

Abre o arquivo **LabClasseAbstrataInterface.uxf** no VS (Visual Studio) Code.

Obs.: É necessário instalar a ferramenta **Umlet** no VS Code, se não tiver instalado em sua estação de trabalho, instale, se tiver dúvida verifique o laboratório anterior.

Instruções para a Submissão no SIGAA

Resolva as questões que não envolvem código em um único arquivo do tipo documento no formato compatível com o Microsoft Word o Libre office Write ou .pdf. Lembre-se de identificar-se apropriadamente para facilitar a correção, colocando números das questões. Também no conteúdo deste arquivo, coloque seu nome e número de matrícula. Em caso de fazer em dupla coloque a matrícula e nome de ambos os estudantes.

Para as questões que envolvem códigos, deve ser enviado:

- O pacote completo criado pelo Eclipse ou no Visual Studio Code OU;
- Uma pasta com todas as classes usadas.

Se houver mais de uma questão de código, lembre-se de separar em pastas ou identificar no nome dos arquivos a qual se refere. Crie o hábito de identificar o código, bem como seu nome e número de matrícula nos comentários antes das classes.

Junte todos os arquivos finais em um único **.zip,** de preferência com seu nome para submeter no SIGAA.

Parte Teórica

1) Sobre o Diagrama de Classe LabClasseAbstrataInterface.uxf **responda as seguintes questões?**

a) Quais são as classes concretas, abstratas e interface do Diagrama de Classe?

b) Descreva as partes do Diagrama de Classe da interface FuncionarioAssalariado?

c) Descreva as partes do Diagrama de Classe da classe Aluno?

d) Descreva resumidamente o código do dialeto XML da Classe Prova e onde se pode encontrá-lo no Diagrama?

Observação: Verifique o arquivo Umlet.pdf do laboratório anterior.

d) Descreva resumidamente o código do dialeto XML da seta que une uma classe a interface e a seta que determina uma relação de herança entre as classes?

Observação: Verifique o arquivo Umlet.pdf do laboratório anterior.

2) Sobre o código-fonte da interface Estudante **responda as seguintes questões?**

a) Qual é o nome do método e quais são os parâmetros de entrada e de saída deste método?

b) Porque o método da interface termina necessariamente com um ponto-e-vírgula (;)?

c) Quem deve implementar o método desta interface?

3) Sobre o código-fonte da Classe Professor **responda as seguintes questões?**

a) Que tipo de classe é a Classe Professor e o que é necessário e suficiente para uma classe ser deste tipo?

b) Porque o método da interface termina necessariamente com um ponto-e-vírgula (;)?

c) Quem deve implementar o método desta interface?

d) A classe apresenta uma anotação @Override para que serve esta anotação?

4) Sobre o código-fonte ProfessorDaEducacaoBasica **responda as seguintes questões?**

a) Existe um método com o mesmo nome da classe. O que faz este tipo de método? Explique cada linha do código-fonte deste método.

b) Que tipo de método é o método elaborarProvas()? A implementação dele é a mesma em todo as classes do projeto onde ele existe? Explique cada linha do código-fonte deste método.

5) Sobre o código-fonte TestaSolucao **responda as seguintes questões?**

a) Quais objetos são instanciados no código e de quais são suas classes de origem?

b) Quais os métodos usados nesta classe e em qual classe ele foi implementado?

Parte Prática

6) Implemente a interface FuncionarioAssalariado considerando o Diagrama dessa Classe descrito em LabClasseAbstrataInterface.uxf e considerando também os códigos fontes já implementados no Apêndice desta tarefa.

7) Implemente a classe ProfessorUniversitario considerando o Diagrama dessa Classe descrito em LabClasseAbstrataInterface.uxf e considerando também os códigos fontes já implementados no Apêndice desta tarefa.

8) Implemente a classe Aluno considerando o Diagrama dessa Classe descrito em LabClasseAbstrataInterface.uxf e considerando também os códigos fontes já implementados no Apêndice desta tarefa.

9) Implemente a classe Prova considerando o Diagrama dessa Classe descrito em LabClasseAbstrataInterface.uxf e considerando também os códigos fontes já implementados no Apêndice desta tarefa.

10) Após complementar o código fonte das classes solicitadas execute a classe de teste cujo código-fonte TestaSolucao.java? Tire um Print Screen da janela de execução.

Apêndice
Código-fonte de Algumas Classes do Projeto

Interface Estudante

```
package LabClasseInterface;
public interface Estudante {
	public void estudar();
}
```

Classe Professor

```
public abstract class Professor implements FuncionarioAssalariado, Estudante {
	private String nome;
	private int nTurmas;
	private double salario;

	public Professor(String nome, int nTurmas, double salario) {
		this.nome = nome;
		this.nTurmas = nTurmas;
		this.salario = salario;
	}


	public abstract Prova[] elaborarProvas();


	public double corrigirProva(Prova prova) {
		String[] respostas = prova.getRespostas();
		double nota = 0;

		for (int i = 0; i < respostas.length; i++) {
			if ( respostas[i].contentEquals("R" + (i + 1)) )
				nota += 10/respostas.length;
		}
		prova.setNota(nota);

		return prova.getNota();
	}

	/**
	 * @return o nome
	 */
	public String getNome() {
		return nome;
	}
	/**
	 * @param nome o nome a ser configurado
	 */
	public void setNome(String nome) {
```

```
            this.nome = nome;
      }
      /**
       * @return o nTurmas
       */
      public int getnTurmas() {
            return nTurmas;
      }
      /**
       * @param nTurmas o nTurmas a ser configurado
       */
      public void setnTurmas(int nTurmas) {
            this.nTurmas = nTurmas;
      }
      /**
       * @return o salario
       */
      public double getSalario() {
            return salario;
      }
      /**
       * @param salario o salario a ser configurado
       */
      private void setSalario(double salario) {
            this.salario = salario;
      }

      @Override
      public void estudar() {
            System.out.println("Professor também estuda!");
      }

      @Override
      public void receberSalario(int nTurmas) {
            this.setSalario(salario + (nTurmas * salario * 0.05));
      }
}
```

Classe ProfessorDaEducacaoBasica

```
package LabClasseInterface;

public class ProfessorDaEducacaoBasica extends Professor {

      public ProfessorDaEducacaoBasica(String nome, int nTurmas, double
salario) {
            super(nome, nTurmas, salario);
      }

@Override
public Prova[] elaborarProvas() {

Prova[] provas = new Prova[4];
```

```
provas[0] = new Prova(new String[]{ "P1Q1", "P1Q2", "P1Q3", "P1Q4", "P1Q5" });
provas[1] = new Prova(new String[]{ "P2Q1", "P2Q2", "P2Q3", "P2Q4", "P2Q5" });
provas[2] = new Prova(new String[]{ "P3Q1", "P3Q2", "P3Q3", "P3Q4", "P3Q5" });
provas[3] = new Prova(new String[]{ "P4Q1", "P4Q2", "P4Q3", "P4Q4", "P4Q5" });

     return provas;
 }

}
```

Classe TestaSolucao.java

```
package LabClasseInterface;

public class TestaSolucao {

  public static void main(String[] args) {
    Professor maria = new ProfessorUniversitario("Maria", 2, 3000);
    Professor jose = new ProfessorDaEducacaoBasica("Jose", 3, 2000);
    Aluno PrimeiroNomedoEstudante = new Aluno("PrimeiroNomedoEstudante",
12345);

   Estudante[] pessoasQueEstudam = new Estudante[3];
   pessoasQueEstudam[0] = maria;
   pessoasQueEstudam[1] = jose;
   pessoasQueEstudam[2] = PrimeiroNomedoEstudante;

      for (Estudante estudante: pessoasQueEstudam)
           estudante.estudar();

   Prova[] provasDaFaculdade = maria.elaborarProvas();

     PrimeiroNomedoEstudante.fazerProva(provasDaFaculdade[0]);
     maria.corrigirProva(provasDaFaculdade[0]);

      System.out.println("PrimeiroNomedoEstudante tirou " +
provasDaFaculdade[0].getNota() + " na prova");
    }
}
```