|  |  |  |  |
| --- | --- | --- | --- |
| **PLANO DE ENSINO** | | | |
| **CURSO** | | **MÓDULO** | **Componente Curricular Sigla** |
| Técnico em Desenvolvimento de Sistemas | | 1º | FPOO |
| **COMPONENTE CURRICULAR** | **AULAS PREVISTAS** | **DOCENTE** | **TURMA(S)** |
| Fundamentos De Programação Orientada A Objeto | 115 | Reenye e Wellington | 1DES |
| **UNIDADE DE COMPETÊNCIA** | **OBJETIVO** | | |
|  | Fundamentos de Programação tem como objetivo proporcionar a aquisição de fundamentos técnicos e científicos relativos à lógica de programação, como ferramenta para desenvolvimento de aplicações, utilizando linguagem orientada a objetos, bem como o desenvolvimento de capacidades sociais, organizativas e metodológicas adequadas a diferentes situações profissionais. | | |

|  |
| --- |
| **FUNDAMENTOS TÉCNICOS E CIENTÍFICOS** |
| 8. Utilizar vetores e matrizes na elaboração do programa (3)  9. Aplicar técnicas de código limpo (clean code)  10. Manipular os diferentes tipos de dados na elaboração de programas  11. Utilizar o ambiente integrado de desenvolvimento (IDE)  12. Utilizar técnicas de versionamento através de softwares específicos  13. Utilizar o paradigma da programação orientada a objetos  14. Elaborar diagramas de classe |

|  |
| --- |
| **CAPACIDADES SOCIAIS, ORGANIZATIVAS E METODOLÓGICAS.** |
| 3. Demonstrar objetividade na coleta de informações  5. Demonstrar visão sistêmica (28)  6. Seguir método de trabalho (21) |

|  |
| --- |
| **CONHECIMENTOS** |
| 3. Programação Orientada a Objetos  3.1. Definição  3.2. Pacotes  3.3. Classes  3.3.1. Abstrata  3.3.2. Interna  3.3.3. Anônima  3.3.4. Atributos  3.3.5. Métodos  3.3.6. Modificadores de acesso (encapsulamento)  3.4. Objetos  3.5. Interface  3.6. Polimorfismo  3.7. Enumerações  3.8. Relacionamentos de objetos  3.8.1. Herança  3.8.2. Agregação  3.8.3. Composição  4. Ambiente de desenvolvimento  4.1. Instalação e configuração  4.2. Gerenciamento de dependências  4.3. Recursos e interfaces  5. Manipulação de arquivos  5.1. Escrita  5.2. Leitura  6. Controle de versões  6.1. Definição  6.2. Repositório  6.2.1. Inicialização  6.2.2. Remoto  6.2.3. Ramificações (branch)  6.2.4. Versionamento |

|  |  |
| --- | --- |
| **ATIVIDADE** | **SITUAÇÃO DE APRENDIZAGEM - Formativa** |
| **TEXTO** | |
| **Obs:** Além da situação-problema a seguir, exercícios com outras temáticas, serão propostos a fim de desenvolver habilidades de abstração.  **Contextualização**: A escola SENAI de Jaguariúna necessita de um sistema de controle de acesso as suas dependências. Este sistema precisa classificar o mais detalhadamente possível todas as pessoas que entram e saem diariamente deste prédio.  **Desafio**: Como técnico em desenvolvimento de sistemas, você foi contratado para fazer parte de uma equipe que criará o sistema e o instalará em um computador localizado na portaria.  Os seguintes requisitos do sistema foram levantados pelo analista de sistemas:  - Classificar todos os visitantes de acordo com sua função;  - Cadastrar cada um conforme sua classificação;  - Registrar qual o motivo da visita a escola;  - Quantificar cada tipo de pessoa e calcular o total.   * Exemplos de tipos de pessoas que acessam o prédio conforme suas funções: Professores, Alunos, Visitantes, Representantes de Empresas, Pais de alunos. * Exemplo de motivos de visita: Ministrar aulas, realizar treinamento, Fazer matrícula, Realizar trabalhos administrativos, Fazer Visitas técnicas. * O analista escolheu a linguagem JavaScript e solicita que como técnico, membro da equipe, você crie as classes que serão necessárias para iniciar o desenvolvimento do sistema, conforme os requisitos apresentados e aplique as convenções corretas, já que outros integrantes da equipe desenvolverão outras partes do sistema.   **Entregas**:   * Apresente um diagrama de classes representando apenas os relacionamentos de herança. * Através da plataforma IDE Eclipse apresente o pacote contendo as classes criadas e seus respectivos atributos e métodos, colocando em prática as técnicas de abstração estudadas. * O código deverá apresentar a correta aplicação das técnicas de herança, agregação e/ou composição e poliformismo. | |

**INSTRUMENTO DE REGISTRO DE AVALIAÇÃO FORMATIVA**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Natureza dos Critérios** | **Fundamentos Técnicos e Científicos ou Capacidades Técnicas** | **Critérios de avaliação**   |  |  |  |  | | --- | --- | --- | --- | |  | Crítico |  | Desejável | | 0 | NÃO Atingiu | 1 | Atingiu | | F | Formativa | S | Somativa | | | **Alunos** | | | | | | | |
|  |  |  |  |  |  |  |  |
| **Competências Técnicas** | 10. Manipular os diferentes tipos de dados na elaboração de programas | Classificação e identificação de atributos e métodos bem com sua aplicabilidade em uma Classe. | F |  |  |  |  |  |  |  |  |
| 11. Utilizar o ambiente integrado de desenvolvimento (IDE) | Domínio das ferramentas básicas disponibilizadas pela IDE e dos recursos de melhoria de performance no desenvolvimento | F |  |  |  |  |  |  |  |  |
| 12. Utilizar técnicas de versionamento através de softwares específicos | Organização do trabalho, estruturação dos projetos de desenvolvimento de softwares, utilização adequada dos softwares específicos para versionamento. | F |  |  |  |  |  |  |  |  |
| 13. Utilizar o paradigma da programação orientada a objetos | Utilização adequada dos recursos de Encapsulamento, Herança, Poliformismo e outros, para o desenvolvimento das aplicações propostas. | F |  |  |  |  |  |  |  |  |
| 14. Elaborar diagramas de classe | Relacionamento dos conceitos de desenvolvimento e análise de diagramas de classe. | F |  |  |  |  |  |  |  |  |
| **Competências de Gestão** | 3. Demonstrar objetividade na coleta de informações | Classificação adequada de objetos, capacidade de abstrair atributos e métodos, aplicação das técnicas de código limpo. | F |  |  |  |  |  |  |  |  |
| 5. Demonstrar visão sistêmica (28) | Apresentação dos códigos desenvolvidos de forma organizada, demonstrando o conhecimento de todas as partes do projeto. | F |  |  |  |  |  |  |  |  |
| 6. Seguir método de trabalho (21) | Abstração de Objetos, persistência de dados e manipulação de arquivos, seguindo as convenções e boas práticas de POO. | F |  |  |  |  |  |  |  |  |
| **Nível de Desempenho** | | | |  |  |  |  |  |  |  |  |
| **Nota** | | | |  |  |  |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **PLANO DE AULA** | | |
| **CONHECIMENTOS** | **ESTRATÉGIAS DE ENSINO** | **INTERVENÇÕES MEDIADORAS** |
| 3. Programação Orientada a Objetos  3.1. Definição  3.2. Pacotes  3.3. Classes  3.3.4. Atributos  3.3.5. Métodos | Exposição dialogada com auxílio de quadro branco e projetor multimídia. | Como você classificaria os objetos encontrados na cantina do Senai, em um sistema de controle de vendas?  Quais atributos seriam necessários para identificar um aluno com conta na cantina? |
| 3.3.1. Abstrata  3.3.2. Interna  3.3.3. Anônima | Exposição dialogada e Demonstrações. | Para que uma classe abstrata é utilizada? |
| 3.3.6. Modificadores de acesso (encapsulamento)  3.4. Objetos | Demonstrações e Atividades Práticas | A técnica de encapsulamento é realmente necessária na POO, por quê? |
| 3.5. Interface  3.6. Polimorfismo  3.7. Enumerações  3.8. Relacionamentos de objetos | Exposição dialogada, exemplos de casos de uso e Atividades Práticas. | A característica de polimorfismo facilita em quê na vida de um programador? |
| 3.8.1. Herança  3.8.2. Agregação  3.8.3. Composição | Exposição dialogada, exemplos de casos de uso e Atividades Práticas. | Em que a característica de herança se difere da agregação e/ou da composição? |
| 4. Ambiente de desenvolvimento  4.1. Instalação e configuração  4.2. Gerenciamento de dependências  4.3. Recursos e interfaces. | Exposição dialogada com o uso de Atividades Práticas. | Quais facilidades de produtividade um IDE (Ambiente de desenvolvimento) nos traz? |
| 5. Manipulação de arquivos | Exposição dialogada e Demonstrações. | Quantos arquivos seriam necessários para um controle de fichas de “fiados” na cantina do SENAI? |
| 5.1. Escrita  5.2. Leitura | Desenvolvimento de Projeto em Grupo. | Você é capaz de desenvolver um sistema que cadastre e liste lançamentos de entradas e saídas em um livro caixa genérico? |
| 6.1. Definição  6.2. Repositório | Desenvolvimento de Projeto em Grupo. | Ao concluir este miniprojeto (Contas a Receber) em grupo, o desenvolvimento compartilhado utilizando repositório foi possível, qual outro recurso vocês utilizaram? |
| 6. Controle de versões  6.2.1. Inicialização  6.2.2. Remoto  6.2.3. Ramificações (branch)  6.2.4. Versionamento | Desenvolvimento de Projeto em Grupo. | Ao desenvolver este miniprojeto (Controle de entradas e saídas em um almoxarifado) quais as principais dificuldades encontradas na divisão das tarefas e no trabalho remoto? |

|  |  |
| --- | --- |
| **ATIVIDADE** | **SITUAÇÃO DE APRENDIZAGEM – Somativa** |
| **TEXTO** | |
| **Obs:** Aplicada como forma de avaliação Somativa.  **Contextualização**: A empresa LAMBDEV, produtora de bebidas, inaugurou um novo depósito e solicitou ao departamento de tecnologia da informação um sistema que classifique o estoque de produtos finais que serão armazenados neste novo ambiente.  **Desafio**: Como técnico em desenvolvimento de sistemas, você foi contratado para fazer parte de uma equipe que criará o sistema e o instalará em um computador localizado no estoque.  Os seguintes requisitos do sistema foram levantados pelo analista de sistemas:  - Classificar todas as bebidas de acordo com seu tipo;  - Cadastrar cada uma conforme sua classificação;  - Registrar as datas de entrada e saída de cada produto;  - Quantificar cada tipo de produto e calcular o total.   * Exemplos de tipos de bebidas fabricadas pela LAMBDEV: Alcoólica, não alcoólica, cerveja, refrigerante, cachaça e suco. * Basicamente as ações que envolvem estes produtos são entradas e saídas do estoque. * O analista escolheu a linguagem JavaScript e solicita que como técnico, membro da equipe, você crie as classes que serão necessárias para iniciar o desenvolvimento do sistema, conforme os requisitos apresentados e aplique as convenções corretas, já que outros integrantes da equipe desenvolverão outras partes do sistema.   **Entregas**:   * Apresente um diagrama de classes representando apenas os relacionamentos de herança. * Através da plataforma IDE Eclipse apresente o pacote contendo as classes criadas e seus respectivos atributos e métodos, colocando em prática as técnicas de abstração estudadas. * O código deverá apresentar a correta aplicação das técnicas de herança e poliformismo. | |

**INSTRUMENTO DE REGISTRO DE AVALIAÇÃO FORMATIVA**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Natureza dos Critérios** | **Fundamentos Técnicos e Científicos ou Capacidades Técnicas** | **Critérios de avaliação**   |  |  |  |  | | --- | --- | --- | --- | |  | Crítico |  | Desejável | | 0 | NÃO Atingiu | 1 | Atingiu | | F | Formativa | S | Somativa | | | **Alunos** | | | | | | | |
|  |  |  |  |  |  |  |  |
| **Competências Técnicas** | 10. Manipular os diferentes tipos de dados na elaboração de programas | Classificação e identificação de atributos e métodos bem com sua aplicabilidade em uma Classe. | F |  |  |  |  |  |  |  |  |
| 11. Utilizar o ambiente integrado de desenvolvimento (IDE) | Domínio das ferramentas básicas disponibilizadas pela IDE e dos recursos de melhoria de performance no desenvolvimento | F |  |  |  |  |  |  |  |  |
| 12. Utilizar técnicas de versionamento através de softwares específicos | Organização do trabalho, estruturação dos projetos de desenvolvimento de softwares, utilização adequada dos softwares específicos para versionamento. | F |  |  |  |  |  |  |  |  |
| 13. Utilizar o paradigma da programação orientada a objetos | Utilização adequada dos recursos de Encapsulamento, Herança, Poliformismo e outros, para o desenvolvimento das aplicações propostas. | F |  |  |  |  |  |  |  |  |
| 14. Elaborar diagramas de classe | Relacionamento dos conceitos de desenvolvimento e análise de diagramas de classe. | F |  |  |  |  |  |  |  |  |
| **Competências de Gestão** | 3. Demonstrar objetividade na coleta de informações | Classificação adequada de objetos, capacidade de abstrair atributos e métodos, aplicação das técnicas de código limpo. | F |  |  |  |  |  |  |  |  |
| 5. Demonstrar visão sistêmica (28) | Apresentação dos códigos desenvolvidos de forma organizada, demonstrando o conhecimento de todas as partes do projeto. | F |  |  |  |  |  |  |  |  |
| 6. Seguir método de trabalho (21) | Abstração de Objetos, persistência de dados e manipulação de arquivos, seguindo as convenções e boas práticas de POO. | F |  |  |  |  |  |  |  |  |
| **Nível de Desempenho** | | | |  |  |  |  |  |  |  |  |
| **Nota** | | | |  |  |  |  |  |  |  |  |

Instrumento de avaliação / exercícios / Atividades.

|  |  |  |  |
| --- | --- | --- | --- |
| **Componente curricular:** Fundamentos De Programação Orientada A Objeto | **Atividade:** Lista de Exercícios 4 | | **Nota:** |
| **Unidade:** SENAI Jaguariúna | **Data: / /** | |  |
| **Aluno:** | **N°:** | **Turma:** |
| **1** – Crie uma classe **Produto** que possua os seguintes atributos: **nome**, **preço** e **validadeDias**. Através da técnica de **encapsulamento** crie seus métodos **getters** e **setters** e um programa **main** que atribua valores aos atributos e os exiba na tela conforme o código abaixo:  **class** TesteProduto**{**  **public static void** main[String[] args)**{**  Produto p = new Produto();  p.setNome("Arroz");  p.setPreco(12);  p.setValidadeDias(365);  System.out.println(“Nome; Preço; Validade em Dias”);  System.out.println(p.getNome+"; "+p.getPreco+"; "+p.getValidadeDias);  **}**  **}**  **2** – Crie um método main em uma classe chamada **TesteAnimal**, que teste a classe **Animal** descrita abaixo, atribuindo valores aos atributos e mostrando os valores na tela:  **class** Animal**{**  String nome;  String tipo;  **int** peso;  **public static setNome**(String nom)**{**  this.nome = nom;  **}**  **public static setTipo**(String tip)**{**  this.tipo = tip;  **}**  **public static setPeso**(**int** pes)**{**  this.peso = pes;  **}**  **}**  **3** – Utilizando a classe Animal descrita no exercício anterior, implemente um método que mostre na tela os atributos da classe no formato abaixo, chame o método de **exibirAnimal**:  Nome do anilam: Urso  Tipo: Mamífero  Peso: 250  **4** – Altere a classe **TesteAnimal** para que utilize o método criado no exercício acima. | | | |

|  |  |  |
| --- | --- | --- |
| **NÍVEIS DE DESEMPENHO** | **NÍVEIS** | **NOTA** |
| Atingiu todos os critérios críticos e desejáveis | 1 | 100 |
| Atingiu todos os critérios críticos e 2 desejáveis | 2 | 90 |
| Atingiu todos os critérios críticos e 1 desejáveis | 3 | 75 |
| Atingiu todos os critérios críticos | 4 | 50 |
| Atingiu 4 critérios críticos | 6 | 40 |
| Atingiu 3 critérios críticos | 7 | 30 |
| Atingiu 2 critérios críticos | 8 | 20 |
| Atingiu 1critério crítico | 9 | 10 |

|  |  |
| --- | --- |
| **NÍVEL MÍNIMO DE DESEMPENHO ESPERADO** | **4** |

|  |  |  |  |
| --- | --- | --- | --- |
| ELABORAÇÃO | DATA | APROVAÇÃO | DATA |
| **Júlio** | **/ /** |  | **/ /** |

***ANEXOS:***

Exercícios ou qualquer material utilizado no dia a dia.

Atividades\_6\_a\_8\_Orientado\_Objeto;

CRONOGRAMA (deixar por último) O cronograma deve ser atualizado a cada turma nova.

**Cronograma e Acompanhamento de Distribuição de Aulas**

**Curso**: Técnico em Desenvolvimento de Sistemas

**Componente Curricular:** Fundamentos De Programação Orientada A Objeto

**Turma:** 1DES

**Professor**: Reenye e Wellington Lima 2º Sem. 2023

|  |  |  |
| --- | --- | --- |
| Programa Analítico | Dia/Mês/Ano  (Previsto) | Dia/Mês/Ano  (Realizado) |
| 3. Programação Orientada a Objetos  3.1. Definição | 25/09/2023 | 25/09/2023 |
| 3.2. Pacotes  3.3. Classes | 29/09/2023 | 29/09/2023 |
| 3.3.4. Atributos  3.3.5. Métodos | 02/10/2023 | 02/10/2023 |
| 3.3.1. Abstrata | 06/10/2023 | 06/10/2023 |
| 3.3.2. Interna  3.3.3. Anônima | 09/10/2023 | 09/10/2023 |
| 3.3.6. Modificadores de acesso (encapsulamento) | 16/10/2023 |  |
| 3.4. Objetos | 20/10/2023 |  |
| 3.5. Interface  3.6. Polimorfismo | 23/10/2023 |  |
| 3.7. Enumerações  3.8. Relacionamentos de objetos | 27/10/2023 |  |
| 3.8.1. Herança | 30/10/2023 |  |
| 3.8.2. Agregação  3.8.3. Composição | 06/11/2023 |  |
| 4. Ambiente de desenvolvimento  4.1. Instalação e configuração | 10/11/2023 |  |
| 4.2. Gerenciamento de dependências  4.3. Recursos e interfaces | 13/11/2023 |  |
| 5. Manipulação de arquivos | 17/11/2023 |  |
| 5.1. Escrita  5.2. Leitura | 24/11/2023 |  |
| 6. Controle de versões  6.1. Definição  6.2. Repositório | 27/11/2023 |  |
| 6.2.1. Inicialização  6.2.2. Remoto | 01/12/2023 |  |
| 6.2.3. Ramificações (branch)  6.2.4. Versionamento  Avaliação Formativa | 04/12/2023 |  |
| Avaliação Formativa | 08/12/2023 |  |
| Atividades de Recuperação | 11/12/2023 |  |
| Avaliação Somativa | 14/12/2023 |  |
| Atividades de Recuperação | 15/12/2023  18/12/2023 |  |
| Elaborado por: Reenye e Wellington  Data: 18/07/2023 | | |