PLANO DE ENSINO

1. IDENTIFICAÇÃO

Campus: Avançado Quedas do Iguaçu
Eixo tecnológico: Informação e Comunicação
Curso: Técnico em Informática Integrado ao Ensino Médio
Componente curricular: Programação Orientada a Objetos
Docente: Odair Moreira de Souza
Carga horária: 120 h/a e 100 h/r
Turno: Matutino
Número de aulas na semana: 3 aulas
Período letivo: 2019
Turma (s): 3º Ano
Coordenador do curso: Odair Moreira de Souza

2. EMENTA

Linguagem Java. Variáveis e Tipos Primitivos. Orientação a Objetos. Encapsulamento. Herança e Polimorfismo. Classe Abstrata. Interfaces. Pacotes. Arrays e Exceptions. As APIs do Java. Collection Framework. A História da linguagem Java.

3. OBJETIVOS

3.1 Objetivo geral

Compreender os conceitos da programação orientada a objetos, adquirindo habilidades para o desenvolvimento de sistemas em camadas.

3.2 Objetivos específicos

- Entender os fundamentos do Paradigma Orientado a Objetos;
- Aprender uma linguagem de Programação Orientada a Objetos;
- Documentar o processo de desenvolvimento orientado a objetos;
- Compreender os principais recursos da linguagem de programação;
- Conhecer os componentes para o desenvolvimento de software com interfaces gráficas e conexão com banco de dados;
- Modelar e implementar problemas utilizando Programação Orientada a Objetos; e,
- Adquirir domínio básico de uma linguagem de programação orientada a objetos através da aplicação prática dos conceitos aprendidos.

4. CONTEÚDO PROGRAMÁTICO

4.1. Ambiente de Desenvolvimento Integrado

- 4.1.1. Características de um Ambiente de Desenvolvimento
- 4.1.2. Ambiente de Desenvolvimento Integrado *Netbeans*

4.2. Linguagem Java

- 4.2.1. Métodos e Parâmetros
- 4.2.2. Tipos de Dados Básicos e Operadores
- 4.2.3. Entrada e Saída de Dados
- 4.2.4. Estruturas de Controle de Fluxo
- 4.2.5. Modularização

4.3. Conceitos da Orientação a Objetos

- 4.3.1. Conceitos de Classes, Objetos, Atributos e Métodos
- 4.3.2. Abstração e Modularidade
- 4.3.3. Encapsulamento e Modificadores de Acesso
- 4.3.4. Construtores e Destrutores

4.4. Herança e Polimorfismo

- 4.4.1. Reutilização de Código
- 4.4.2. Generalização e Especialização
- 4.4.3. Sobrescrita e Sobrecarga de Métodos

4.5. Classes Abstratas e Interfaces

- 4.5.1. Classes Abstratas
- 4.5.2. Métodos Abstratos
- 4.5.3. Padronização
- 4.5.4. Contratos

4.6. Pacotes

- 4.6.1. Organização de pacotes
- 4.6.2. Níveis de Visibilidade
- 4.6.3. Empacotamento de Bibliotecas
- 4.6.4. Empacotamento de Aplicações

4.7. Documentação

- 4.7.1. Documentação via Javadoc
- 4.7.2. Linguagem UML

4.8. Relacionamentos

- 4.8.1. Dependência
- 4.8.2. Agregação
- 4.8.3. Composição

4.9. Exceções e Erros

- 4.9.1. Erros e Exceptions
- 4.9.2. Capturando e tratando Exceptions

4.10. Classe String

- 4.10.1. Pool de Strings
- 4.10.2. Imutabilidade
- 4.10.3. Classe StringBuilder

4.11. Collections

4.11.1. Listas, Conjuntos e Coleções

4.11.2. Laço Foreach

4.11.3. Generics

4.12. Componentes GUI

4.12.1. Componentes Swing

4.12.2. Gerenciamento de Layout

4.12.3. Events, Listeners e Sources

4.13. Persistência de Dados

4.13.1. APIs para Persistência de dados

5. METODOLOGIA DE ENSINO E AVALIAÇÃO

A metodologia a ser empregada no processo de ensino-aprendizagem consiste em aulas expositivas e dialogadas, incentivando a participação dos discentes durante a explanação do conteúdo, resolução de problemas reais, construção e contextualização dos conhecimentos abordados.

O conteúdo teórico será abordado utilizando como referencial teórico a bibliografia sugerida, por meio de exemplos e estudos de caso e ao final de cada assunto, ou conjunto de assuntos relacionados, será proposto aos discentes o desenvolvimento de exercícios individuais ou em dupla para fixação da teoria apresentada, os alunos irão realizar apresentações das soluções desenvolvidas.

As aulas práticas serão ministradas no laboratório de informática e para todos os conteúdos abordados serão apresentados exemplos desenvolvidos em ambiente computacional e exercícios. Em seguida solicita-se aos discentes que apliquem os conceitos expostos, com o intuito de incentivar a reflexão e a habilidade de raciocínio para resolução de problemas.

As atividades de desenvolvimentos de trabalhos compreenderão a construção de soluções computacionais para os problemas propostos e que propiciem a fixação dos conteúdos previamente identificados, será utilizado problemas de outras áreas do conhecimento, tais como, matemática, física e entre outras, para desenvolver soluções computacionais aplicadas.

5.1 Recursos didáticos

Será utilizado laboratório de informática, projetor multimídia e plataformas online de ensino-aprendizagem de programação.

6. CRITÉRIOS DE AVALIAÇÃO

A avaliação será realizada seguindo as orientações da Resolução nº 50/2017, que estabelece as normas de avaliação dos processos de ensino e aprendizagem no IFPR. Em cada bimestre, o conceito do aluno será composto por meio de sua participação e desempenho nos seguintes pontos:

- Participação e desempenho nos trabalhos individuais;
- Desenvolvimento de aplicações práticas;
- Desempenhos em avaliações individuais teóricas e práticas;
- Participação, assiduidade e proatividade.

Considera-se que para cada bimestre serão aplicados, no mínimo quatro atividades avaliativas práticas de desenvolvimentos (40% do conceito bimestral) e uma ou duas avaliações práticas de desenvolvimento de software (60% do conceito bimestral). Ressalta-se que os trabalhos devem ser apresentados para a turma e/ou somente para o professor em formato de arguição.

Os resultados serão apresentados a cada atividade avaliativa, sendo explicitado o diagnóstico feito pelo docente. De acordo com as normas da Instituição, os alunos receberão os conceitos A, B, C ou D nos períodos determinados pelo IFPR e no final do conteúdo de cada área curricular.

Interdisciplinaridade

Essa disciplina poderá ter atividades avaliativas interdisciplinares, envolvendo conteúdos relacionados com as disciplinas de Banco de Dados, Física e Matemática por meio de trabalhos práticos de desenvolvimentos.

Critérios de Aprovação

Conforme previsto na Resolução nº 50/2017 do IFPR, Art. 16. A aprovação dos estudantes ocorrerá considerando os seguintes critérios:

I – obtenção de conceito A, B ou C no componente curricular e frequência igual ou superior a 75% (setenta e cinco por cento) da carga horária total no período letivo dos cursos técnicos de nível médio.

7. RECUPERAÇÃO

I - Programa de Atividades e de Orientação: A recuperação do estudante poderá ser realizada no decorrer do módulo da disciplina através de atendimento mais direto e individualizado e com atividades diferenciadas que possam contemplar sua aprendizagem, baseada na Resolução nº 50/2017 do IFPR. O aluno com conceito insuficiente em um bimestre será convocado a participar de recuperação paralela contínua durante o próximo bimestre, exceto no último bimestre, pois isso possibilitará ao alunos tempo e atendimento para compreender o conteúdo antes da avaliação de recuperação. Caso a aprendizagem ainda for considerada insuficiente, o estudante cursa a disciplina novamente como progressão, em horários previamente combinados.

II - Formas de Avaliação: As atividades avaliativas de recuperação serão ofertadas em contraturno dos estudantes. Além disso, os alunos contam com atendimentos individualizados pelo professor. As recuperações paralelas serão agendadas com no mínimo 15 (quinze) dias de antecedência da aplicação.

III - Direito de Realizar as Avaliações de Recuperação: Conforme Resolução CONSUP/IFPR nº 50/2017, serão oferecidos estudos de recuperação paralela ou retomada dos conteúdos a todos os estudantes, independente do conceito atingido ser B, C ou D.

8. BIBLIOGRAFIA

8.1 Bibliografia Básica

1. DEITEL, Harvey M.; DEITEL, Paul J. **Java:** como programar. São Paulo, 8ª ed. Prentice Hall Brasil, 2010.

2. JR ENGHOLM, H. **Análise e Design Orientado a Objetos**. São Paulo, 1ª ed. Novatec, 2013.

3. SCHILDT, H. **Java para iniciantes**. Porto Alegre, 5ª ed. Bookman, 2013.

4. TAFNER, Malcon A.; CORREIA, C. **Análise Orientada a Objetos**. Florianópolis, 2ª ed. Visual Books, 2006.

5. TURINI, R. **Desbravando Java e Orientação a Objetos:** Um guia para o iniciante da linguagem. São Paulo, 1ª ed. Casa do Código, 2014.

8.2 Bibliografia Complementar

1. FILHO, Antônio M. S. **Introdução à Programação Orientada a Objetos com C++**. Rio de Janeiro, 1ª ed. Elsevier, 2010.

2. FURGERI, S. **Programação Orientada a Objetos:** Conceitos e Técnicas. São Paulo, 1ª ed. Érica, 2015.

3. LARMAN, C. **Utilizando UML e Padrões:** Uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo. Porto Alegre, 3ª ed. Bookman, 2007.

4. WEST, D. **Use a Cabeça! Análise & Projeto Orientado ao Objeto**. Rio de Janeiro, 1ª ed. Alta Books, 2007.

9. OBSERVAÇÕES

- A distribuição do conteúdo das aulas é uma previsão e poderá ser adequado durante o andamento do ano letivo para atender aos reajustes em função do desempenho da turma ou para tratar eventualidades.
- O Google Classroom será o canal de comunicação digital entre o docente e os discentes, para disponibilização dos materiais das aulas, listas de exercícios, implementações exemplos, submissão de atividades, avisos para a turma, agendamento de atividades.
- O acesso dos estudantes aos laboratórios para o desenvolvimento de atividades em horários alternativos aos das aulas deverá ser solicitado ao professor e ter anuência do responsável pelo laboratório.
- O local e os horários de atendimentos do docente e do monitor de programação serão divulgados nos murais de avisos do campus e na seção de avisos da turma do Google Classroom.

Quedas do Iguaçu, 12 de fevereiro de 2019.

_____

Odair Moreira de Souza
Docente