INF0396 - PROGRAMAÇÃO ORIENTADA A OBJETOS - TB

Summary	Nesta disciplina aprenderemos sobre Programação Orientada a Objetos com Java.
URL	poo
Category	Web
Environment	web, kiosk, io2016, pwa-dev-summit, pwa-roadshow, chrome-dev-summit-2016, io2017, typtwd17, gdd17, cds17, io2018, tag-web, jsconfeu, devfest18, io2019
Status	Published
Feedback Link	https://github.com/googlecodelabs/your-first-pwapp/issues
Author	Renata Dutra Braga
Author LDAP	Braga
Analytics Account	UA-52746336-1

Apresentação

Universidade Federal de Goiás
Escola de Engenharia Elétrica, Mecânica e de Computação
Bacharelado em Ciência da Computação
Disciplina: INF0396 - Programação Orientada a Objetos
Turma B: 2023/2
Professora: Dra. Renata Dutra Braga

Última atualização: 30-09-2023.

Plano de Ensino e Cronograma

Faça a leitura do Plano de Ensino [LINK] e esclareça suas dúvidas.

Nota: Para ter acesso ao plano de ensino, é necessário fazer o *login* utilizando o seu e-mail institucional.

Universidade Federal de Goiás
Escola de Engenharia Elétrica e da Computação
Engenharia de Computação
Matriz Curricular: ECOMP-ENG-2014
Plano de Disciplina
Ano Letivo: 2023 - 2º Semestre

Dados da Disciplina

Código	Nome	Carga Horária	
		Teórica	Prática
133	ALGORITMOS E PROGRAMAÇÃO II	32	32

Prof(a): Renata Dutra Braga

Turma: A

Ementa

Abstração; classe; objeto; herança; polimorfismo. Interface gráfica, entrada e saída (streams). Tratamento de exceção. Concorrência (threads). Ferramentas de desenvolvimento: testes de unidade; controle de versão e geradores (como GNU Make). Modelagem e especificação elementares de aplicações orientadas a objeto. Projeto orientado a objeto: noções, patterns e arquiteturas. Implementação de aplicações orientadas a objetos.

Objetivo Geral

Capacitar o estudante a compreender, de maneira integralizadora, todo o processo de desenvolvimento de aplicações empregando o paradigma denominado orientação a objetos e utilizando a linguagem de programação Java.

Cronograma Aula a Aula

O Cronograma está disponível neste LINK.

Nota: Para ter acesso ao cronograma, é necessário fazer o *login* utilizando o seu e-mail institucional.

Universidade Federal de Goiás
Escola de Engenharia Elétrica, Mecânica e de Computação
Bacharelado em Engenharia de Computação

Cronograma Aula a Aula

Componente Curricular: INF0396 - PROGRAMAÇÃO ORIENTADA A OBJETOS - TB

Turmas: A, B e C - 2023/2
Professores: Me. Dirson Santos de Campos e Dra. Renata Dutra Braga

1. CRONOGRAMA DA DISCIPLINA

UNIDADE 1: Introdução à Orientação a Objetos (Carga horária: 8h):
- Apresentação da metodologia, e tópicos a serem apresentados na disciplina;
- Conceitos básicos sobre programação orientada a objetos (POO) em Java
- Abstração, classes e objetos em POO
- Encapsulamento, visibilidade e modificadores de acesso
- Métodos e construtores em Java

UNIDADE 2: Herança e Polimorfismo (Carga horária: 8h):
- Herança e extensão de classes
- Polimorfismo e sobrescrita de métodos
- Interfaces e classes abstratas
- Exercícios práticos de herança e polimorfismo

UNIDADE 3: Interface Gráfica e Entrada/Saída (Carga horária: 8h):
- Introdução à interface gráfica em POO
- Construção de interfaces gráficas com bibliotecas
- Manipulação de entrada e saída (streams) em POO
- Exercícios práticos de interface gráfica e entrada/saída

UNIDADE 4: Tratamento de Exceções (Carga horária: 8h):
- Tratamento de exceções em POO
- Lançamento e captura de exceções
- Hierarquia de exceções
- Prática de tratamento de exceções

UNIDADE 5: Concorrência e Threads (Carga horária: 6h):
- Introdução à concorrência em POO
- Sincronização e comunicação entre threads
- Exercícios práticos de concorrência

UNIDADE 6: Ferramentas de Desenvolvimento (Carga horária: 10h):
- Testes de Unidade com o JUnit
- Controle de versão com o Git
- Automação de Build com o Maven
- Prática com ferramentas de desenvolvimento

UNIDADE 7: Modelagem e Especificação Orientada a Objetos (Carga horária: 8h):
- Princípios de modelagem orientada a objetos
- Diagramas UML para especificação
- Modelagem de relacionamentos entre classes
- Prática de modelagem e especificação

UNIDADE 8: Design Patterns e Arquiteturas (Carga horária: 8h):
- Noções de design patterns (padrões de criação, estruturais e comportamentais)
- Arquitetura de software orientada a objetos (MVC – Model View Controller)
- Aplicação de padrões de projeto
- Implementação de projetos orientados a objetos

Notas da disciplina

Acesse as notas da disciplina na planilha abaixo:

Planilha de Notas

Nota: Para ter acesso às notas da disciplina, é necessário fazer o *login* utilizando o seu e-mail institucional.

Aula 1

Objetivos da Aula:
- Apresentar o plano da disciplina (ementa, objetivos, conhecimentos prévios e relação com outras disciplinas, cronograma, critérios e formas de avaliação e de verificação de frequência, forma de divulgação de resultados, horários e forma de atendimento,

bibliografia, mecanismos de comunicação entre discentes e docente, postura ética esperada; Discussão do trabalho prático da disciplina).
- Disponibilizar o material da disciplina (plano de ensino e artigos).
- Criar a conta no GitHub

Assunto

- Plano de ensino
- GitHub e SIGAA serão as ferramentas para recebimento dos exercícios da disciplina
- Criar conta no GitHub
- Introdução a orientação a objetos

Exercício E1.1

1. Criar conta no Github (https://github.com), ou usar uma existente.
2. Criar um repositório necessariamente 'público' de nome poo-2023-02, exatamente como está grafado, ou seja, letras minúsculas e hífen para separar poo do número 2023, e outro hífen para separar este número dos dígitos 02.
3. Este repositório será usado durante toda a disciplina. Você é responsável por atualizá-lo conforme as atividades requisitadas. Sua avaliação será realizada sobre o conteúdo depositado neste repositório e conforme os prazos previstos no cronograma da disciplina.

Exercício E1.2

Responda o exercício abaixo. Ele contém as seguintes perguntas:

- Nome completo
- Número de matrícula
- Quais são as suas expectativas em relação à disciplina?
- Tem experiência na área de Programação Orientada a Objetos? Descreva sobre.
- Minha conta no GitHub: Coloque aqui a URL do seu repositório recém-criado. Por exemplo, https://github.com/aluno-poo/poo-2023-02, onde aluno-poo é substituído pela sua conta no GitHub.

Exercício A1.1

Entrega em 26/09/2023, até às 23:59

Introdução a O. O.

1. Lei este documento - autor Fábio Nogueira de Lucena (acesso disponível apenas com o login institucional)
2. IDE para programação em Java:

Que requer instalação	Online
IntelliJ IDEA, https://www.jetbrains.com/pt-br/idea/	Online Java, https://www.online-java.com/
Eclipse, https://www.eclipse.org/downloads/packages/	Replit, https://replit.com/~
Visual Studio Code, https://code.visualstudio.com/	JDoodle, https://www.jdoodle.com/online-java-compiler/

3. Vamos entender os códigos abaixo?

```
// Definição de uma classe simples chamada "Pessoa"
public class Pessoa {
    // Atributos (variáveis de instância) da classe
    private String nome; // Atributo privado
    public int idade;    // Atributo público

    // Construtor da classe
    public Pessoa(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }

    // Métodos da classe
    public void saudacao() {
        System.out.println("Olá, meu nome é " + nome + " e eu tenho
" + idade + " anos.");
    }

    // Método público para acessar o atributo privado "nome"
    public String getNome() {
        return nome;
    }

    // Método público para modificar o atributo privado "nome"
```

```
    public void setNome(String novoNome) {
        nome = novoNome;
    }
}
```

```
public class ExemploClasse {
    public static void main(String[] args) {
        // Criando um objeto da classe Pessoa
        Pessoa pessoa1 = new Pessoa("Alice", 25);

        // Acessando o atributo público "idade"
        System.out.println("Idade: " + pessoa1.idade);

        // Usando o método público para acessar o atributo privado
"nome"
        System.out.println("Nome: " + pessoa1.getNome());

        // Usando o método público para modificar o atributo
privado "nome"
        pessoa1.setNome("Bob");

        // Chamando o método da classe
        pessoa1.saudacao();
    }
}
```

Resultado

1. Reflexão e discussão com os estudantes sobre os conceitos iniciais e O.O.
2. Escolha da IDE (Visual Studio Code)

Aula 2

Assunto

1. Abstração, classes e objetos em POO, encapsulamento, visibilidade, modificadores de acesso, métodos e construtores em Java

Antes de praticar...

1. Orientação a Objetos: Fundamentação
2. Leia os slides para entender mais sobre Orientação a Objetos
3. Reflexão e discussão sobre orientação a objetos. Vamos observar os itens abaixo?!

✅ Objetos são instâncias de classes

```
// Classe que representa um carro
class Carro {
    String marca;
    String modelo;
}

// Instância de um objeto da classe Carro
Carro meuCarro = new Carro();
meuCarro.marca = "Toyota";
meuCarro.modelo = "Corolla";
```

✅ Objetos no mundo real são representados em software por instâncias de classes

No exemplo acima, o objeto `meuCarro` representa um carro do mundo real como uma instância da classe Carro.

✅ Classe inclui dados e comportamentos

```
class Pessoa {
    String nome;
    int idade;

    void cumprimentar() {
        System.out.println("Olá, meu nome é " + nome);
    }
}
```

✅ **Objetos possuem seus próprios dados**

```
Pessoa pessoa1 = new Pessoa();
pessoa1.nome = "Alice";
pessoa1.idade = 30;

Pessoa pessoa2 = new Pessoa();
pessoa2.nome = "Bob";
pessoa2.idade = 25;
```

✅ **Objetos compartilham comportamento da classe**

Ambos os objetos `pessoa1` e `pessoa2` compartilham o mesmo comportamento definido na classe `Pessoa`, como o método cumprimentar.

✅ **Comportamento é descrito via métodos**

Como mostrado no exemplo anterior, o método `cumprimentar` descreve o comportamento da classe Pessoa.

✅ **Chamar um método é enviar uma mensagem**

```
pessoa1.cumprimentar(); // Chama o método cumprimentar para pessoa1
pessoa2.cumprimentar(); // Chama o método cumprimentar para pessoa2
```

✅ **Aplicação OO é um conjunto de objetos que trocam mensagens entre eles**

```
public class AplicacaoOO {
    public static void main(String[] args) {
        Pessoa pessoa1 = new Pessoa();
        pessoa1.nome = "Alice";
        pessoa1.idade = 30;

        Pessoa pessoa2 = new Pessoa();
        pessoa2.nome = "Bob";
        pessoa2.idade = 25;

        pessoa1.cumprimentar(); // Alice envia uma mensagem
cumprimentar()
        pessoa2.cumprimentar(); // Bob envia uma mensagem
cumprimentar()
```

```
    }
}
```

Exercício E1.3

Objetivo

Crie o código correspondente aos seguintes itens/restrições. Execute o programa e observe os resultados.

Itens/Restrições

1. Crie a classe Livro. Esta classe deve possuir os atributos titulo, autor, ano e editora. O titulo é o nome da obra (sequência de caracteres). O segundo desses atributos deve ser uma sequência de caracteres correspondente ao autor da obra em questão. O ano deve ser um inteiro representando o ano de publicação do livro e editora uma sequência de caracteres correspondente ao nome da editora.
2. Crie métodos set/get que permitam definir um valor e obtê-lo, para cada atributo/propriedade desta classe. Estes métodos seguem regras de formação bem definidas. Por exemplo, para a propriedade autor, os métodos correspondentes devem ser identificados por `setAutor` e `getAutor`.
3. Crie a classe `TestaLivro`. Esta classe deverá criar uma instância para cada um dos três livros mais vendidos pela Amazon. Posteriormente, o estado de cada instância deverá ser exibido na saída padrão. Em tempo, Amazon é uma das principais livrarias virtuais do planeta. Os *bestsellers* desta livraria podem ser obtidos em www.amazon.com.
4. Uma abordagem frequente para exibir o estado de uma instância, ou seja, os valores das propriedades da instância, usa o método String `toString()`. Este método é herdado da classe `Object` e, por conseguinte, o uso dele exige que a classe derivada faça uma sobreposição. Convém ressaltar que, em Java, toda classe herda de Object, inclusive arrays.
5. No momento em que este texto foi feito o número um da lista era *The Da Vince Code*, Dan Brown, publicado pela editora Doubleday em 2003. Para esta instância a saída correspondente a ser produzida deve se assemelhar ao que se vê abaixo:

```
The Da Vince Code
Dan Brown
Doubleday, 2003
```

Resposta (disponibilizar depois)

Código-fonte

Resultados

- Exercício 1.3 submetida no repositório público criado no GitHub.

Fazendo um Commit no GitHub:

- Clone o repositório: No seu ambiente de desenvolvimento local, clone o repositório que você criou na aula passada usando o comando git clone, seguido do URL do repositório.
- **Resolva o exercício**: Escreva o código conforme solicitado no Exercício E1.3.
- **Adicione e confirme mudanças**: Use os comandos git add para adicionar os arquivos modificados e git commit para confirmar as mudanças localmente.
- **Envie para o GitHub**: Use o comando git push para enviar as alterações para o repositório no GitHub.
- **Verifique no GitHub**: Acesse o repositório no GitHub para verificar se as mudanças foram enviadas com sucesso.

Atenção!
Verifique se o Git está instalado na sua máquina. Mais detalhe, acesse:
https://git-scm.com/download/