FIAP GRADUAÇÃO

FIAP

ENTERPRISE APPLICATION DEVELOPMENT

Prof. Me. Thiago T. I. Yamamoto

#01 - APRESENTAÇÃO

 *thiagoyama@gmail.com*  *thiagoyama*

#01 - AGENDA

- Apresentação do Professor
- Objetivos da disciplina
- Conteúdo programático
- Metodologia e Avaliações

PROFESSOR – SHORT BIO

THIAGO T. I. YAMAMOTO

Mestre em Ciências, Gestão e Informática em Saúde pela Universidade Federal de São Paulo. Pós-graduado em Engenharia de Sistemas. Bacharel em Ciências da Computação pela Universidade Estadual Paulista - Unesp/Bauru. Mais de 10 anos de experiência na área de TI, como desenvolvedor de sistemas nas empresas: Autbank, UOL e Ericsson Telecomunicações. Professor em curso graduação e pós-graduação da FIAP - ministra várias disciplinas de desenvolvimento de sistemas. Certificado ITIL V3, PSM I e OCJA.

FIAP

ENTERPRISE APPLICATION DEVELOPMENT

OBJETIVOS

- Preparar o professional para o **Mercado de Trabalho;**
- Trabalhar com frameworks **utilizados** no **mercado**;
- Desenvolver ***back end*** de aplicações **Java EE**;
- Apresentar a **Pataforma .NET** e implementar aplicações web com **ASP.NET Core e Entity Framework;**

CONTEÚDO PROGRAMÁTICO

1º Semestre

- **ORM** – JPA/Hibernate**;**
- **EJB**
  - Serialização Sockes e Streams;
  - RMI e JNDI

2º Semestre

- **Plataforma .NET**
  - ASP.NET Core;
  - Entity Framework Core;

ASP.NET

PRÉ-REQUISITOS

- **Lógica de programação**
  - Conhecer os operadores relacionais, estruturas de seleção, repetição, trabalhar com vetores e etc.;
- **Java e Orientação a Objetos**
  - Sintaxe da linguagem, construtores, métodos, atributos, tratamento de exceções, collections e etc.;
  - Os pilares da orientação a objetos: abstração, herança, polimorfismo e encapsulamento;
- **Banco de dados e SQL**
  - Tabelas, Colunas e Relacionamentos;
  - SQL, select, order by, count, group by e etc..

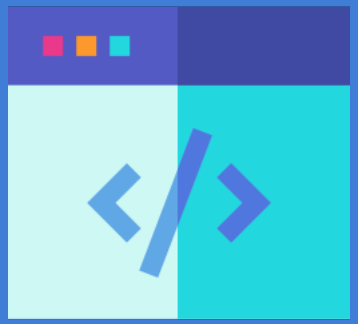
METODOLOGIA E AVALIAÇÕES

- **Metodologia**
  - Aulas “Hands On”;
  - Projetos no Github;

- **Avaliações**
  - **1 NAC** Teórica;
  - **1 NAC** Prática;
  - **Prova Semestral** - Prática;
  - **AM –** Não terá entrega;

REVISÃO

JAVA



- Classe;
- Objeto;
- Herança;
- Encapsulamento;
- Atributos e Métodos;
- Construtores;
- Enums;
- Interfaces;
- O que mais?

JAVA - FUNDAMENTOS

FIAP

- Palavras reservadas;
- Classes:
  - Qual a diferença entre classe e objeto?
- Métodos;
  - Diferença de sobrecarga e sobrescrita?
- Atributos:
  - Quais os tipos e valores padrões?
- Construtores:
  - Precisa sempre ter o construtor **“cheio” e vazio?**

JAVA - OPERADORES

- Operadores matemáticos:

  + , - , * , / , %

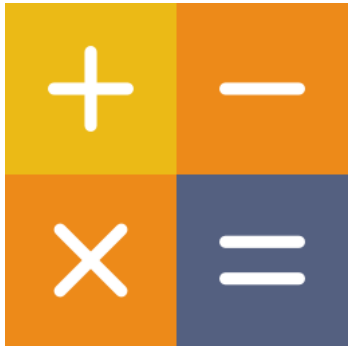
- Operadores relacionais:

  ==, !=, >=, <=, >, <, !, &&, ||

- Instrução if e else;

- Operador ternário:

  – (condicao) ? seVerdadeiro : seFalso;



JAVA - LOOPS

FIAP

- FOR

```
for (int i=0; i<10; i++) { }
```

- WHILE

```
while(x == 0) { }
```

- DO WHILE

```
do { } while(x <0);
```

- FOR EACH

```
for (String item : lista) { }
```

JAVA - COLLECTIONS

FIAP

- LISTS

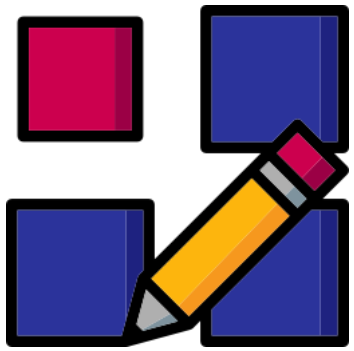
```
List<String> lista = new ArrayList<String>();
```

- SETS

```
Set<Integer> set = new HashSet<Integer>();
```

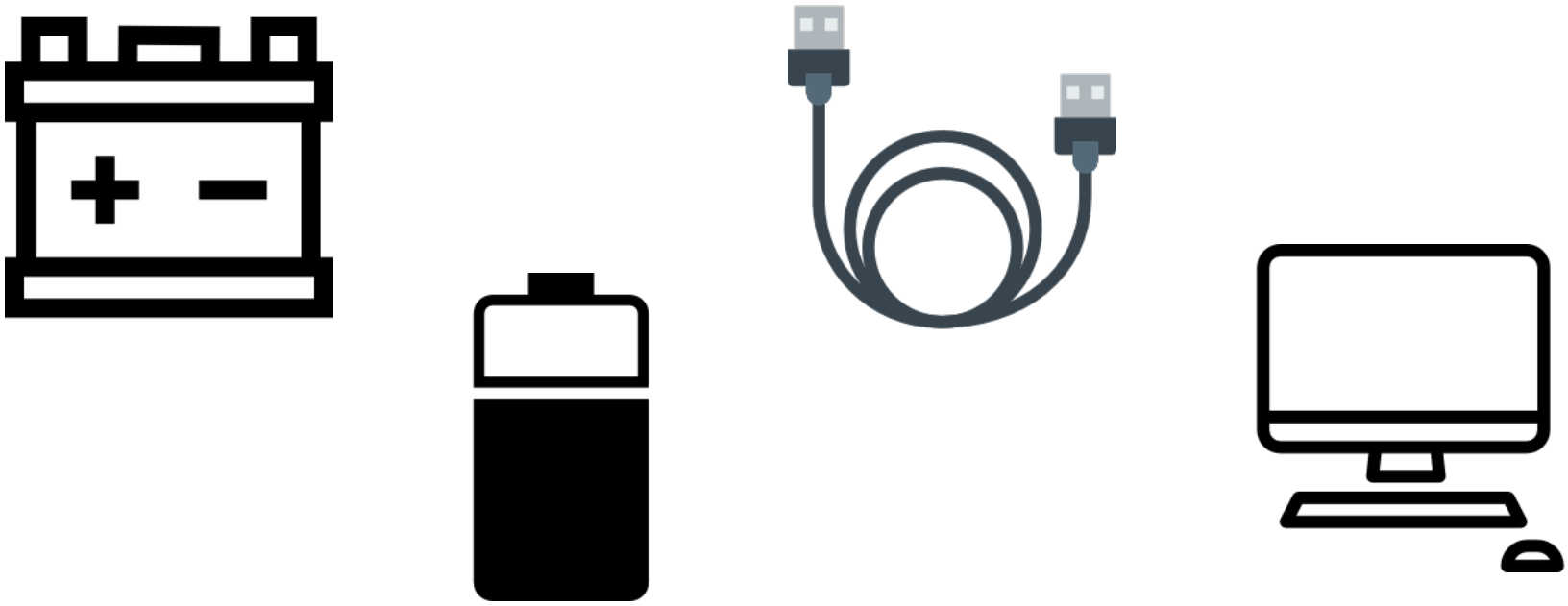
- MAPS

```
Map<Integer,String> map = new HashMap<>();
```



INTERFACES

- Uma **interface** é um conjunto nomeado de **comportamentos** para o qual um implementador precisa fornecer o código;
- Define as **assinaturas** dos métodos;



INTERFACES

FIAP

```
public interface ClienteDAO {
        void cadastrar(Cliente cliente);
        List<Cliente> listar();
}
```

A interface define dois métodos.

Duas classes implementam a interface, uma para utilizar o banco Oracle e outro para o MySQL.

```
public class ClienteDAOMySQL implements ClienteDAO {

    @Override
    public void cadastrar(Cliente cliente) { //… }

    @Override
    public List<Cliente> listar() { //… }

}
```

```
public class ClienteDAOOracle implements ClienteDAO {

    @Override
    public void cadastrar(Cliente cliente) { //… }

    @Override
    public List<Cliente> listar() { //… }

}
```

DATAS

- DATE
  - Classe que armazena o tempo, porém a maioria dos métodos estão marcados como **deprecated;**
- CALENDAR
  - Classe **abstrata** para trabalhar com Data no Java:
  - `Calendar hoje = Calendar.getInstance();`
  - `Calendar data = new GregorianCalendar(ano,mes,dia);`

DATAS - FORMATAÇÃO

FIAP

- SimpleDateFormat

```
Calendar data = Calendar.getInstance();

SimpleDateFormat format = new
      SimpleDateFormat("dd/MM/yyyy");

format.format(data.getTime());
```

DATAS – JAVA 8

FIAP

Java 8 possui uma nova API para datas:

- LocalDate – data, sem horas;
  - `LocalDate hoje = LocalDate.now();`
  - `LocalDate data = LocalDate.of(ano, mes, dia);`
- LocalTime - horas, sem data;
  - `LocalTime time = LocalTime.now();`
  - `LocalTime horas = LocalTime.of(horas, minutos);`
- LocalDateTime – data e horas;
  - `LocalDateTime dateTime = LocalDateTime.now();`
  - `LocalDateTime dataHora = LocalDateTime.of(ano, mes, dia, hora, minutos);`

FIAP

Formatação de datas:

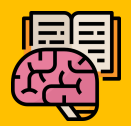
```
LocalDate hoje = LocalDate.now();

DateTimeFormatter formatador =
      DateTimeFormatter.ofPattern("dd/MM/yyyy");

hoje.format(formatador);
```

Artigo sobre API de Datas do Java 8:

http://blog.caelum.com.br/conheca-a-nova-api-de-datas-do-java-8/



ENUM

- Define um conjunto de constantes, valores que não podem ser modificados.

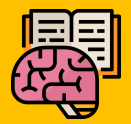
```
public enum Dias {

        SEGUNDA, TERCA, QUARTA, QUINTA, SEXTA,
        SABADO, DOMINGO;

}
```

Artigo sobre enums:

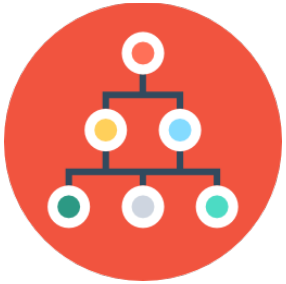
https://www.devmedia.com.br/tipos-enum-no-java/25729



ORIENTAÇÃO À OBJETOS

POO

- Abstração
  - Representação do objeto, características e ações;
- Encapsulamento
  - Restringir o acesso às propriedades e métodos;
- Herança
  - Reutilização de código;
- Polimorfismo
  - Modificação do comportamento de um método herdado;



FIAP

"Se a vida não ficar mais fácil, trate de ficar mais forte."