FIAP GRADUAÇÃO

DIGITAL BUSINESS ENABLEMENT

Prof. Alexandre C. Jesus

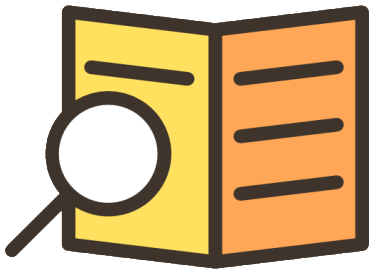
#02 – JAVA

 *profalexandre.jesus@fiap.com.br*  *alecarlosjesus*

PERCURSO

- Java Application

#02 - AGENDA



- Java EE
  - Java EE Arquitetura
  - Java EE Pacotes
- Linguagem Java
  - Fundamentos
  - Operadores
  - Loops
  - Collections
  - Interfaces
  - Datas
  - Enum
- Programação orientada a objetos
- Build e Deployment

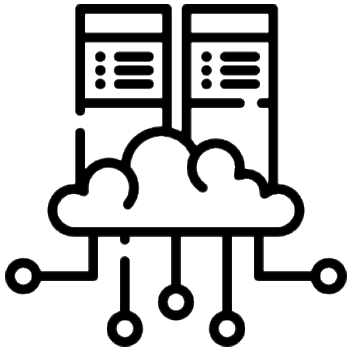
JAVA EE

JAVA EE

- **Java Enterprise Edition** é uma plataforma de desenvolvimento de sistemas de grande porte, que oferece ao desenvolverdor recursos para criação de sistemas com segurança, escalabilidade, integridade, confiabilidade entre outros (requisitos não funcionais). Consiste em uma série de especificações bem detalhadas de como os serviços devem funcionar.

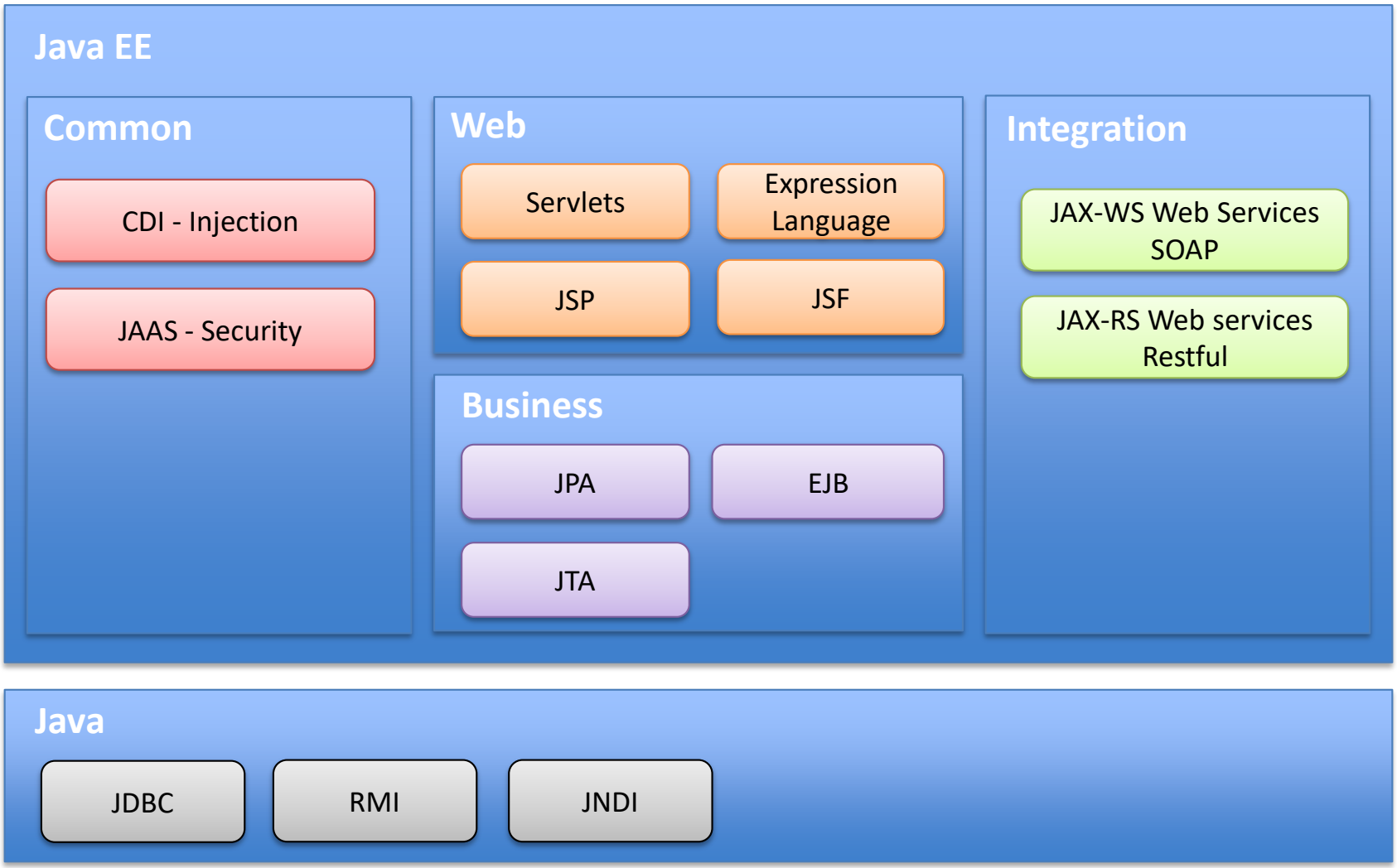
JAVA EE - ARQUITETURA

- **Container**
  - Runtime Enviorment (Ex. Tomcat, Jboss, etc..)



- **Componentes**
  - Classes Java, páginas web e etc..
- **Serviços**
  - Transações, segurança, acesso remoto e etc..
- **Packaging**
  - Empacotar os componentes todos juntos;
- **Deployment**
  - Colocar o “pacote” dentro do servidor para ser executado;

JAVA EE - ARQUITETURA



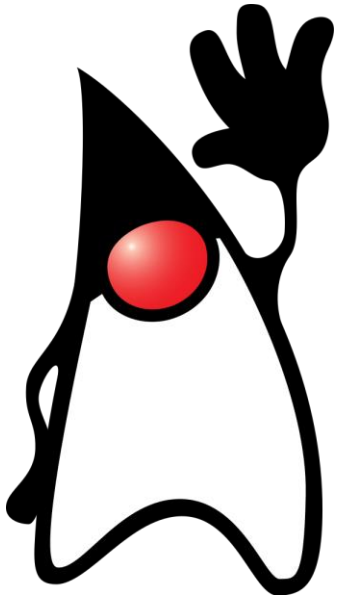
JAVA - PACOTES

- Para **disponibilizar o código da aplicação** é necessário "empacotar" todas as classes e arquivos em um único arquivo.
- Existem alguns tipos de pacotes que servem para diferentes situações:
  - **JAR –** biblioteca, aplicação console/desktop;
  - **WAR (Web Archive)** – aplicação web, pode possuir arquivos JARs, HTML, CSS, JS e etc.
  - **EAR (Enterprise Archive)** – aplicação web corporativa, pode conter vários WARs e EJBs.

JAVA

JAVA

- **Java** é uma linguagem de programação orientada a objetos utilizada para desenvolver uma ampla variedade de sistemas.
- **Visão Geral:**
  - Compilador Java
  - JVM
  - Coletor de lixo
  - JDK
  - JRE



Fundamentos da linguagem Java:

https://www.ibm.com/developerworks/br/java/tutorials/j-introtojava1/index.html

JAVA - FUNDAMENTOS

- **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

- **FOR**

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

- **WHILE**

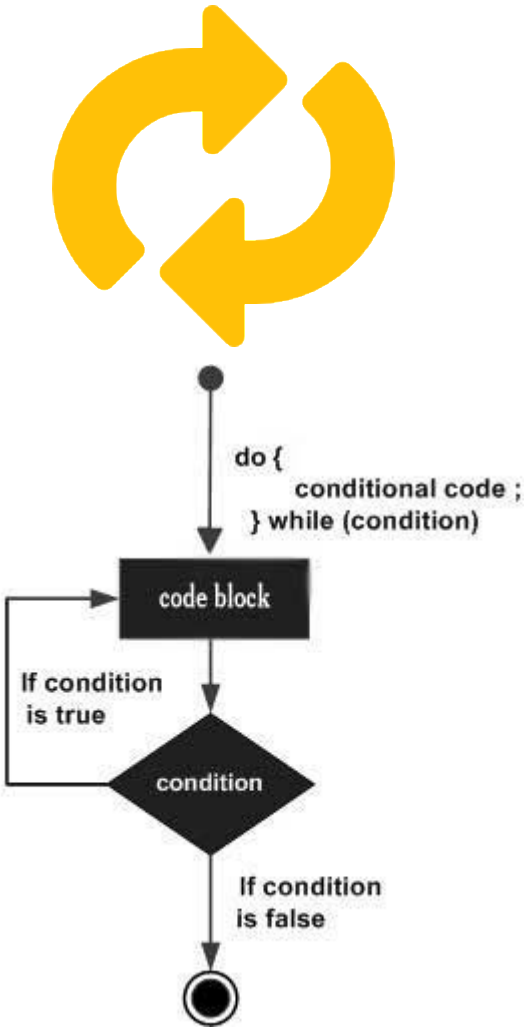
```
while(x == 0) { }
```

- **DO WHILE**

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

- **FOR EACH**

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



JAVA - COLLECTIONS

- **LISTS**

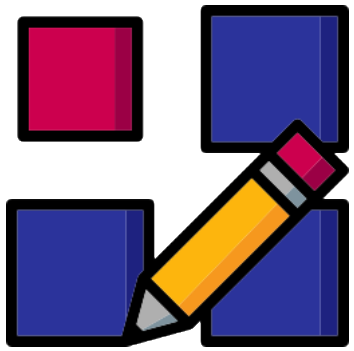
```
List<String> lista = new ArrayList<String>();
```

- **SETS**

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

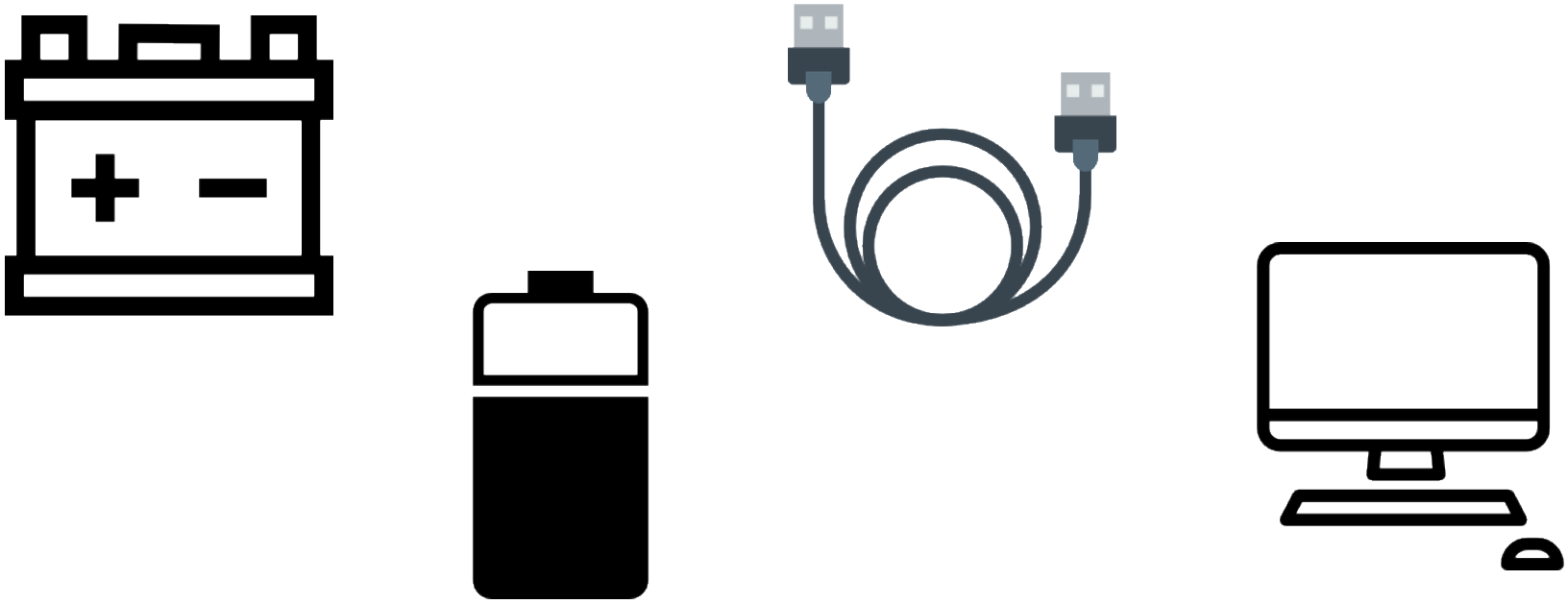
- **MAPS**

```
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

```
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

- **SimpleDateFormat**

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

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

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

Java 8 possui uma nova API para datas:

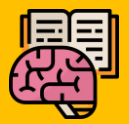
- **LocalDate –** data, sem horas;
  - `LocalDate hoje = LocalDate.now();`
  - `LocalDate data = LocalDate.of(ano, mes, dia);`
- **LocalTime -** horas, sem data;
  - `LocalTime time = LocalTime.now();`
  - `LocalTime horas = LocalTime.of(horas, minutos);`
- **LocalDateTime –** data e horas;
  - `LocalDateTime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

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

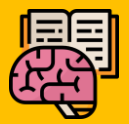
- **Define um conjunto de constantes, valores que não podem ser modificados.**

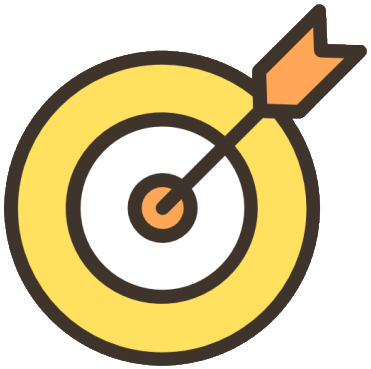
```
public enum Dias {

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

}
```

```
public enum Dias {

SEGUNDA("seg"),
TERCA("ter"),
QUARTA("qua"),
QUINTA("qui"),
SEXTA("sex"),
SABADO("sab"),
DOMINGO("dom");

public String descricao;

Dias(String descricao) {
this.descricao = descricao;
}

public String getDescricao() {
return descricao;
}

}
```

Artigo sobre enums:

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

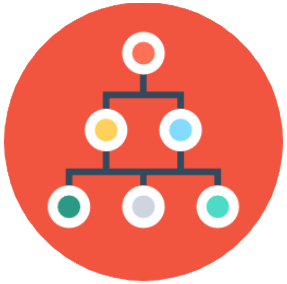




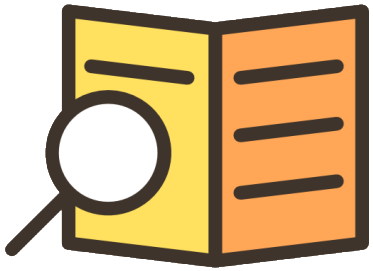
ORIENTAÇÃO A OBJETOS

PROGRAMAÇÃO ORIENTADA A OBJETOS

- **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;



PRÁTICA!



1. Crie uma aplicação Java: **“01-Loja-App”**;

2. Crie uma classe: **br.com.fiap.loja.TerminalConsulta;**

   O usuário deve informar um código de um produto e a aplicação deve retornar a descrição do produto, de acordo com as regras:

   – Código 401 – Camiseta branca;

   – Código 402 – Camiseta azul;

   – Código 403 – Camiseta rosa;

   – Outro Código – Produto não encontrado;

   A aplicação deve ter uma interface texto, que apresente o **nome** da **Loja** e a **Data Atual**;

   Utilize as **teclas de atalho** <Ctrl+Shift+O>, <Ctrl+Shift+F> e <Ctrl+1>;

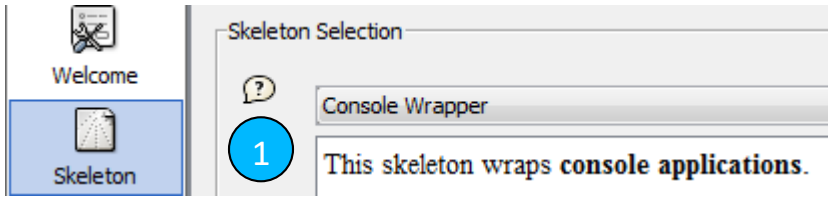
   Realize o **debug** da aplicação;

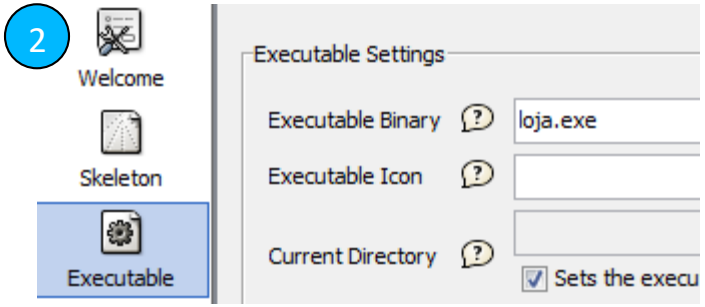
BUILD & DEPLOYMENT

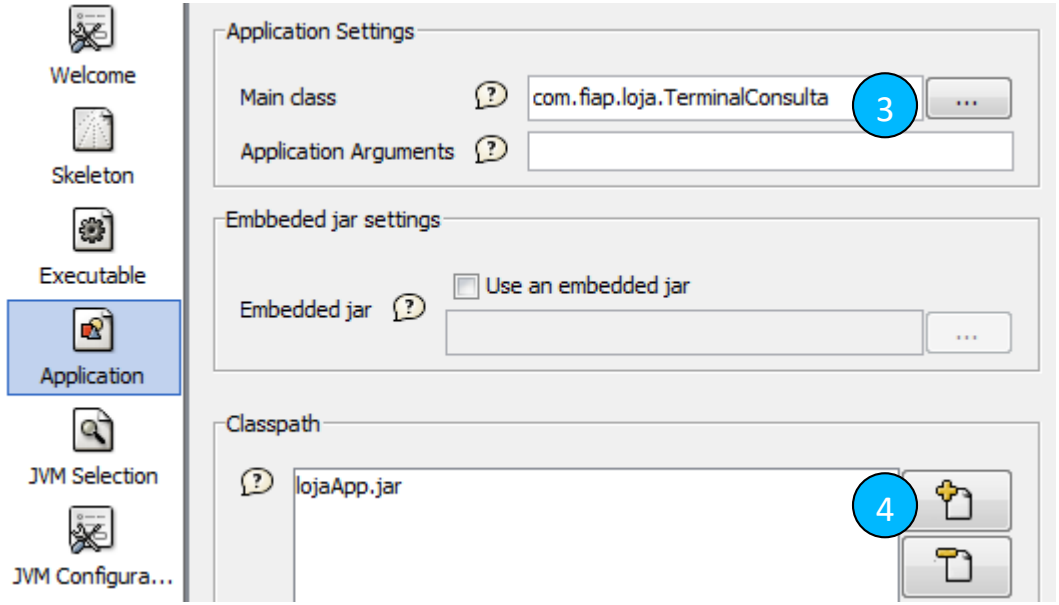
- O deployment Java é realizado pelo **empacotamento** de classes (.class) agrupadas em um arquivo de deployment (.jar);
  - **Eclipse:** Export > Java > Runnable Jar File;
  - **Linha de comando (Win/Mac/Linxu):** java –cp minhaApp.jar com.fiap.Aplicacao
- Para windows, há a opção de **gerar um arquivo .exe** que age como wrapper (casca) para aplicações windows.
  - Utilitário externo para gerar o .exe: **Jsmooth** (http://jsmooth.sourceforge.net)

BUILD & DEPLOYMENT

- Siga os passos para gerar o arquivo exe:



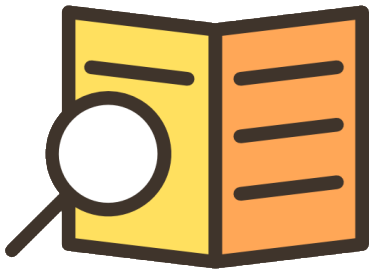




PRÁTICA! DEPLOYMENT

1. Faça o deployment da aplicação **01-Loja-App**

VOCÊ APRENDEU...



- Sobre **Java EE**, sua arquitetura principal e tipos de arquivos para deployment;
- Revisão da linguagem Java: tipos de dados, operadores, loops, collections, **datas, enums e interafaces;**
- Revisão da **programação orientada a objetos**, abstração, encapsulamento, polimorfismo e herança;
- Realizar o deployment de um programa **Console Java (exe)**;

"Para de perseguir o dinheiro e comece a perseguir o sucesso"
– Tony Hsieh