Aplicativos em JAVA. Vamos analisar as principais diferenças entre os três exemplos de aplicações Java:

1️⃣ Aplicação Java Pura com NetBeans

📌 **Exemplo:**

```
public class OlaMundo {
    public static void main(String[] args) {
        System.out.println("Olá, Mundo!");
    }
}
```

✅ **Características**

✓ **Execução via terminal:** Não tem interface gráfica, apenas exibe texto no console.
✓ **Simples e rápido:** Útil para aprender a base da linguagem Java.
✓ **Independente de frameworks:** Usa apenas a **JDK padrão**.
✓ **Menos dependências:** Apenas uma classe com o método main.

❌ **Limitações**

⛔ Apenas interage com o usuário via **texto no console**.
⛔ Não permite botões, janelas ou formulários.

2️⃣ Aplicação Java com Swing (Interface Gráfica)

📌 **Exemplo:**

```
import javax.swing.*;


public class OlaMundoSwing {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Minha Janela");
        JLabel label = new JLabel("Olá, Mundo!", SwingConstants.CENTER);
        frame.add(label);
        frame.setSize(300, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
```

}

✅ **Características**

✓ **Interface Gráfica (GUI):** Usa **Swing** para criar janelas e botões.
✓ **Multiplataforma:** Funciona no Windows, Linux e macOS.
✓ **Não depende de frameworks externos:** O Swing já vem embutido no Java.

❌ **Limitações**

⛔ O Swing tem uma aparência **antiga** comparado a frameworks modernos.
⛔ Requer **mais código** para interfaces mais elaboradas.
⛔ Não é ideal para aplicações **web** ou escaláveis.

3️⃣ Aplicação Java com Spring Boot (Backend Web)

📌 **Exemplo:**

```
import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;


@RestController

public class OlaMundoController {


  @GetMapping("/")

  public String hello() {

    return "Olá, Mundo!";

  }

}
```

✅ **Características**

✓ **Aplicação Web:** Funciona via navegador em vez de uma janela gráfica.
✓ **Baseada em APIs:** O Spring Boot facilita a criação de **APIs REST**.
✓ **Extremamente escalável:** Ideal para sistemas grandes, como e-commerces ou redes sociais.
✓ **Possui injeção de dependências:** Código modular e organizado.

❌ **Limitações**

⛔ Precisa de **mais configuração** comparado a um simples main().
⛔ **Não tem interface gráfica embutida**, precisa de **HTML/CSS/JavaScript** para frontend.
⛔ Mais pesado que uma aplicação pura em Java.

🎯 Comparação Final

Característica	Java Puro	Swing	Spring Boot
Interface	Console (texto)	Gráfica (janelas)	Web (navegador)
Complexidade	Simples	Média	Alta
Uso comum	Scripts simples	Aplicações Desktop	Aplicações Web e APIs
Escalabilidade	Baixa	Média	Alta
Dependências	Nenhuma	Nenhuma	Spring Framework
Execução	Via terminal	Aplicação desktop	Servidor web

🔥 Conclusão

- **Java Puro** → Melhor para **testes rápidos**, programas simples e aprendizado básico.
- **Swing** → Ideal para **aplicações desktop pequenas**, mas limitado para grandes projetos.
- **Spring Boot** → Melhor para **sistemas complexos**, **web apps** e **APIs REST**.

Se quiser desenvolver um **sistema grande e moderno**, Spring Boot é a melhor escolha. Se quer um programa pequeno e offline, Swing pode ser útil.