Program 3: Working with Gradle: Setting Up a Gradle Project, Understanding Build Scripts (Groovy and Kotlin DSL), Dependency Management and Task Automation

Step 1: Download & Install IntelliJ IDEA

- Go to: https://www.jetbrains.com/idea/download
- Download **Community Edition** (Free).
- Install IntelliJ IDEA by following the installation steps for your OS.

Step 2: Create a New Gradle Project

Open IntelliJ IDEA

- Click **New Project**.

Select Project Type

- Choose **Gradle** on the left sidebar.
- In **Project SDK**, select **Java SDK** (download if not installed).
- Make sure **Groovy** DSL is selected (not Kotlin).

Configure Project

- Name your project (e.g., `GradleDemo`).
- Select **Use auto-import** to keep dependencies synced.
- Finish.

Step 3: Understand the Gradle Project Structure

```
GradleDemo/
├── build.gradle          ← Groovy-based build script
├── settings.gradle
└── src/
    └── main/
        └── java/         ← Place Java code here
```

Step 4: Writing Your First Groovy Build Script (build.gradle)

*(By default, IntelliJ creates a **basic build.gradle** file. Let's **modify it manually** to understand.)*

- Open `build.gradle` : you'll see something like

```
plugins {
    id 'java'
}

group 'org.example'
version '1.0-SNAPSHOT'
```

```
repositories {
    mavenCentral()  // Tells Gradle where to find dependencies
}

dependencies {
    // You can add libraries here
    testImplementation 'junit:junit:4.13.2'
}
```

Step 5: Adding Dependencies

Example: Add Google's Guava library

```
dependencies
{
    implementation 'com.google.guava:guava:32.1.2-jre'
}
```

- **implementation** means it's needed for the project during compile/run time.
- Gradle will download it from **Maven Central** automatically.

After adding the dependency:

1. IntelliJ will **auto-import**, or you can click **"Load Gradle Changes"**.
2. Guava will be downloaded and added to your project automatically.

Step 6: Task Automation with Custom Gradle Tasks (Groovy)

Create a Simple Custom Task

At the **bottom** of build.gradle, add:

```
task helloWorld {
    doLast {
        println 'Hello, Gradle!'
    }
}
```

Run the Task

In IntelliJ's **Gradle Tool Window** (on the right):

- Expand `Tasks` ➜ `other`.
- You'll see `helloWorld`.
- Double-click it.

OR from Terminal (in IntelliJ): `./gradlew helloWorld`

Step 7: Automating Build Steps

Example: Task that runs after `build`

```
tasks.build {
    doLast {
        println "Build finished! You can deploy now."
    }
}
```

Now, when you run: `./gradlew build`

At the end, it will print: `Build finished! You can deploy now.`