Creating a JAR (Java ARchive) file from a simple "Hello, World!" Java program involves compiling the Java code and then packaging the compiled class files into a JAR file. Here are the steps to achieve this:

**Write the Java Program**

Create a file named HelloWorld.java with the following content:

public class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, World!");

}

}

**Compile the Java Program**

Open a terminal or command prompt and navigate to the directory where HelloWorld.java is located. Compile the Java file using the javac command:

javac HelloWorld.java

**Create a Manifest File**

To specify the entry point of the application (the main class), you need to create a manifest file. Create a file named manifest.txt with the following content:

Main-Class: HelloWorld

**Create the JAR File**

Use the jar command to create the JAR file. Run the following command in the terminal:

jar cfm HelloWorld.jar manifest.txt HelloWorld.class

This command creates a JAR file named HelloWorld.jar that includes the HelloWorld.class file and the manifest file.

**Run the JAR File**

You can run the JAR file using the java -jar command:

java -jar HelloWorld.jar

**Notes**

* Ensure that the JDK (Java Development Kit) is installed and the javac and jar commands are available in your system's PATH.
* The manifest.txt file should have a newline at the end to be properly recognized by the jar command.
* If you have additional class files or resources to include in the JAR, list them after manifest.txt in the jar command.

JRE vs JDK

Java Development Kit (JDK) and Java Runtime Environment (JRE) are both parts of the Java platform, but they serve different purposes.

### Java Runtime Environment (JRE)

* **Purpose**: The JRE provides the libraries, Java Virtual Machine (JVM), and other components needed to run applications written in Java. It does not contain tools for Java development (like compilers and debuggers).
* **Components**:
  + **Java Virtual Machine (JVM)**: The engine that runs Java applications.
  + **Core Libraries**: Pre-written libraries that Java applications use for common tasks.
  + **Java Plug-in**: A component to run applets in browsers.
  + **Java Web Start**: Allows you to deploy standalone applications over a network.
* **Use Case**: If you only need to run Java applications and do not need to write or compile Java code, you only need the JRE.

### Java Development Kit (JDK)

* **Purpose**: The JDK is a full-featured software development kit for Java, which includes the JRE along with tools for developing, debugging, and monitoring Java applications.
* **Components**:
  + **Java Runtime Environment (JRE)**: Everything needed to run Java applications.
  + **Development Tools**:
    - **javac**: The Java compiler that translates Java source code into bytecode.
    - **jar**: The tool for packaging Java classes into JAR files.
    - **javadoc**: Generates API documentation from Java source code comments.
    - **jdb**: The Java debugger.
    - **Other tools**: Includes various utilities for monitoring and profiling Java applications.
* **Use Case**: If you need to develop Java applications, you need the JDK. It includes the compiler, libraries, and other tools necessary for Java development.

### Comparison

| **Feature** | **JDK** | **JRE** |
| --- | --- | --- |
| Contains JVM | Yes | Yes |
| Contains Libraries | Yes | Yes |
| Contains Compiler | Yes (javac) | No |
| Development Tools | Yes (debugger, javadoc, jar, etc.) | No |
| Primary Use | Developing and running Java applications | Running Java applications only |

### Installation

### apt install openjdk-17-jre-headless

* apt install openjdk-21-jdk-headless
* java --version