This program does the following:

1. It defines a Folder class to represent a folder in the file system. Each Folder object contains the folder’s name, the total size of the files in the folder, the number of files, and a list of its child folders.
2. The printFolder method in the Folder class prints the folder’s information and recursively prints the information of its child folders, with an appropriate indentation.
3. The main method reads the name of the folder to scan from the command line argument, constructs the folder tree using the constructFolder method, and then prints the folder tree.
4. The constructFolder method constructs a Folder object for the given File object. If the File object is a directory, it recursively constructs Folder objects for its child files and adds them to its list of child folders. If the File object is a regular file, it updates the number of files and the total size.

You can run this program with the path of the folder to scan as the command line argument. For example, if you’re using the terminal, you can compile the program with javac Main.java and then run it with java Main /path/to/folder.

Please note that this program doesn’t handle any potential IOException that may be thrown when accessing the file system. Depending on your needs, you might want to add appropriate error handling. Also, this program counts all files, including hidden files. If you want to exclude hidden files, you can do so by adding an appropriate check in the constructFolder method.

As for creating a javac of it, javac is the Java compiler which you use to compile your Java source code into bytecode. You can do this from the command line. Here’s how you can compile your Java program:

1. Open a command prompt or terminal.
2. Navigate to the directory where your Java file is located.
3. Run the following command: javac Main.java

This will compile your Main.java file and produce a Main.class file, which is the bytecode that can be run on the Java Virtual Machine (JVM).

To run your program, you can use the java command followed by the name of the class that contains the main method (without the .class extension). In your case, you can run your program with the following command: java Main /path/to/folder.

Yes, you can use the same JDK that IntelliJ IDEA is using to set your JAVA\_HOME. Here’s how you can do it:

1. Open IntelliJ IDEA.
2. Go to File -> Project Structure.
3. In the Project Structure dialog that appears, under the Project SDK section, you’ll see the path to the JDK that IntelliJ IDEA is using.
4. Copy this path.
5. Open a new terminal window.
6. Type the following command and press Enter:

For Windows:

set JAVA\_HOME=" C:\Program Files\Java"

set PATH=%JAVA\_HOME%\bin;%PATH%

For Linux:

export JAVA\_HOME=copied\_path

export PATH=$JAVA\_HOME/bin:$PATH

Replace copied\_path with the path you copied from IntelliJ IDEA.

1. To verify that the change was successful, type java -version in the terminal. You should now see that it’s using the same Java version as IntelliJ IDEA.

Please note that these changes will only apply to the current terminal session. If you open a new terminal window, you will have to enter these commands again. To make these changes permanent, you can add the set or export commands to your system’s environment variables or to a startup script like ~/.bashrc or ~/.bash\_profile for Linux or Environment Variables in the System Properties for Windows.

It seems like you’re using PowerShell, which has a different syntax for setting environment variables compared to the traditional Command Prompt (cmd.exe).

In PowerShell, you can set environment variables using the $env: prefix. Here’s how you can set JAVA\_HOME and update PATH in PowerShell:

$env:JAVA\_HOME="C:\Program Files\Java"

$env:PATH="$env:JAVA\_HOME\bin;$env:PATH"

Please replace "C:\Program Files\Java" with the actual path to your JDK 17 installation.

After running these commands, you can verify that JAVA\_HOME is set correctly and that PATH includes the JDK bin directory by running these commands:

echo $env:JAVA\_HOME

echo $env:PATH

Please note that these changes will only apply to the current PowerShell session. If you open a new PowerShell window, you will have to enter these commands again. To make these changes permanent, you can add them to your system’s environment variables.