Running Java Applications

Running Java App

In order to run your Java Applications you must first compile all of the source code into it's `.class` byte code file respectively. Then in order to run your code you have to specify the fully qualified name for the class to the `java` command.

> When you are running the code, the current working directory doesn't really matter as you can change the search path using the -cp or --classpath option

Fully qualified name

The fully qualified name is the class under the corresponding package. A package is just a list of related files, think of it as a folder. When you write a Java class say under the `src/com/yy` directory call it `Circle.java`, then `Circle.class`'s fully qualified name is `com.yy.Circle` after you have compiled it into byte code.

> src is not part of the package name, src itself is not a package, but inside the src directory all of it is considered to be java packages, and they have their respectively qualified name starting from inside src.

How does `java` run the program?

`java` will take a fully qualified name, not a directory to the `.class` file! Keep in mind!

When you use `java` along with a fully qualified name it is going to do the following:

1. Search for the compiled version of the fully qualified named class
2. Load the class
3. Check that the class has a `main` method
4. Call that method and passing it the command line arguments if any

Why Java cannot find the class

Remember that fully qualified name is not a file path! It is the name to refer to that class underneath `src`.

If your `.class` that you want to run is under the fully qualified name `com.yy.Circle` then you must specify that as the fully qualified name, not `Circle` or `circle`.

Wrong classpath

Now, when you execute `java` and provide a fully qualified name, it is going to search for that `.class` in couple of places. If you set the `CLASSPATH` environment variable then it is going to search through that list of directory to find `.class`. If not then it is going to check if you have provided any `-cp/-classpath` argument to the `java` command and check through those. If you didn't specify it then the current directory is going to be used as the classpath.

- To make it clear say you are executing `com.acme.example.Foon`, the `Foon.class` class
- The full file path to the `Foon.class` is `/usr/local/acme/classes/com/acme/example/Foon.class`
- Your current working directory is `/usr/local/acme/classes/com/acme/example/`

Then in order to run `Foon.class` you have to run

```
java -cp /usr/local/acme/classes com.acme.example.Foon
```

Now Java is able to find the correct `Foon.class` file by following the classpath you have specified, and looking it under `/usr/local/acme/classes/com/acme/example` directory to find that `Foon.class` file correctly.

Basically you have to provide the source directory path to the `.class` that you have compiled. That's all. It will follow the fully qualified name underneath `classes` directory to find `com` folder, then follow `acme` folder then `example` folder and finally find the correctly file.

> It is as if it cd to the classpath you have provided, then try to locate that file by following your qualified name like a folder traversal.

How do I import classes I wrote in same package?

You do not need to import it, you can just use it directly because they exist in the same package.

Wait I wrote a class directly under src, how do I import it?

You cannot directly import classes you have written directly under src folder because it technically does not have a package name. However, it is able to be exposed to outside.

Revision #2
Created 2023-01-31 03:34:34 UTC by Tamarine
Updated 2023-03-17 16:51:43 UTC by Tamarine