Bazel with Java

1. Install the JDK

1. Install Java JDK (preferred version is 11, however versions between 8 and 15 are supported).
2. Set the JAVA_HOME environment variable to point to the JDK.
    - On Linux/macOS:

      ```
      export JAVA_HOME="$(dirname $(dirname $(realpath $(which javac))))"
      ```

    - On Windows:
        a. Open Control Panel.
        b. Go to "System and Security" > "System" > "Advanced System Settings" > "Advanced" tab > "Environment Variables..." .
        c. Under the "User variables" list (the one on the top), click "New...".
        d. In the "Variable name" field, enter JAVA_HOME.
        e. Click "Browse Directory...".
        f. Navigate to the JDK directory (for example C:\Program Files\Java\jdk1.8.0_152).
        g. Click "OK" on all dialog windows.

1. Create the following directory structure:

```
java-tutorial
├── BUILD
├── src
│   └── main
│       └── java
│           └── com
│               └── example
│                   ├── cmdline
│                   │   ├── BUILD
│                   │   └── Runner.java
│                   ├── Greeting.java
│                   └── ProjectRunner.java
└── MODULE.bazel
```

java-tutorial/BUILD.bazel file

```
java_binary(
    name = "ProjectRunner",
```

```
    srcs = glob(["src/main/java/com/example/*.java"]),
    main_class = "main.java.com.example.ProjectRunner",
)
```

Greeting.java

```
package main.java.com.example;

public class Greeting {
    public static void sayHi() {
        System.out.println("Hi!");
    }
}
```

ProjectRunner.java

```
package main.java.com.example;

public class ProjectRunner {
    public static void main(String args[]) {
        Greeting.sayHi();
    }
}
```

Navigate to java-tutorial directory and execute the commands on terminal

$ bazel build //:ProjectRunner

$ bazel-bin/ProjectRunner

or

$bazel run //:ProjectRunner

Review Dependency graph

```
$ bazel query  --notool_deps --noimplicit_deps "deps(//:ProjectRunner)"
--output graph
```

Copy the output string to http://www.webgraphviz.com/ . you will be able to see dependency graph

Specify multiple build targets

Modify java-tutorial/BUILD.bazel file

```
java_binary(
    name = "ProjectRunner",
    srcs = glob(["src/main/java/com/example/ProjectRunner.java"]),
    main_class = "main.java.com.example.ProjectRunner",
```

```
    deps = [":greeter"],
)
java_library(
    name = "greeter",
    srcs = ["src/main/java/com/example/Greeting.java"],
)
```

Navigate to java-tutorial directory and run
$bazel run //:ProjectRunner

Again use query to visualize the dependency

Use multiple packages

cmdline/BUILD.bazel

```
java_binary(
    name = "runner",
    srcs = ["Runner.java"],
    main_class = "main.java.com.example.cmdline.Runner",
    deps = ["//:greeter"],
)
```

Java-tutorial/BUILD.bazel

```
java_binary(
    name = "ProjectRunner",
    srcs = glob(["src/main/java/com/example/ProjectRunner.java"]),
    main_class = "main.java.com.example.ProjectRunner",
    deps = [":greeter"],
)
java_library(
    name = "greeter",
    srcs = ["src/main/java/com/example/Greeting.java"],
    visibility = ["//src/main/java/com/example/cmdline:__pkg__"],

)
```

Navigate to java-tutorial directory and run

```
$ bazel build //src/main/java/com/example/cmdline:runner
```

```
Check dependencies using bazel query
```

$ bazel query --notool_deps --noimplicit_deps
"deps(//src/main/java/com/example/cmdline:runner)" --output graph

Package a Java target for deployment

Examine the content of runner.jar file

$ jar tf bazel-bin/src/main/java/com/example/cmdline/runner.jar

META-INF/
META-INF/MANIFEST.MF
main/
main/java/
main/java/com/
main/java/com/example/
main/java/com/example/cmdline/
main/java/com/example/cmdline/Runner.class

It is not showing greeting.class, so it can run locally, but not outside the development environment.

To run independent of development environment, add "_deploy.jar"

```
$ bazel build //src/main/java/com/example/cmdline:runner_deploy.jar
$ jar tf bazel-bin/src/main/java/com/example/cmdline/runner_deploy.jar
```

```
META-INF/
META-INF/MANIFEST.MF
build-data.properties
main/
main/java/
main/java/com/
main/java/com/example/
main/java/com/example/cmdline/
main/java/com/example/cmdline/Runner.class
main/java/com/example/Greeting.class
```

Adding External dependencies

BUILD.bazel

```
load("@rules_java//java:defs.bzl", "java_binary", "java_library")


java_binary(
    name = "ProjectRunner",
    srcs = ["src/main/java/com/example/ProjectRunner.java"],
    main_class = "main.java.com.example.ProjectRunner",
    deps = [":greeter"],
)
java_binary(
    name="salutations",
    main_class = "main.java.com.example.Salutations",
    runtime_deps = [":textio"],
```

```
)

java_library(
    name="textio",
    srcs = ["src/main/java/com/example/Salutations.java"],
    deps = [":greeter","@maven//:org_beryx_text_io"],
    #visibility = ["//src/main/java/com/example/cmdline:__pkg__"],
)
java_library(
    name = "greeter",
    srcs = ["src/main/java/com/example/Greeting.java"],
    visibility = ["//src/main/java/com/example/cmdline:__pkg__"],

)
```

MODULE.bazel

```
bazel_dep(name = "rules_jvm_external", version = "5.3")
maven = use_extension("@rules_jvm_external//:extensions.bzl", "maven")
maven.install(
    artifacts = [
        "org.beryx:text-io:3.4.1",
    ],
    fetch_sources = True,
    repositories = [
       "https://maven.google.com",
        "https://repo1.maven.org/maven2",
        "https://jcenter.bintray.com/",
        "http://uk.maven.org/maven2",
    ],
)
use_repo(maven, "maven")
```

Salutations.java

```
package main.java.com.example;
import org.beryx.textio.TextIO;
import org.beryx.textio.TextIoFactory;


public class Salutations {

    public static void main(String[] args) {
        TextIO textIO = TextIoFactory.getTextIO();

        String user = textIO.newStringInputReader()
            .withDefaultValue("admin")
            .read("Username");

        String password = textIO.newStringInputReader()
            .withMinLength(6)
            .withInputMasking(true)
            .read("Password");

    }
}
```