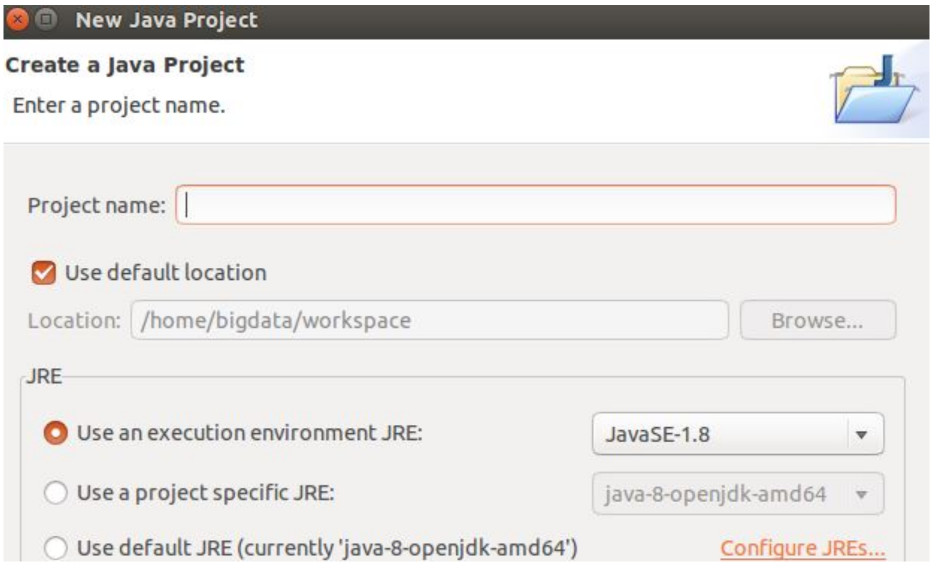
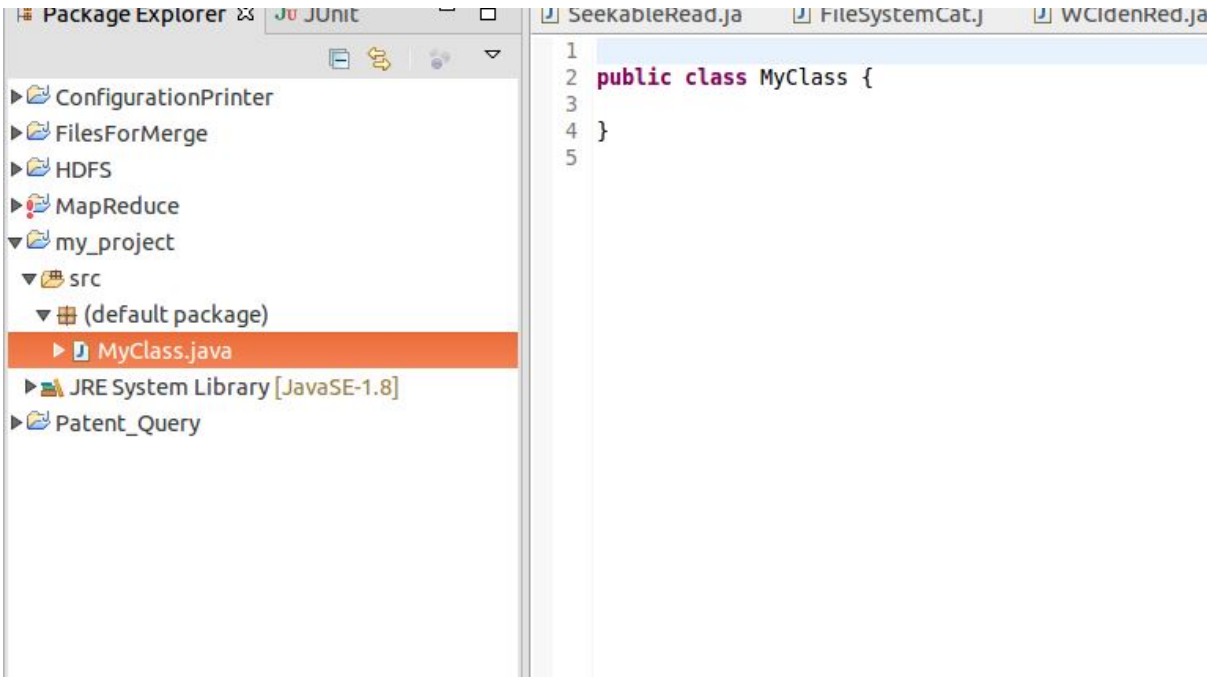
In this subject, it is recommended (but not compulsory) that you edit Java programs in Eclipse which is installed in the BigdataVM. This document provides an instruction to set up the environment in Eclipse for developing the java codes related to this subject.

Create a Java Project: On the Eclipse menu, “File” -> “New” -> “Java Project”.


New Java Project
Create a Java Project
Enter a project name.
Project name:
Use default location
Location: /home/bigdata/workspace
Browse...
JRE
Use an execution environment JRE: JavaSE-1.8
Use a project specific JRE: java-8-openjdk-amd64
Use default JRE (currently 'java-8-openjdk-amd64')
Configure JREs...

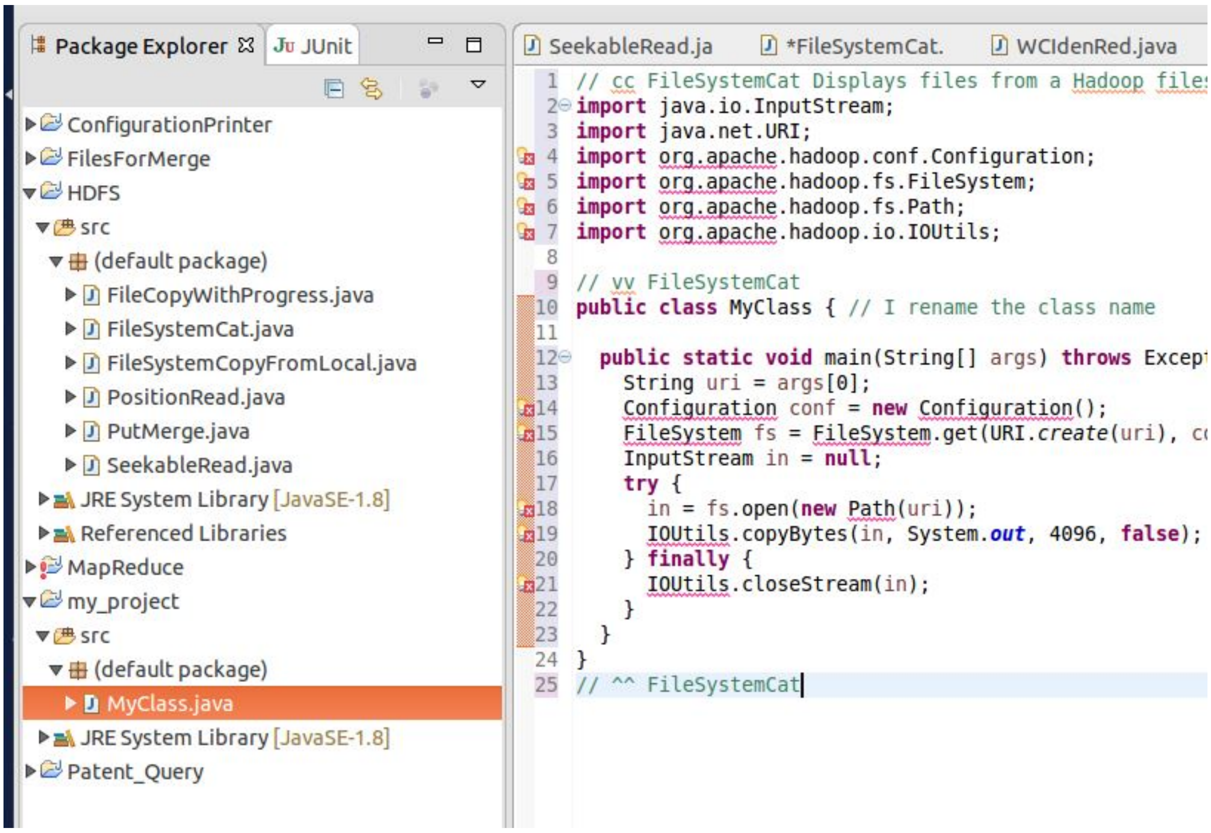
Give your new project a name. After the new project is created, right-click its folder symbol, “New” -> “Class” to create Java Class. You can just leave the “Package” blank and enter a name for your new java class.



```java

public class MyClass {

}

```

Copy the given codes, say, FileSystemCat, to the class.

(Note. Because you use an existing piece of Java code, you have to make sure the class names are the same. For example, the class name you entered previous should be “FileSystemCat”, or change the class name in the given code.)



```
// cc FileSystemCat Displays files from a Hadoop file
import java.io.InputStream;
import java.net.URI;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IOUtils;

// vv FileSystemCat
public class MyClass { // I rename the class name

  public static void main(String[] args) throws Except
    String uri = args[0];
    Configuration conf = new Configuration();
    FileSystem fs = FileSystem.get(URI.create(uri), c
    InputStream in = null;
    try {
      in = fs.open(new Path(uri));
      IOUtils.copyBytes(in, System.out, 4096, false);
    } finally {
      IOUtils.closeStream(in);
    }
  }
}
// ^^ FileSystemCat
```

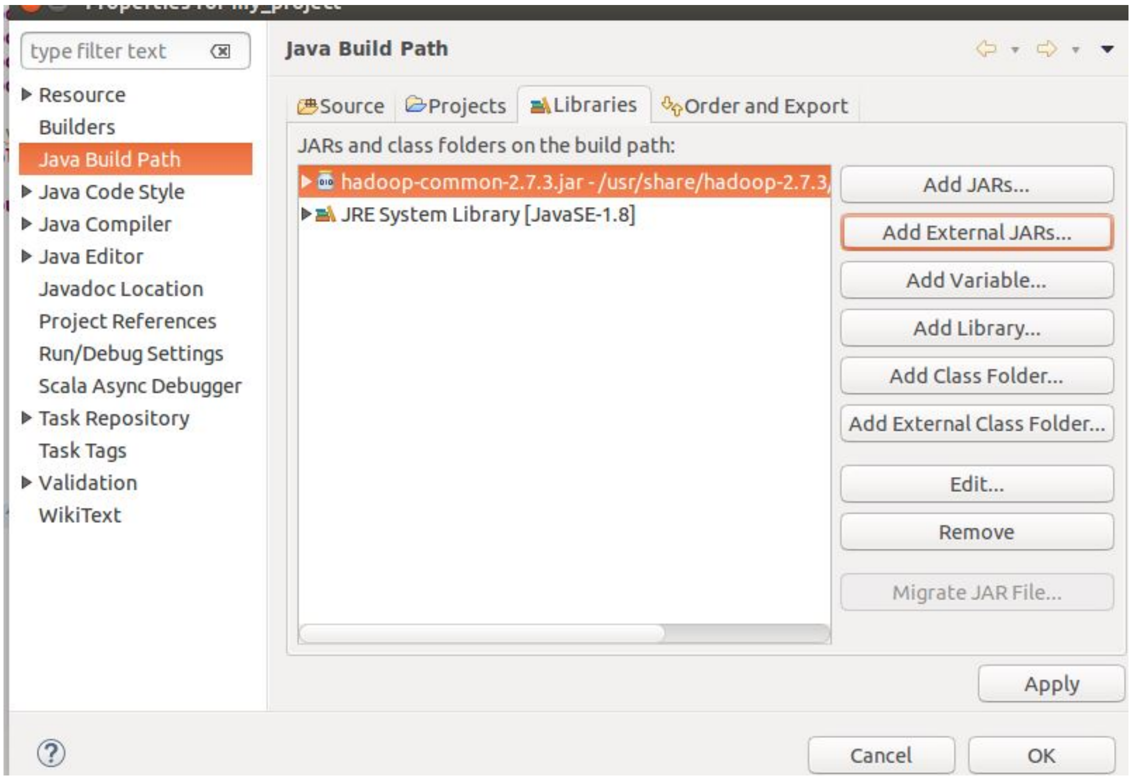
Next, you need to manually add jar files of Hadoop into the Java project environment, to resolve the errors in the “import” lines of the program:

Right click “my_project” -> “Build Path”-> “Configure Build Path” -> “Libraries” -> “Add External JARs”.

All necessary JARs (e.g., jar files) for Hadoop in this subject are included in the $HADOOP_HOME/share/hadoop folder (and its sub-folders), which come with the Hadoop installation.

Navigate to this folder.

For this application (FileSystemCat), the libraries are in the hadoop-command-2.7.3.jar in the “common” sub-folder of $HADOOP_HOME/share/hadoop. (You can type “echo $HADOOP_HOME” in Terminal to see its complete path.)

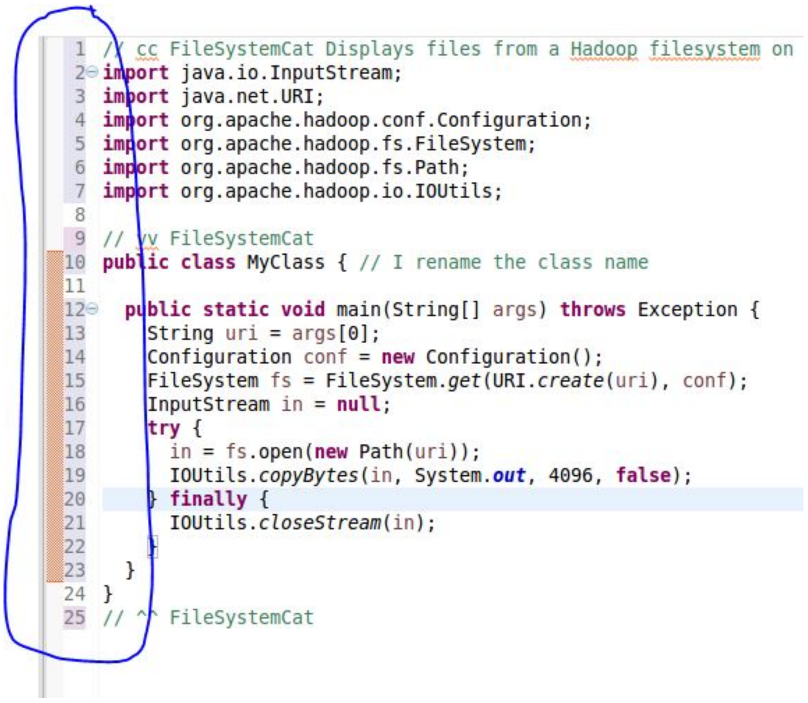


Resource
Builders
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Project References
Run/Debug Settings
Scala Async Debugger
Task Repository
Task Tags
Validation
WikiText

JARs and class folders on the build path:

- hadoop-common-2.7.3.jar - /usr/share/hadoop-2.7.3/
- JRE System Library [JavaSE-1.8]

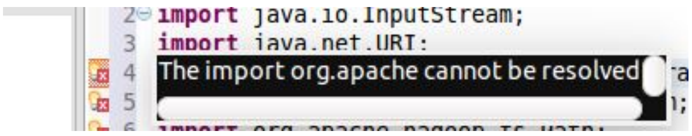
Click “OK” to import it. You will see all “error” are resolved.



```
// cc FileSystemCat Displays files from a Hadoop filesystem on
import java.io.InputStream;
import java.net.URI;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IOUtils;

// vv FileSystemCat
public class MyClass { // I rename the class name

  public static void main(String[] args) throws Exception {
    String uri = args[0];
    Configuration conf = new Configuration();
    FileSystem fs = FileSystem.get(URI.create(uri), conf);
    InputStream in = null;
    try {
      in = fs.open(new Path(uri));
      IOUtils.copyBytes(in, System.out, 4096, false);
    } finally {
      IOUtils.closeStream(in);
    }
  }
}
// ^^ FileSystemCat
```

There are many java codes provided to you and you can copy them to your Eclipse in the similar way.

However, most java programs require multiple external jar files. If you see the error “The import org.apache[.xxx.xxx] cannot be resolved” such as the following one, it means that you didn’t add all necessary jar files.



```
import java.io.InputStream;
import java.net.URI;
```

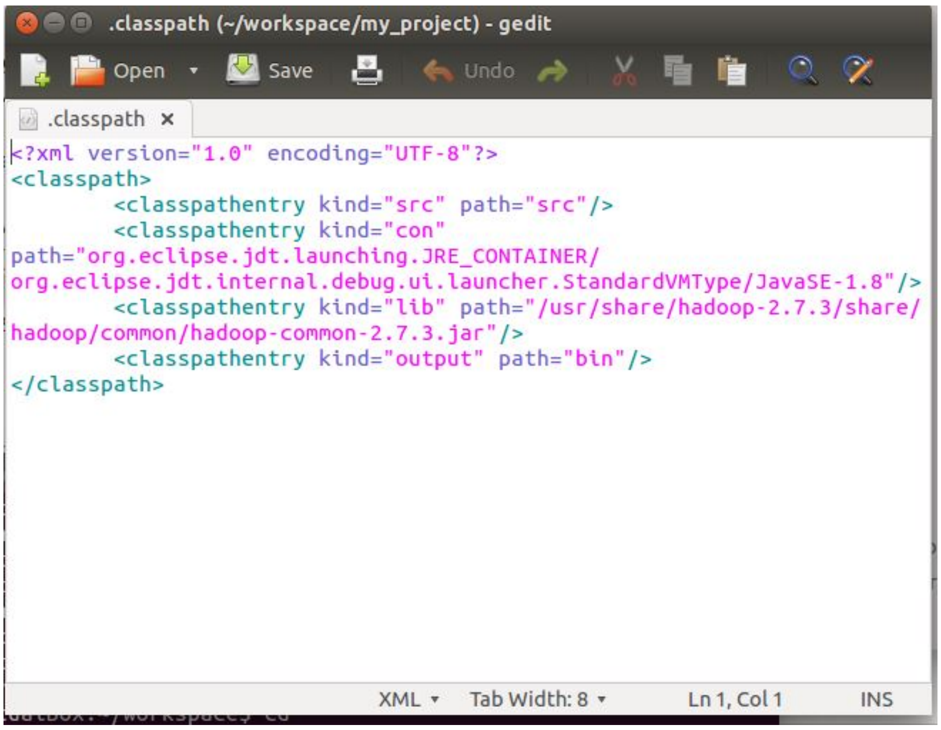
The import org.apache cannot be resolved

An “over-shooting” method is to add ALL jar files in $HADOOP_HOME/share/hadoop folder and (its sub-folders) into your java project environment at once.

Use the given .classpath file on Moodle

The jars files you added to your java project will be recorded in a hidden file called .classpath in the root directory of the project. You can use the following command in the Terminal to view it.

```
cd  workspace/my_project
ls -a
. ..  bin  .classpath  .project  .settings  src
gedit .classpath
```



.classpath (~/workspace/my_project) - gedit

```
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
        <classpathentry kind="src" path="src"/>
        <classpathentry kind="con"
path="org.eclipse.jdt.launching.JRE_CONTAINER/
org.eclipse.jdt.internal.debug.ui.launcher.StandardVMType/JavaSE-1.8"/>
        <classpathentry kind="lib" path="/usr/share/hadoop-2.7.3/share/
hadoop/common/hadoop-common-2.7.3.jar"/>
        <classpathentry kind="output" path="bin"/>
</classpath>
```

For convenience, a given classpath.txt file on Moodle. It specifies the paths to jar files that should be sufficient for your implementation tasks (but you can always include other jars you want in the above manner).

You can copy and paste information in given classpath.txt to the .classpath in your project’s root directory. Restart Eclipse and you will see all the jars have been added to your project environment.

Compilation

Once you complete editing your programs, you can copy to them a convenient folder. Then, you can compile your sourcecode and generate a jar in the command-line by following the laboratory practice in Week 2.