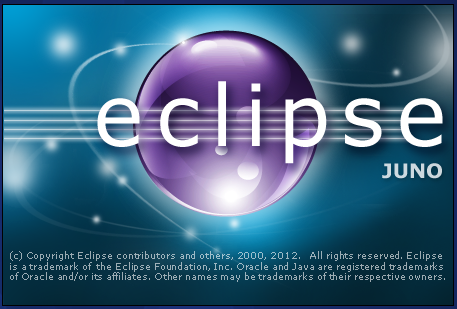
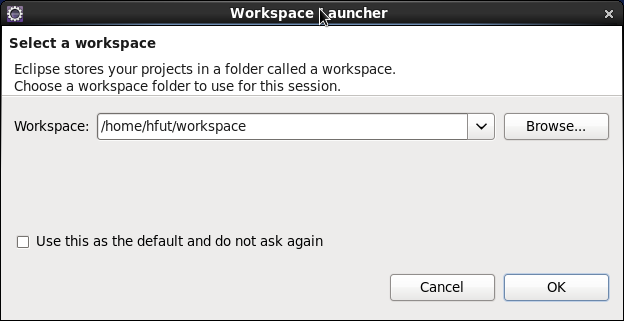
**8. 分布式文件系统HDFS上的编程实践**

**一、创建Eclipse工程**

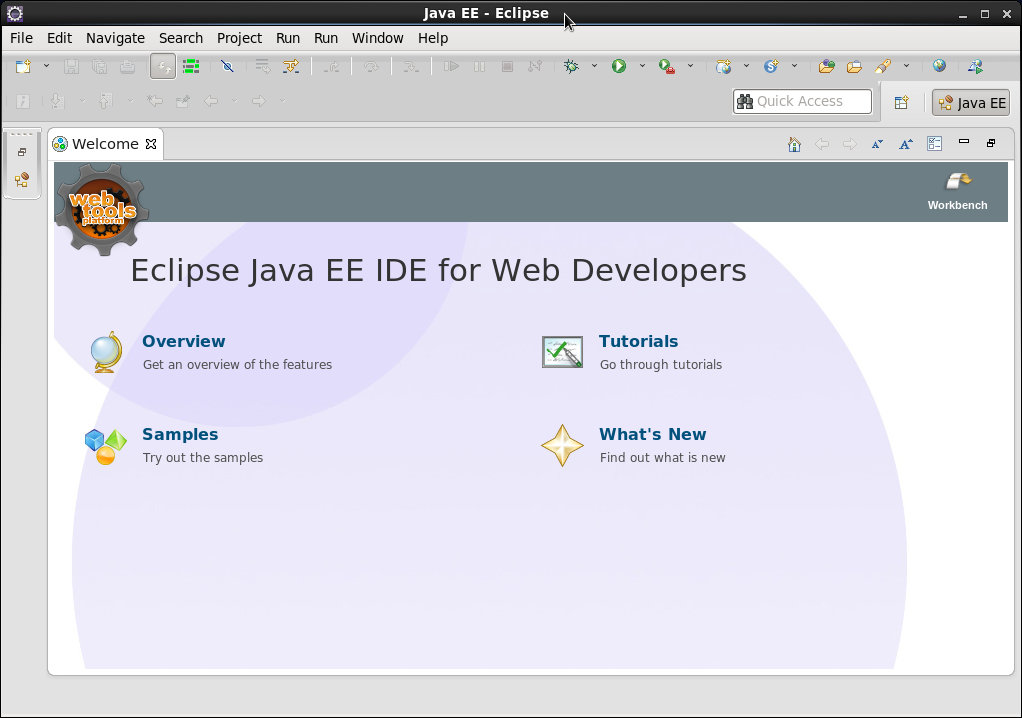
双击eclipse桌面快捷方式



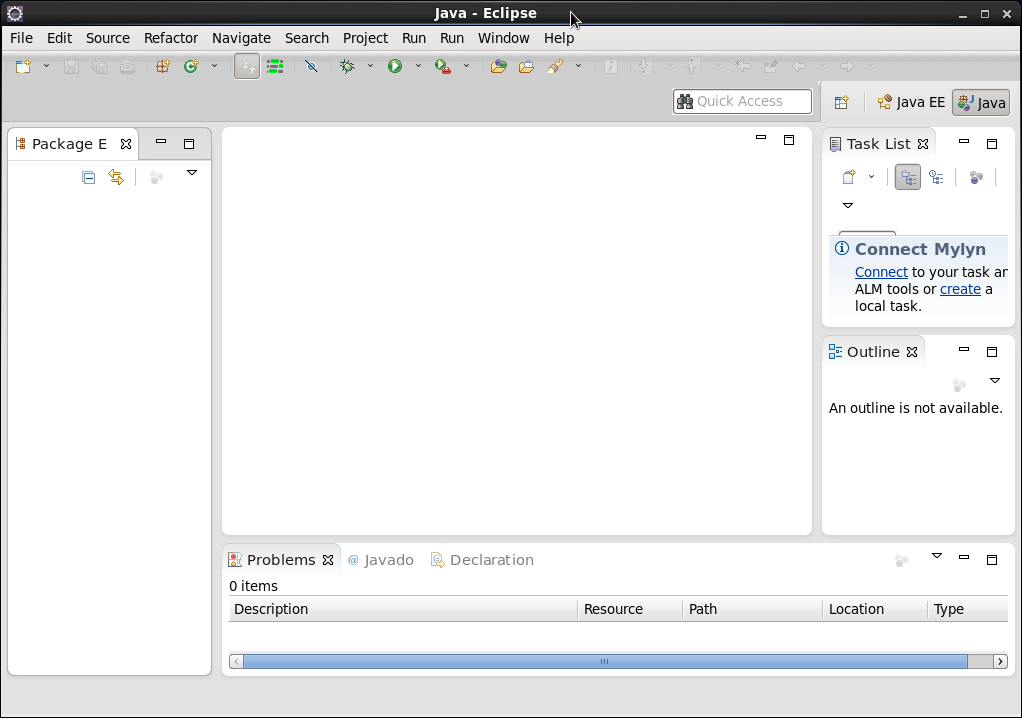
第一次打开Eclipse，需要填写workspace(工作空间)，用来保存程序所在的位置，填写“/home/hfut/workspace”，如下图：



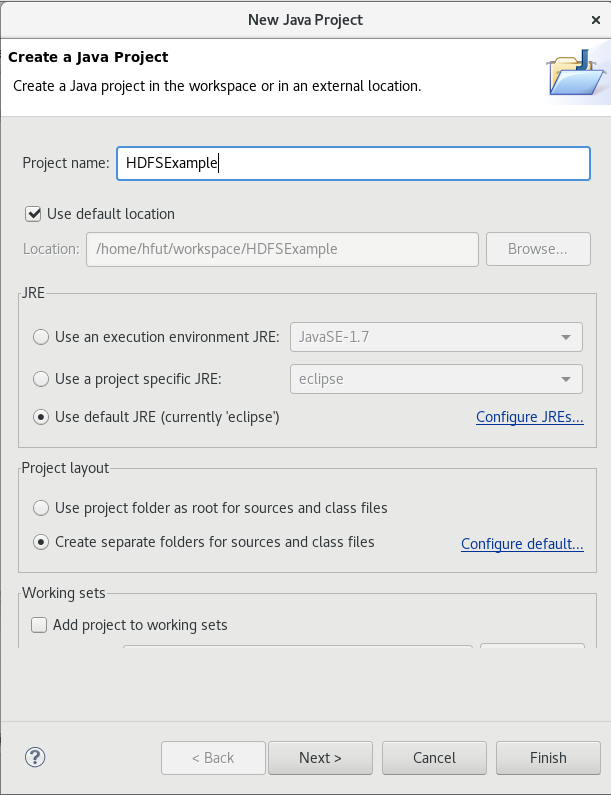
单击“OK”按钮，进入Eclipse软件。



关闭“Welcome”页面，单击菜单栏“Window->Open Persperctive->Java”切换到“Java”视图。

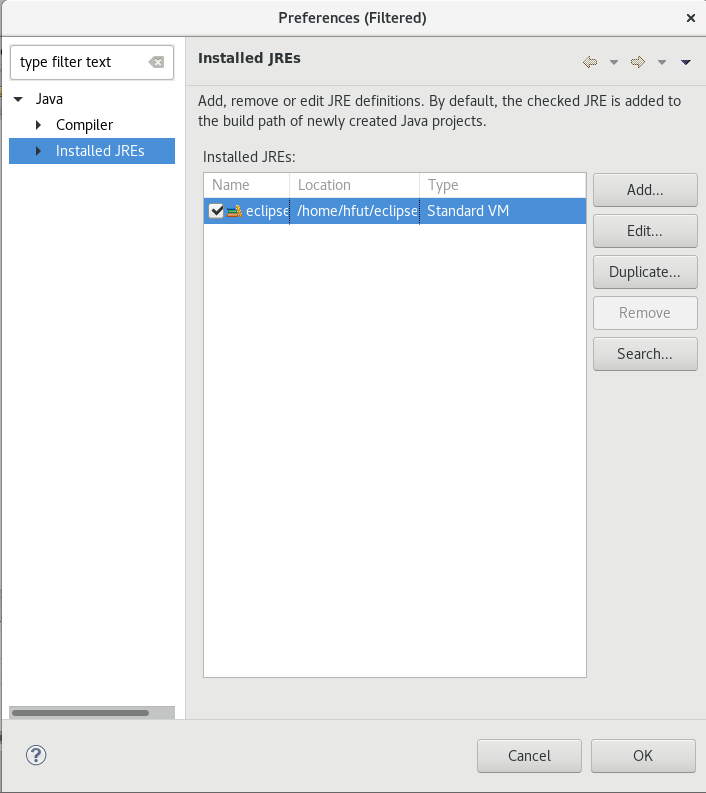


选择“File->New->Java Project”菜单，开始创建一个Java工程，弹出下图窗口。

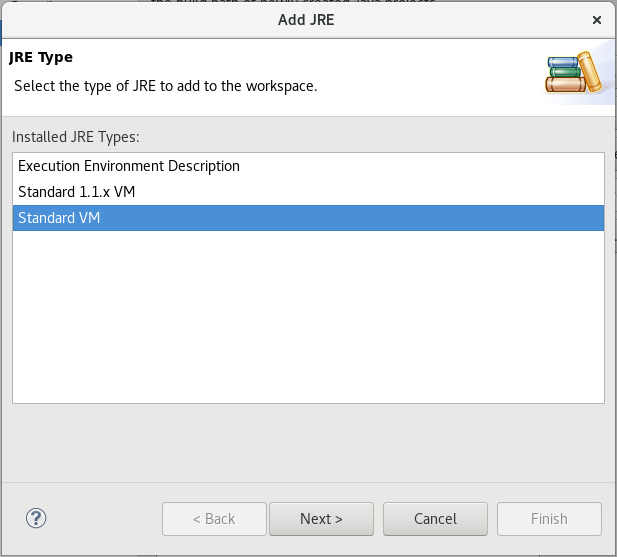


在“Project name”后面输入工程名称“HDFSExample”，使用默认的 “Create separate folders for sources and class files”， 这个Java工程的所有文件都保存到“/home/hfut/workspace/HDFSExample”目录下。

选择“Use default JRE”，点击“Configure JREs…”

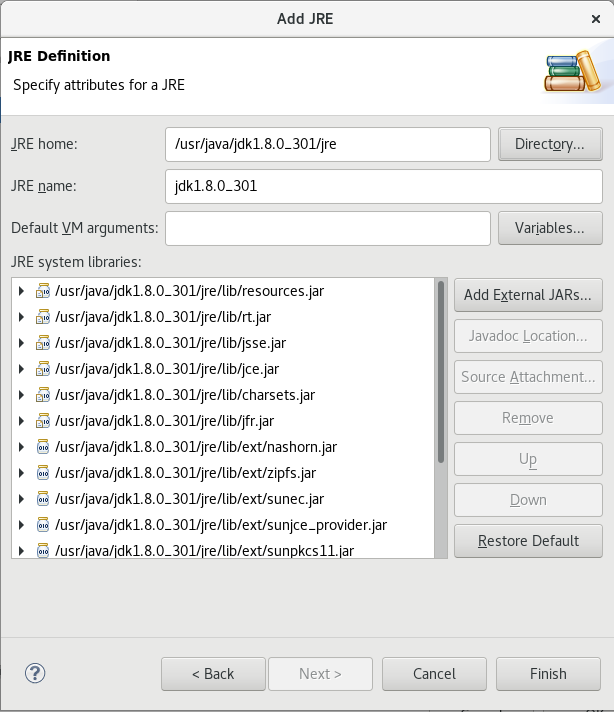


点击“Add”

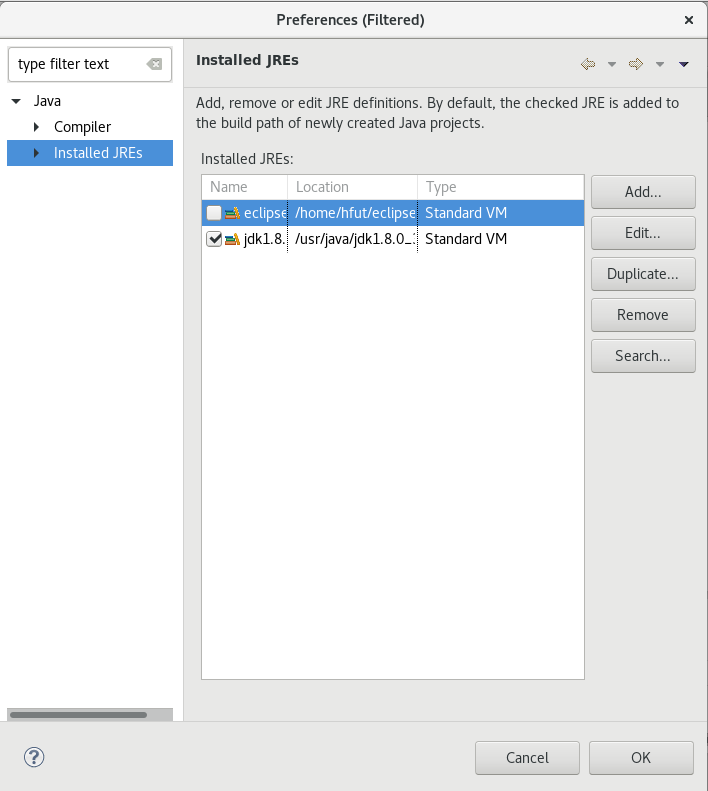


选择“Standard VM”，界面底部的“Next>”按钮，进入下一步的设置。

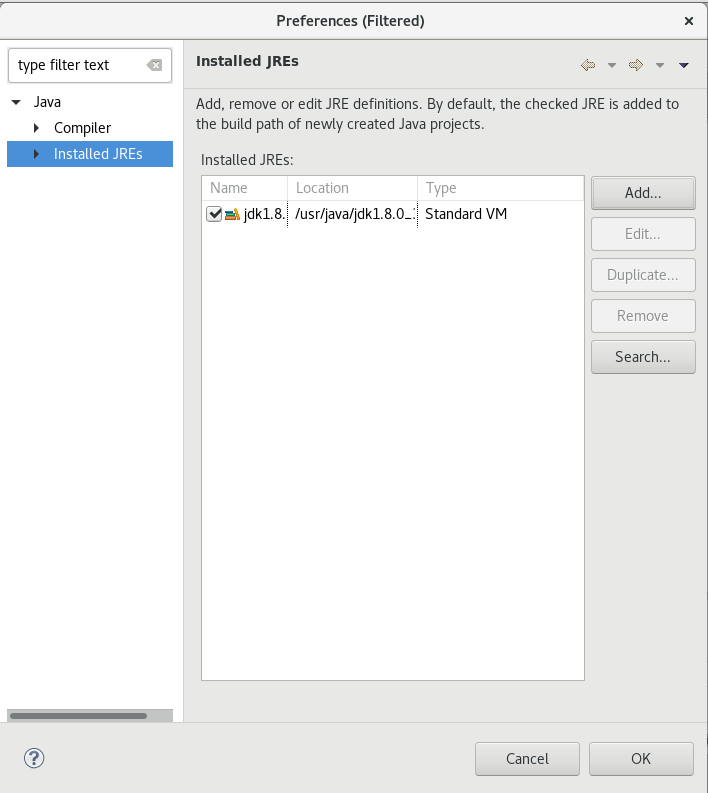
点击“Directory”，选择File System->usr->java->jdk1.8.0\_301->jre，点击界面底部的“OK”按钮



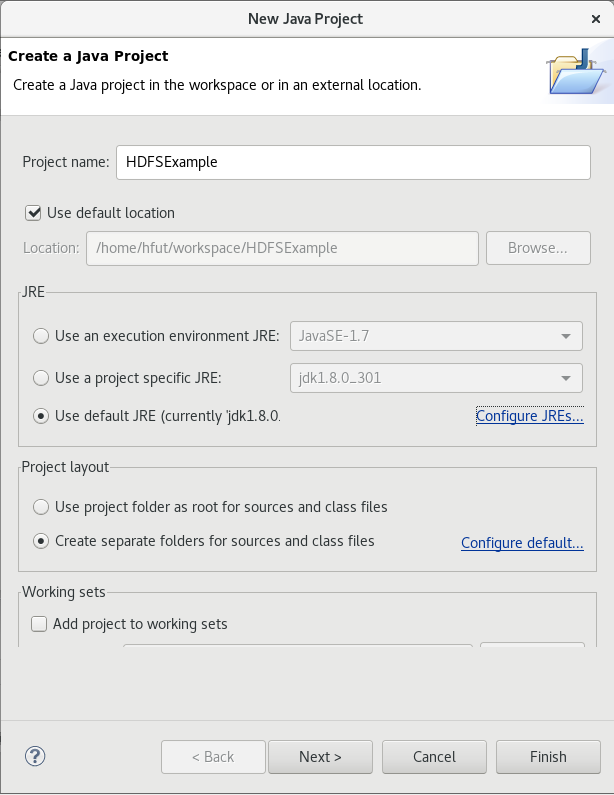
点击界面底部的“Finish”按钮



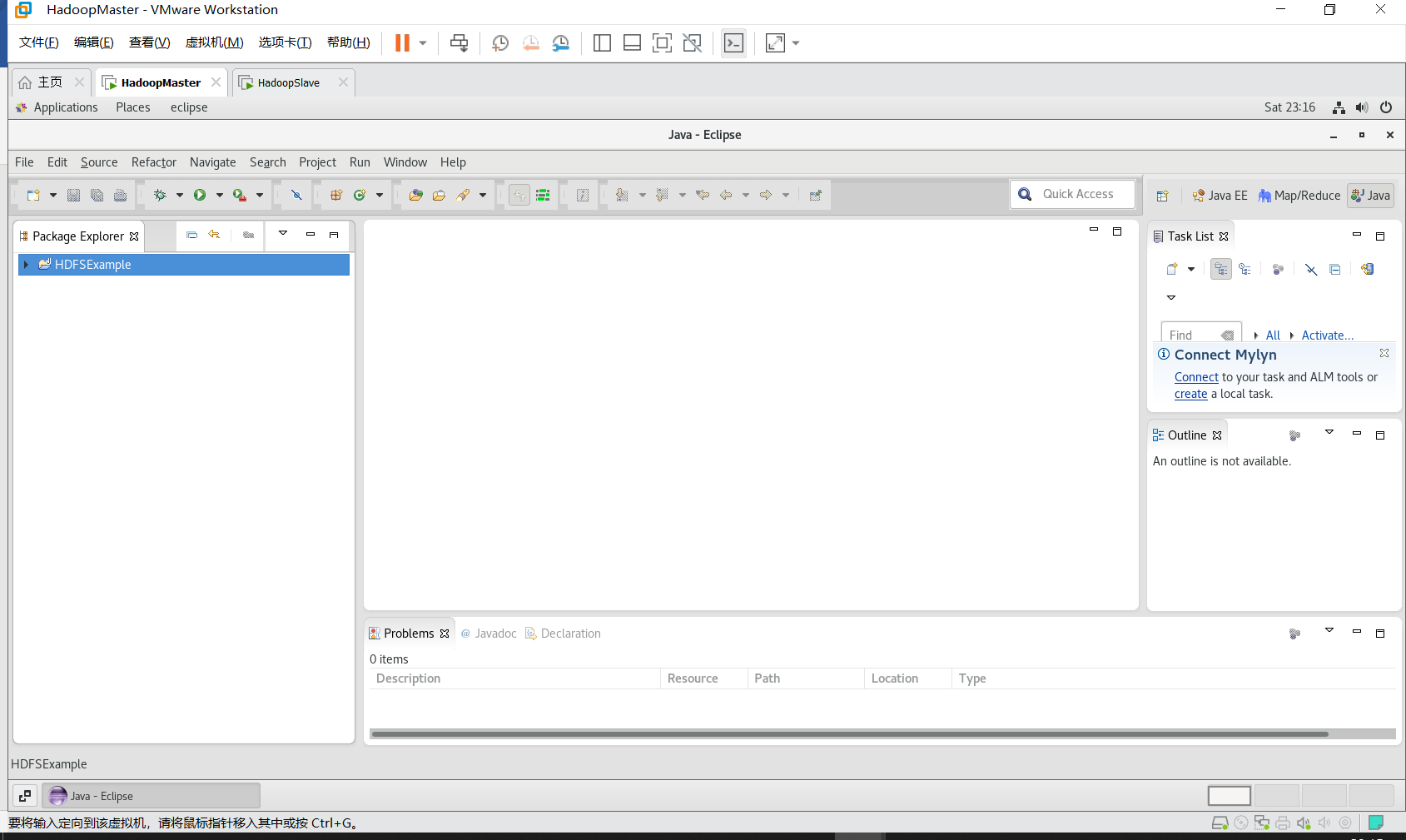
点击“Remove”按钮，删除Eclipse自带的JRE 1.7，保留新配置的JRE 1.8



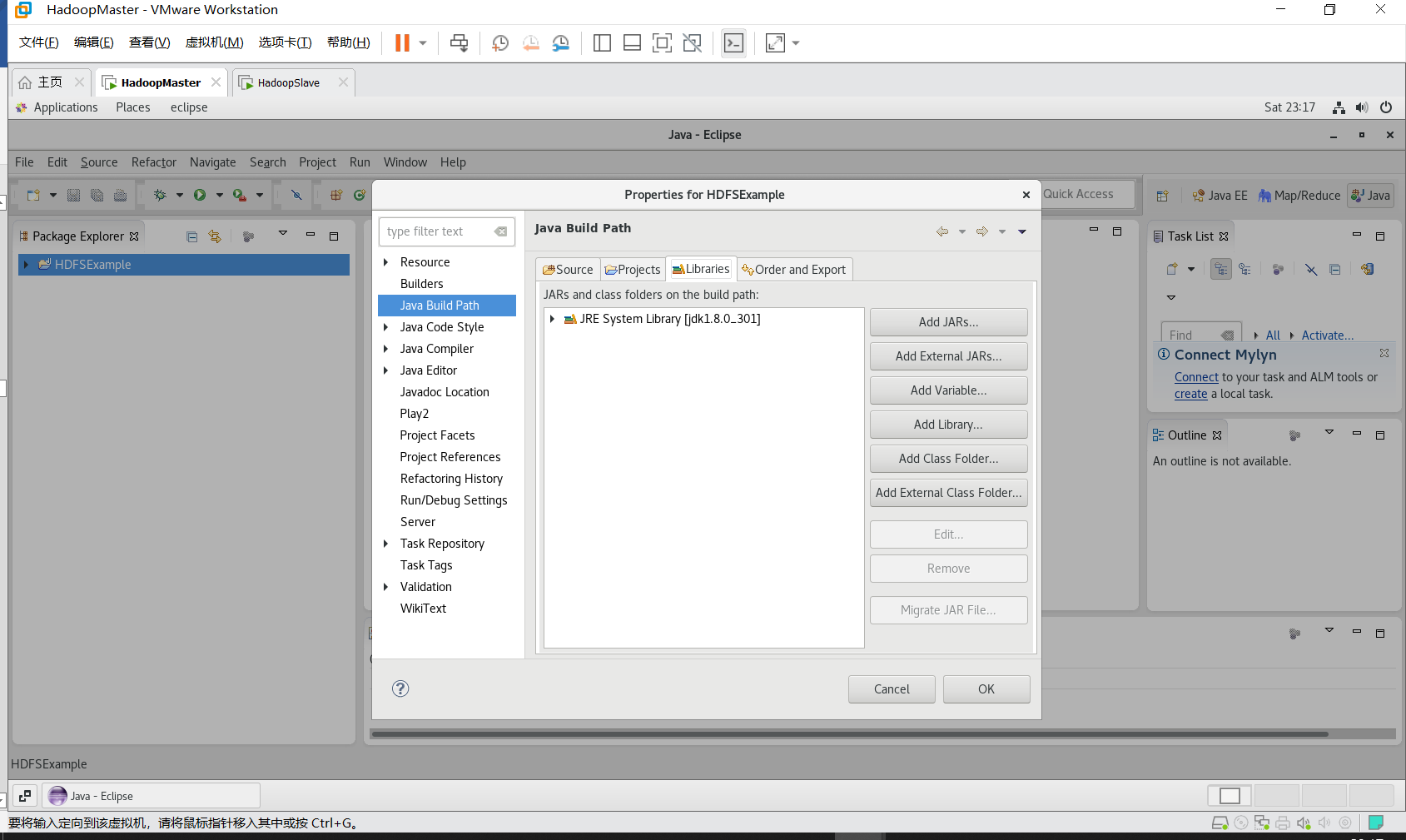
点击界面底部的“OK”按钮，进入下一步的设置。



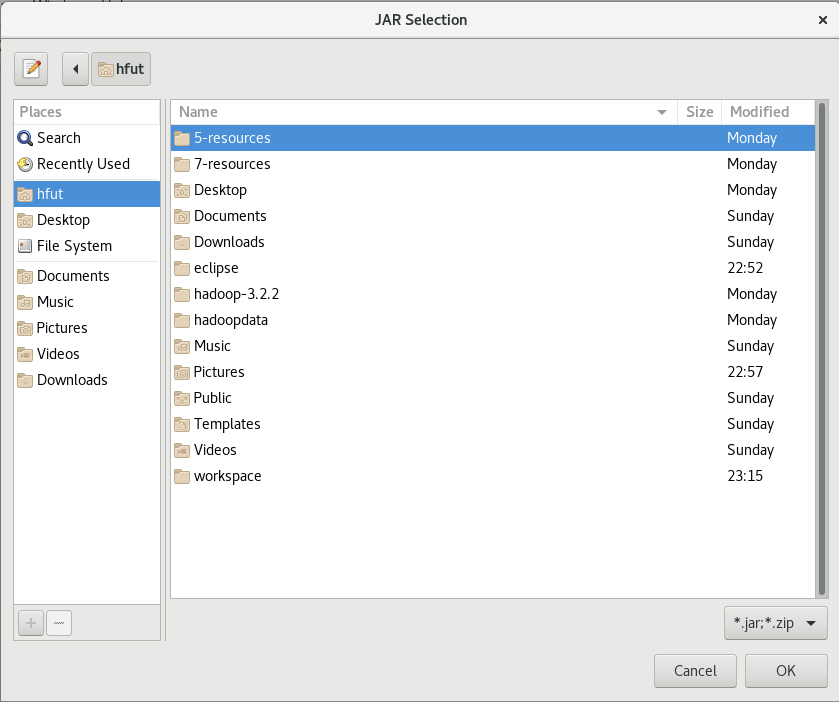
界面底部的“Finish”按钮，完成设置。



在左侧“HDFSExample”工程上依次点击右键->Build Path->Configure Build Path…->选择Libraries选项卡，出现下图的窗口，需要在这个界面中加载该Java工程所需要用到的JAR包，这些JAR包中包含了可以访问HDFS的Java API。



点击界面右侧的“Add External JARs…”按钮，会弹出下图所示窗口。



在该界面中，上面的一排目录按钮（即“hfut”、“hadoop-3.2.2”、“share”、“hadoop”，当点击某个目录按钮时，就会在下面列出该目录的内容。

**为了编写一个能够与HDFS交互的Java应用程序，一般需要向Java工程中添加以下JAR包：**

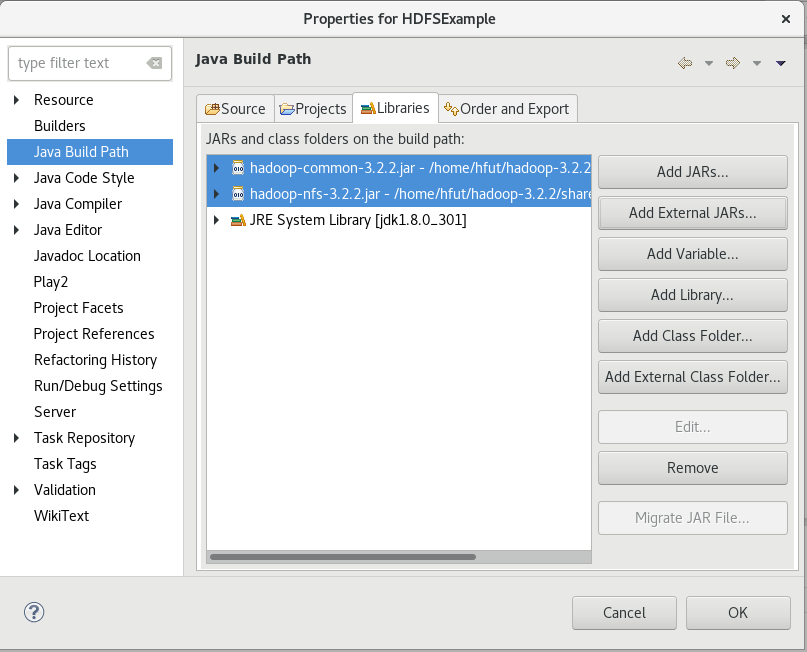
**（1）“/home/hfut/hadoop-3.2.2/share/hadoop/common”目录下的hadoop-common-3.2.2.jar和haoop-nfs-3.2.2.jar；**

**（2）“/home/hfut/hadoop-3.2.2/share/hadoop/common/lib”目录下的所有JAR包；**

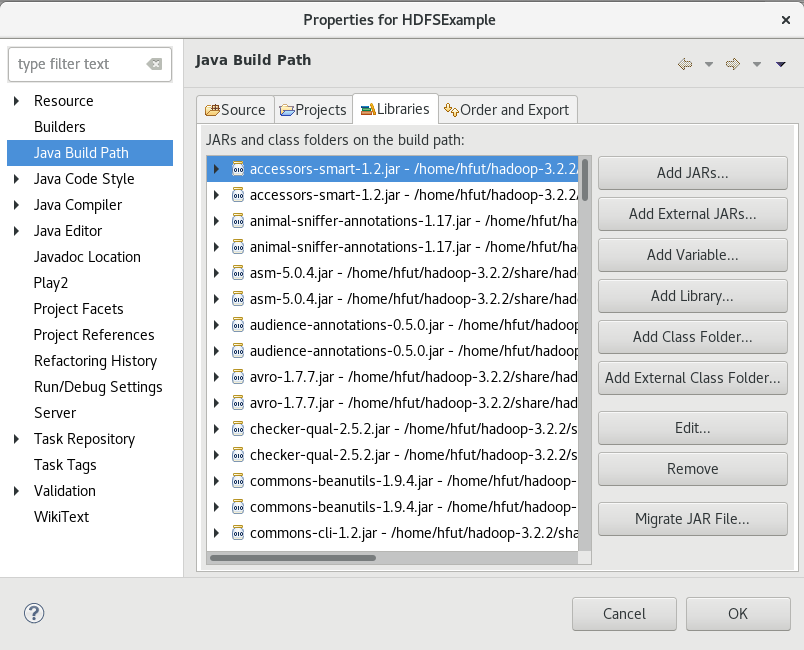
**（3）“/home/hfut/hadoop-3.2.2/share/hadoop/hdfs”目录下的haoop-hdfs-3.2.2.jar、haoop-hdfs-client-3.2.2.jar、haoop-hdfs-nfs-3.2.2.jar；**

**（4）“/home/hfut/hadoop-3.2.2/share/hadoop/hdfs/lib”目录下的所有JAR包。**

例如，如果要把“/home/hfut/hadoop-3.2.2/share/hadoop/common”目录下的hadoop-common-3.2.2.jar和haoop-nfs-3.2.2.jar添加到当前的Java工程中，可以在界面中点击目录按钮，进入到common目录，然后，界面会显示出common目录下的所有内容。



从这个界面中可以看出，hadoop-common-3.2.2.jar和haoop-nfs-3.2.2.jar已经被添加到当前Java工程中。然后，按照类似的操作方法，可以再次点击“Add External JARs…”按钮，把剩余的其他JAR包都添加进来。需要注意的是，当需要选中某个目录下的所有JAR包时，可以使用“Ctrl+A”组合键进行全选操作。



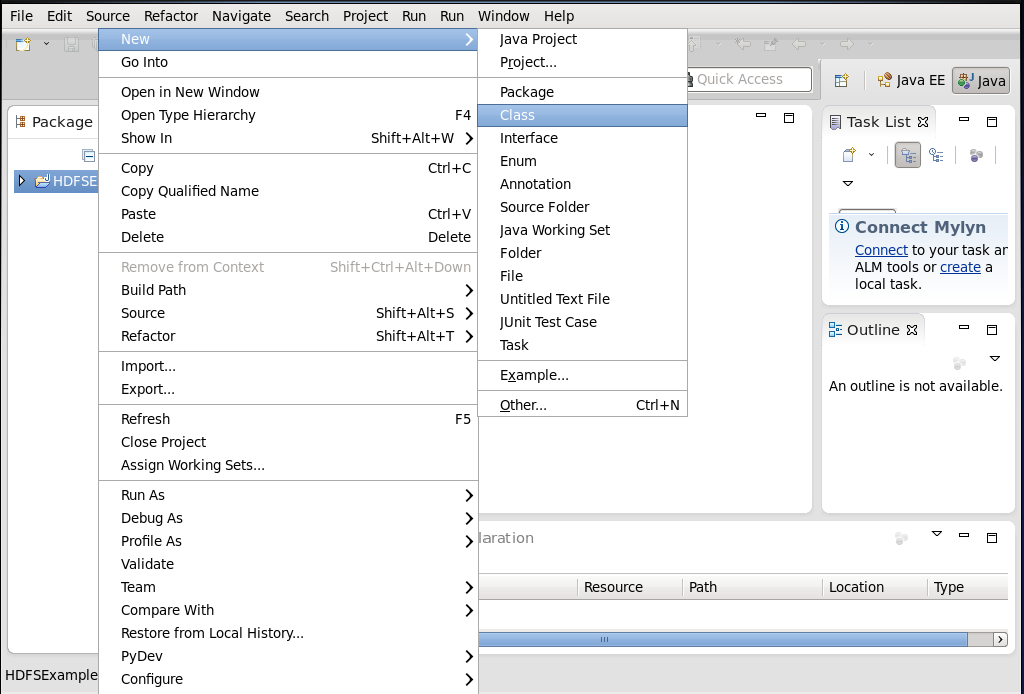
全部添加完毕以后，就可以点击界面右下角的“OK”按钮，完成配置。

**二、编写一个Java应用程序检测HDFS中是否存在一个文件**

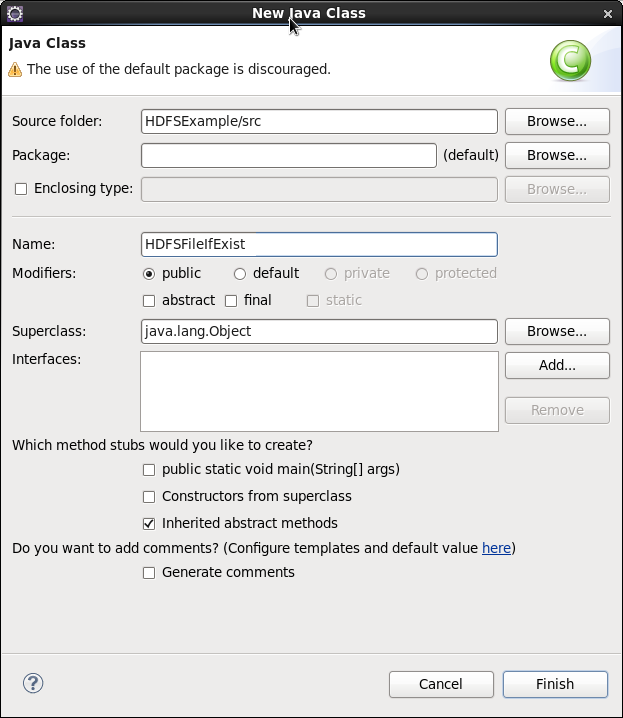
下面编写一个Java应用程序，用来检测HDFS中是否存在一个文件。

（1）编写Java程序

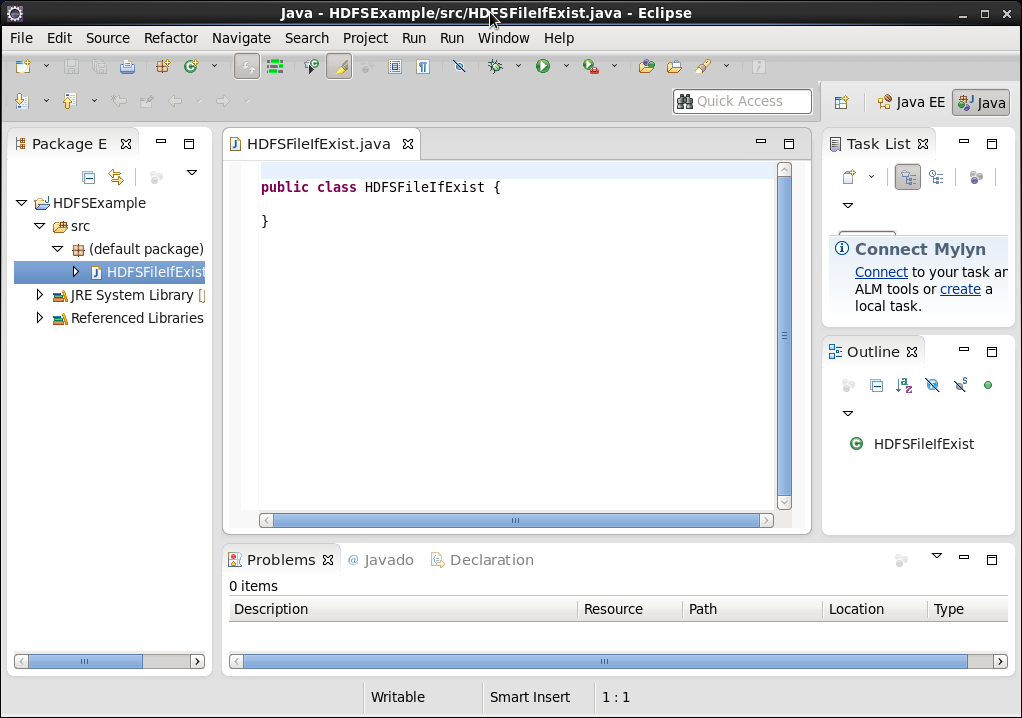
请在Eclipse工作界面左侧的“Package Explorer”面板中，找到刚才创建好的工程名称“HDFSExample”，然后在该工程名称上点击鼠标右键，在弹出的菜单中选择“New->Class”菜单。



出现如下界面：



在该界面中，只需要在“Name”后面输入新建的Java类文件的名称，这里采用名称“HDFSFileIfExist”，其他都可以采用默认设置，然后，点击界面右下角“Finish”按钮，出现下图所示界面。



可以看出，Eclipse自动创建了一个名为“HDFSFileIfExist.java”的源代码文件，请在该文件中输入以下代码：

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

public class HDFSFileIfExist {

public static void main(String[] args){

try{

String fileName = "/user/hfut/input/myLocalFile.txt";

Configuration conf = new Configuration();

conf.set("fs.defaultFS", "hdfs://master:9000");

conf.set("fs.hdfs.impl", "org.apache.hadoop.hdfs.DistributedFileSystem");

FileSystem fs = FileSystem.get(conf);

if(fs.exists(new Path(fileName))){

System.out.println("文件存在");

}else{

System.out.println("文件不存在");

}

}catch (Exception e){

e.printStackTrace();

}

}

}

该程序用来测试HDFS中是否存在一个文件，其中有一行代码：

String fileName = "/user/hfut/input/myLocalFile.txt";

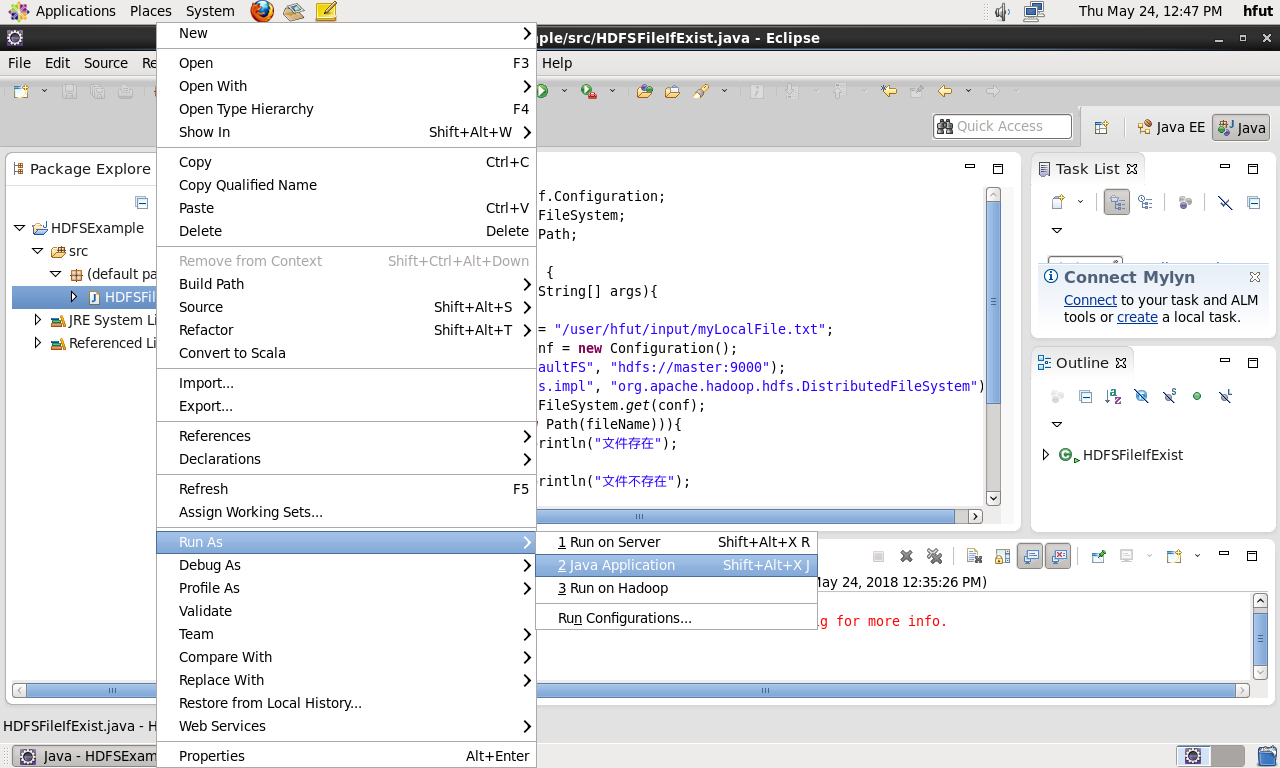
这行代码给出了需要被检测的文件名称是“myLocalFile.txt”，给出路径全称“/user/hfut/input/”，也就是测试HDFS中的“/user/hfut/input/”目录下是否存在myLocalFile.txt文件。

（2）运行程序

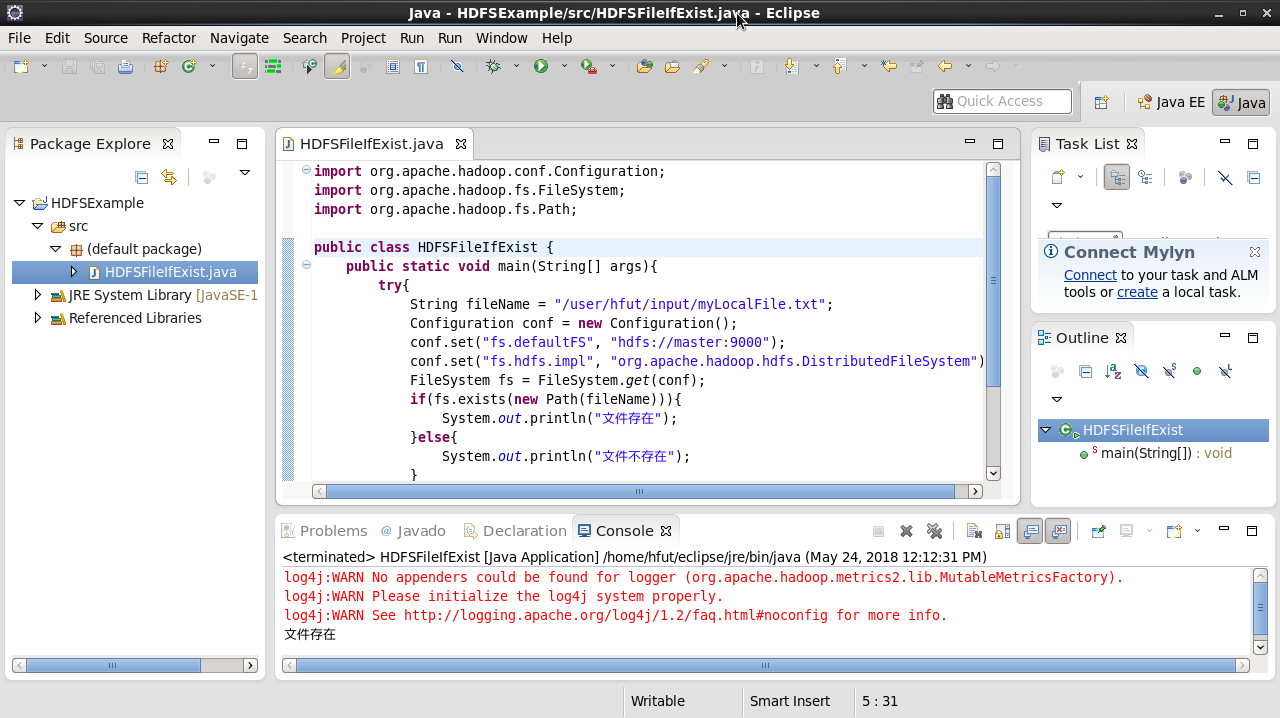
在开始编译运行程序之前，请一定确保Hadoop已经启动运行，如果还没有启动，需要打开一个Linux终端，输入以下命令启动Hadoop：

[hfut@master ~]$ start-all.sh

在左侧“Package Explore”面板的“HDFSExample”项目树上，右击“HDFSFileIfExist.java”，单击“Run As”->“2 Java Application”。



得到下图的运行结果：

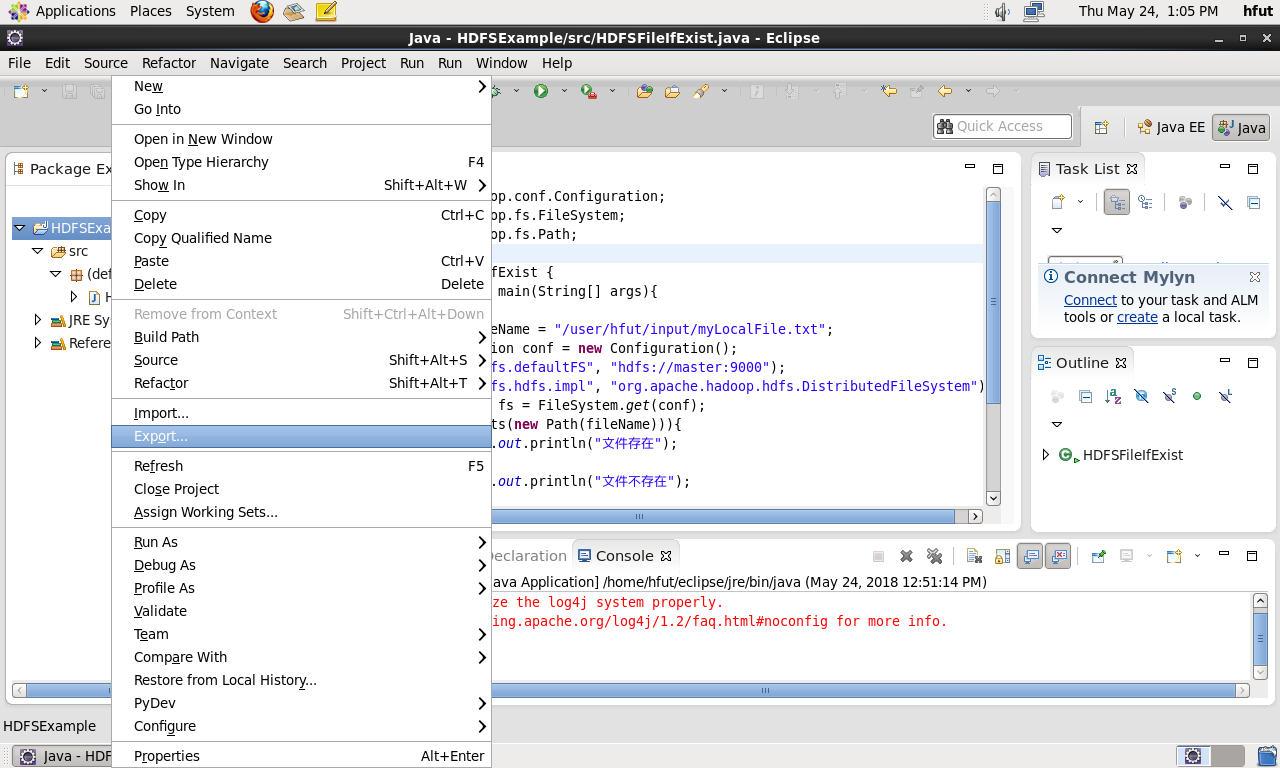


（3）部署到Hadoop平台上运行

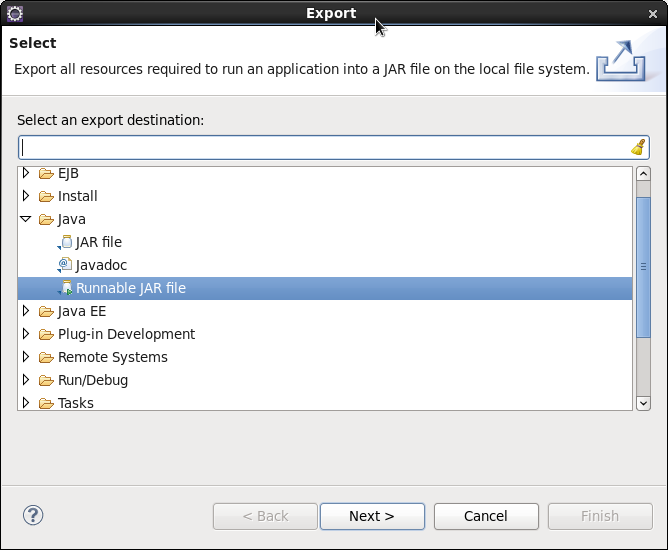
下面介绍如何把Java应用程序生成JAR包，部署到Hadoop平台上运行。首先，在Hadoop安装目录下新建一个名称为myapp的目录，用来存放我们自己编写的Hadoop应用程序，可以在Linux的终端中执行如下命令：

[hfut@master ~]$ mkdir ~/hadoop-3.2.2/myapp

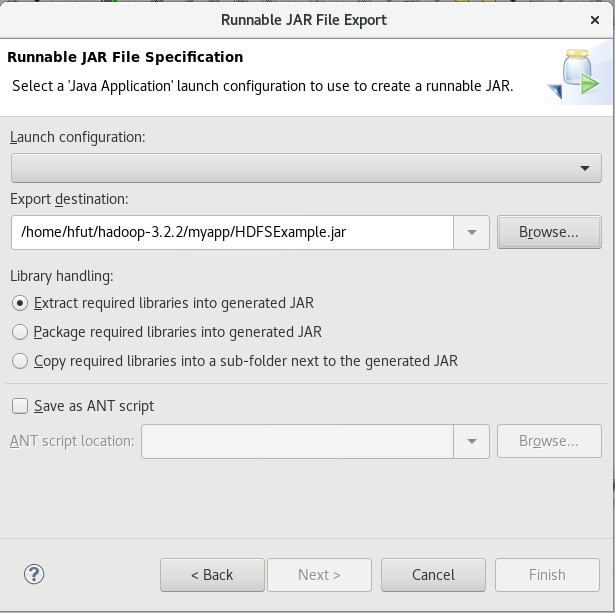
然后，请在Eclipse工作界面左侧的“Package Explorer”面板中，在工程名称“HDFSExample”上点击鼠标右键，在弹出的菜单中选择“Export”，如下图所示。



然后，会看到弹出以下界面：



在该界面中，选择“Runnable JAR file”，然后，点击“Next>”按钮，弹出如下所示界面。



在该界面中，“Launch configuration”用于设置生成的JAR包被部署启动时运行的主类，需要在下拉列表中选择刚才配置的类“HDFSFileIfExist-HDFSExample”。在“Export destination”中需要设置JAR包要输出保存到哪个目录，比如，这里设置为“/home/hfut/hadoop-3.2.2/myapp/HDFSExample.jar”。在“Library handling”下面选择“Extract required libraries into generated JAR”。然后，点击“Finish”按钮，会出现如下图所示界面。



可以忽略该界面的信息，直接点击界面右下角的“OK”按钮，启动打包过程。打包过程结束后，会出现一个警告信息界面，如下图所示。



可以忽略该界面的信息，直接点击界面右下角的“OK”按钮。至此，已经顺利把HDFSExample工程打包生成了HDFSExample.jar。可以到Linux系统中查看一下生成的HDFSExample.jar文件，可以在Linux的终端中执行如下命令：

[hfut@master ~]$ ls ~/hadoop-3.2.2/myapp/

可以看到，“~/hadoop-3.2.2/myapp/”目录下已经存在一个HDFSExample.jar文件。现在，就可以在Linux系统中，使用hadoop jar命令运行程序，命令如下：

[hfut@master ~]$ hadoop jar ~/hadoop-3.2.2/myapp/HDFSExample.jar

或

[hfut@master ~]$ java -jar ~/hadoop-3.2.2/myapp/HDFSExample.jar

命令执行结束后，会在屏幕上显示执行结果“文件存在”。

至此，检测HDFS文件是否存在的程序，就顺利部署完成了。

**三、编写一个Java应用程序读/写HDFS文件**

（1）读取HDFS文件程序

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.fs.FSDataInputStream;

**public** **class** HDFSReadFile {

**public** **static** **void** main(String[] args) {

**try** {

Configuration conf = **new** Configuration();

conf.set("fs.defaultFS", "hdfs://master:9000");

conf.set("fs.hdfs.impl",

"org.apache.hadoop.hdfs.DistributedFileSystem");

FileSystem fs = FileSystem.*get*(conf);

Path file = **new** Path("/user/hfut/input/myLocalFile.txt");

FSDataInputStream getIt = fs.open(file);

BufferedReader d = **new** BufferedReader(**new** InputStreamReader(getIt));

String content = d.readLine(); // 读取文件一行

System.*out*.println(content);

d.close(); // 关闭文件

fs.close(); // 关闭hdfs

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

（2）写HDFS文件程序

**import** org.apache.hadoop.conf.Configuration;

**import** org.apache.hadoop.fs.FileSystem;

**import** org.apache.hadoop.fs.FSDataOutputStream;

**import** org.apache.hadoop.fs.Path;

**public** **class** HDFSWriteFile {

**public** **static** **void** main(String[] args) {

**try** {

Configuration conf = **new** Configuration();

conf.set("fs.defaultFS", "hdfs://master:9000");

conf.set("fs.hdfs.impl",

"org.apache.hadoop.hdfs.DistributedFileSystem");

FileSystem fs = FileSystem.*get*(conf);

**byte**[] buff = "Hello world".getBytes(); // 要写入的内容

String filename = "/user/hfut/test.txt"; // 要写入的文件名

FSDataOutputStream os = fs.create(**new** Path(filename));

os.write(buff, 0, buff.length);

System.*out*.println("Create:" + filename);

os.close();

fs.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

}