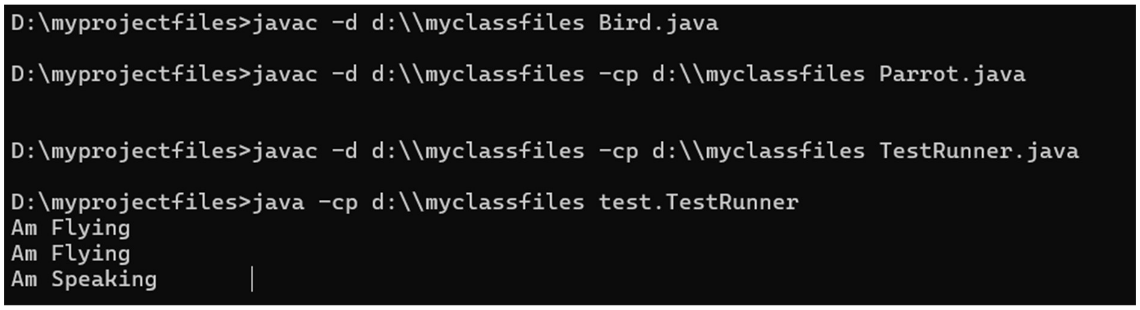
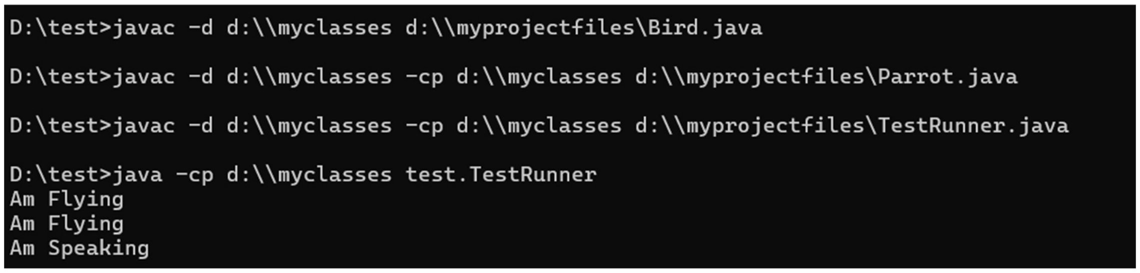
OUTPUT OF EXERCISE 1



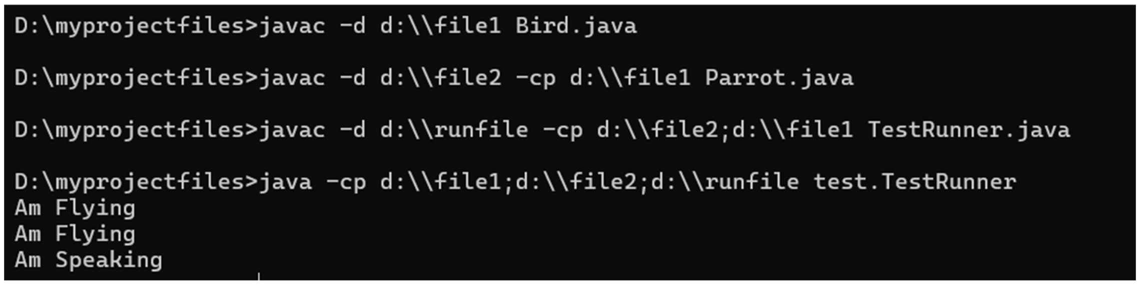
```
D:\myprojectfiles>javac -d d:\\myclassfiles Bird.java

D:\myprojectfiles>javac -d d:\\myclassfiles -cp d:\\myclassfiles Parrot.java


D:\myprojectfiles>javac -d d:\\myclassfiles -cp d:\\myclassfiles TestRunner.java

D:\myprojectfiles>java -cp d:\\myclassfiles test.TestRunner
Am Flying
Am Flying
Am Speaking
```

OUTPUT OF EXERCISE 2



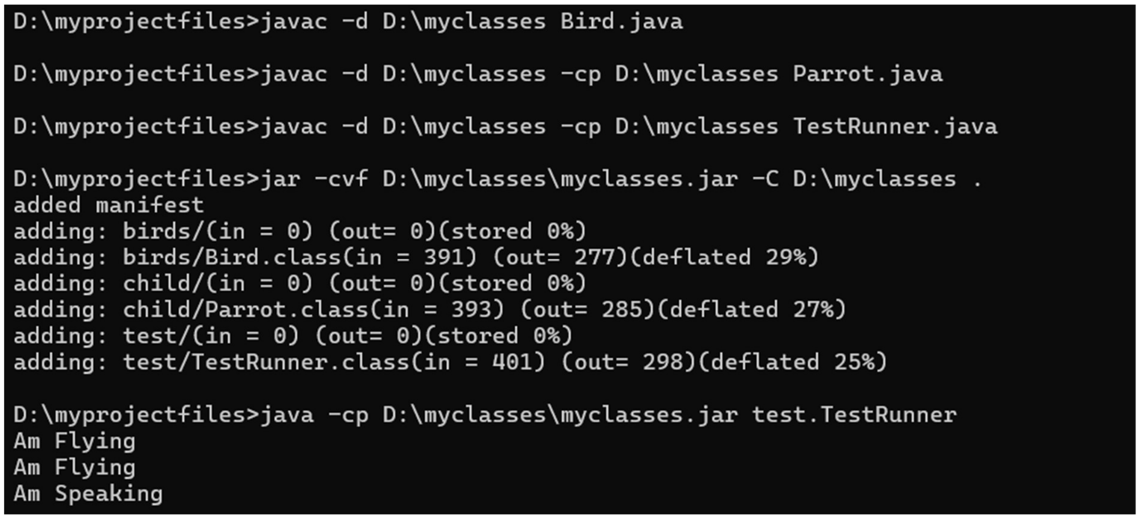
```
D:\test>javac -d d:\\myclasses d:\\myprojectfiles\Bird.java

D:\test>javac -d d:\\myclasses -cp d:\\myclasses d:\\myprojectfiles\Parrot.java

D:\test>javac -d d:\\myclasses -cp d:\\myclasses d:\\myprojectfiles\TestRunner.java

D:\test>java -cp d:\\myclasses test.TestRunner
Am Flying
Am Flying
Am Speaking
```

OUTPUT OF EXERCISE 3



```
D:\myprojectfiles>javac -d d:\\file1 Bird.java

D:\myprojectfiles>javac -d d:\\file2 -cp d:\\file1 Parrot.java

D:\myprojectfiles>javac -d d:\\runfile -cp d:\\file2;d:\\file1 TestRunner.java

D:\myprojectfiles>java -cp d:\\file1;d:\\file2;d:\\runfile test.TestRunner
Am Flying
Am Flying
Am Speaking
```

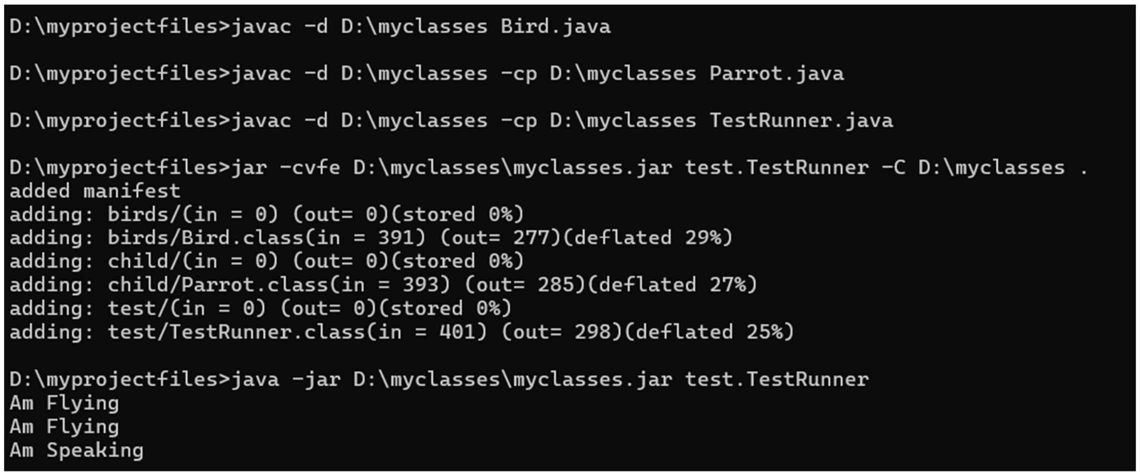
OUTPUT OF EXERCISE 4



```
D:\myprojectfiles>javac -d D:\myclasses Bird.java

D:\myprojectfiles>javac -d D:\myclasses -cp D:\myclasses Parrot.java

D:\myprojectfiles>javac -d D:\myclasses -cp D:\myclasses TestRunner.java

D:\myprojectfiles>jar -cvf D:\myclasses\myclasses.jar -C D:\myclasses .
added manifest
adding: birds/(in = 0) (out= 0)(stored 0%)
adding: birds/Bird.class(in = 391) (out= 277)(deflated 29%)
adding: child/(in = 0) (out= 0)(stored 0%)
adding: child/Parrot.class(in = 393) (out= 285)(deflated 27%)
adding: test/(in = 0) (out= 0)(stored 0%)
adding: test/TestRunner.class(in = 401) (out= 298)(deflated 25%)

D:\myprojectfiles>java -cp D:\myclasses\myclasses.jar test.TestRunner
Am Flying
Am Flying
Am Speaking
```

After creating jar, we can specify the directory for the classes inside jar using classpath.

Using Manifest File :



```
D:\myprojectfiles>javac -d D:\myclasses Bird.java

D:\myprojectfiles>javac -d D:\myclasses -cp D:\myclasses Parrot.java

D:\myprojectfiles>javac -d D:\myclasses -cp D:\myclasses TestRunner.java

D:\myprojectfiles>jar -cvfe D:\myclasses\myclasses.jar test.TestRunner -C D:\myclasses .
added manifest
adding: birds/(in = 0) (out= 0)(stored 0%)
adding: birds/Bird.class(in = 391) (out= 277)(deflated 29%)
adding: child/(in = 0) (out= 0)(stored 0%)
adding: child/Parrot.class(in = 393) (out= 285)(deflated 27%)
adding: test/(in = 0) (out= 0)(stored 0%)
adding: test/TestRunner.class(in = 401) (out= 298)(deflated 25%)

D:\myprojectfiles>java -jar D:\myclasses\myclasses.jar test.TestRunner
Am Flying
Am Flying
Am Speaking
```

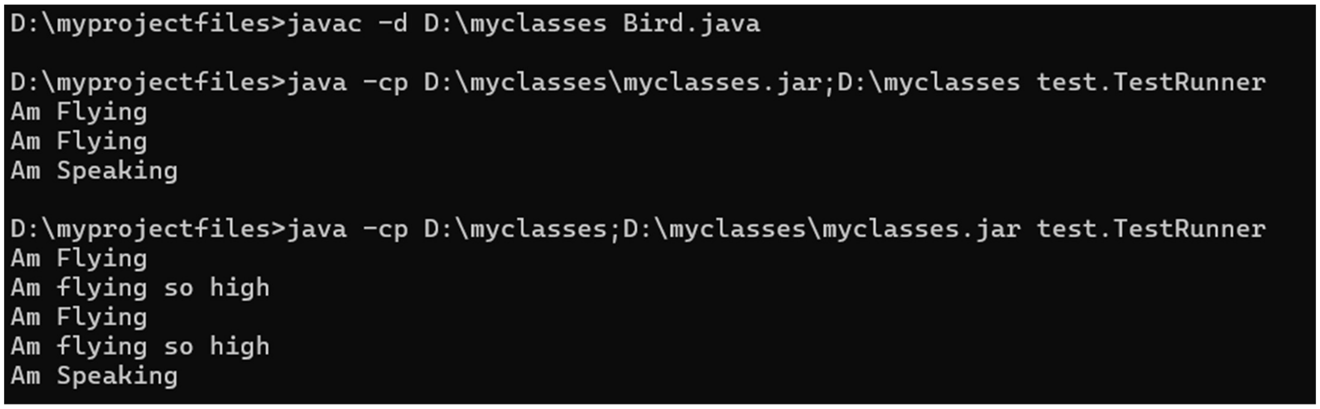
Without -e operation modifier (entry point) , I faced an Error, "no main manifest attribute, in myclasses.jar". Upon surfing google I found 2 methods to overcome it.

One is to extract the existing MANIFEST.MF file and specific the Main Class to it by editing it manually and then updating the jar.

Other is the usage of -e to directly specific the Main Class attribute during the time of creation of the Jar.

Here, I have used -e operation modifier to add the Main-Class attribute to the MANIFEST.MF file without manually editing it.

OUTPUT OF EXERCISE 5



```
D:\myprojectfiles>javac -d D:\myclasses Bird.java

D:\myprojectfiles>java -cp D:\myclasses\myclasses.jar;D:\myclasses test.TestRunner
Am Flying
Am Flying
Am Speaking

D:\myprojectfiles>java -cp D:\myclasses;D:\myclasses\myclasses.jar test.TestRunner
Am Flying
Am flying so high
Am Flying
Am flying so high
Am Speaking
```

When jar file is typed first in the classpath, older Bird.class (without the new print statement) inside the JAR is used.
When D:\myclasses is typed first in the classpath, new Bird.class is used and got the updated output which has "Am flying so high".