Packages and Interfaces

Interface

- An interface is basically a kind of class.
- Like classes, interfaces contains methods and variables.
- Difference : interfaces define only abstract methods and final variables.
- Interface do not specify any code to implement the methods.
- Interfaces are designed to support dynamic method resolution at runtime.

Defining an interface

```
access interface name
 {
      return-type method-name1(parameter-list);
       ……….
       return-type method-nameN(parameter-list);

      type final-varname1 = value;
        ……….
       type final-varnameN = value;
  }
```

Methods are declared with no bodies.

All variables are implicitly **final** and **static.**

```
interface interfacename
{
        variable declaration
        method declaration
}
```

Example:

```
 interface item
{
        static final int code=1000;
          void display();
 }
```

Example

```
interface Callback
{
   void callback(int param);
}
```

Implementing Interfaces

access class classname [extends superclass] [implements *interface [,interface...]]*

{

```
        // class-body
```

}

```
class Client implements Callback
{
    public void callback(int p)
   {
            System.out.println("callback called with " + p);
   }
}
```

When we implement an interface method, it must be declared as public.

We can also define additional members.

```
class Client implements Callback
 {
       public void callback(int p)
      {
            System.out.println("callback called with " + p);
      }
     void nonIfaceMeth()
     {
            System.out.println(“Added method");
     }
}
```

Accessing Implementations Through Interface References

```
class TestIface
 {
   public static void main(String args[])
   {
            Callback  c = new Client();
            c.callback(42);
   }
}
```

c can be used to access the callback() method, it can not access any other members of the Client class.

```
// Another implementation of Callback.

class AnotherClient implements Callback
{
   public void callback(int p)
   {
            System.out.println("Another version of callback“+(p+10));
   }
}
```

```
class TestIface2
{
   public static void main(String args[])
   {
            Callback c = new Client();
            AnotherClient ob = new AnotherClient();
            c.callback(42);

             c = ob; // c now refers to AnotherClient object
            c.callback(42);
   }
}
```

OUTPUT: callback called with 42

another version of callback 52.

Partial Implementations

If a class includes an interface but does not fully implement the methods defined by that interface, then that class must be declared as **abstract.**

```
abstract class Incomplete implements Callback
  {
        int a, b;
        void show()
        {
              System.out.println(a + " " + b);
        }
        // ...
  }
```

Applying Interfaces

```
// Define an integer stack interface.
interface   IntStack
{
    void push(int item); // store an item
    int pop(); // retrieve an item
}
```

```
class FixedStack implements IntStack
{
        private int stck[];
        private int tos;
        FixedStack(int size)
       {      stck = new int[size];
              tos = -1;
        }
        public void push(int item)
       {      if(tos==stck.length-1) // use length member
              System.out.println("Stack is full.");
              else
              stck[++tos] = item;
       }
```

```
    // Pop an item from the stack
    public int pop()
    {
            if(tos < 0)
            {           System.out.println("Stack underflow.");
                         return 0;
            }
            else
            return stck[tos--];
    }
}
```

```
class IFTest
{       public static void main(String args[])
        {       FixedStack mystack1 = new FixedStack(5);
                FixedStack mystack2 = new FixedStack(8);
                for(int i=0; i<5; i++) mystack1.push(i);
                for(int i=0; i<8; i++) mystack2.push(i);
                System.out.println("Stack in mystack1:");
                for(int i=0; i<5; i++)
                System.out.println(mystack1.pop());
                System.out.println("Stack in mystack2:");
                for(int i=0; i<8; i++)
                System.out.println(mystack2.pop());
   }
}
```

Variables in Interfaces

We can use interfaces to import shared constants into multiple classes.

```
interface SharedConstants
{               int NO = 0;
                int YES = 1;
                int MAYBE = 2;
                int LATER = 3;
}
Class A implements  SharedConstants
{       int x;
       A() {   x=1; }
      void  fun()
      {         if (x > 1 ) return NO; else return YES;      }
}
```

Interfaces Can Be Extended

```
interface A
 {
        void meth1();
}

interface B extends A
{
   void meth2();
}
```

```
// This class must implement all of A and B
class MyClass implements B
{
        public void meth1()
       {             System.out.println("Implement meth1().");
        }

        public void meth2()
       {             System.out.println("Implement meth2().");
       }
}
```

```
class prg
 {      public static void main(String arg[])
       {              MyClass ob = new MyClass();
                      ob.meth1();
                     ob.meth2();
        }
}
```

Any class that implements an interface must implement all methods defined by that interface.

Create an interface Area with following method

```
void findArea(int, int);
```

Create a two class namely Rectangle and Triangle which implements Area interface.

Write a java program to find the area of rectangle and triangle.

```
interface Area
{
    float findArea(int x, int y);
}

class Rectangle implements Area
{
   public float findArea(int x, int y)
   {
            return (x*y);
   }
}
```

```
class Triangle implements Area
{
   public float findArea(int x, int y)
   {
            return (0.5*x*y);
   }
}
```

```
class interfacePrg
{      public static void main(String args[])
       {
            Rectangle r = new Rectangle();
            Triangle  t = new Triangle();
            Area a;
            a= r;
            System.out.println("Area of rectangle ="+a.findArea(10,20));
            a= t;
            System.out.println("Area of triangle ="+a.findArea(5,10));
   }
}
```

Create an interface Vehicle with 2 methods

```
void numberOfSeats();
void numberOfWheel();
```

Create a two class namely car and bike which implements vehicle interface.

Why do we use interface ?

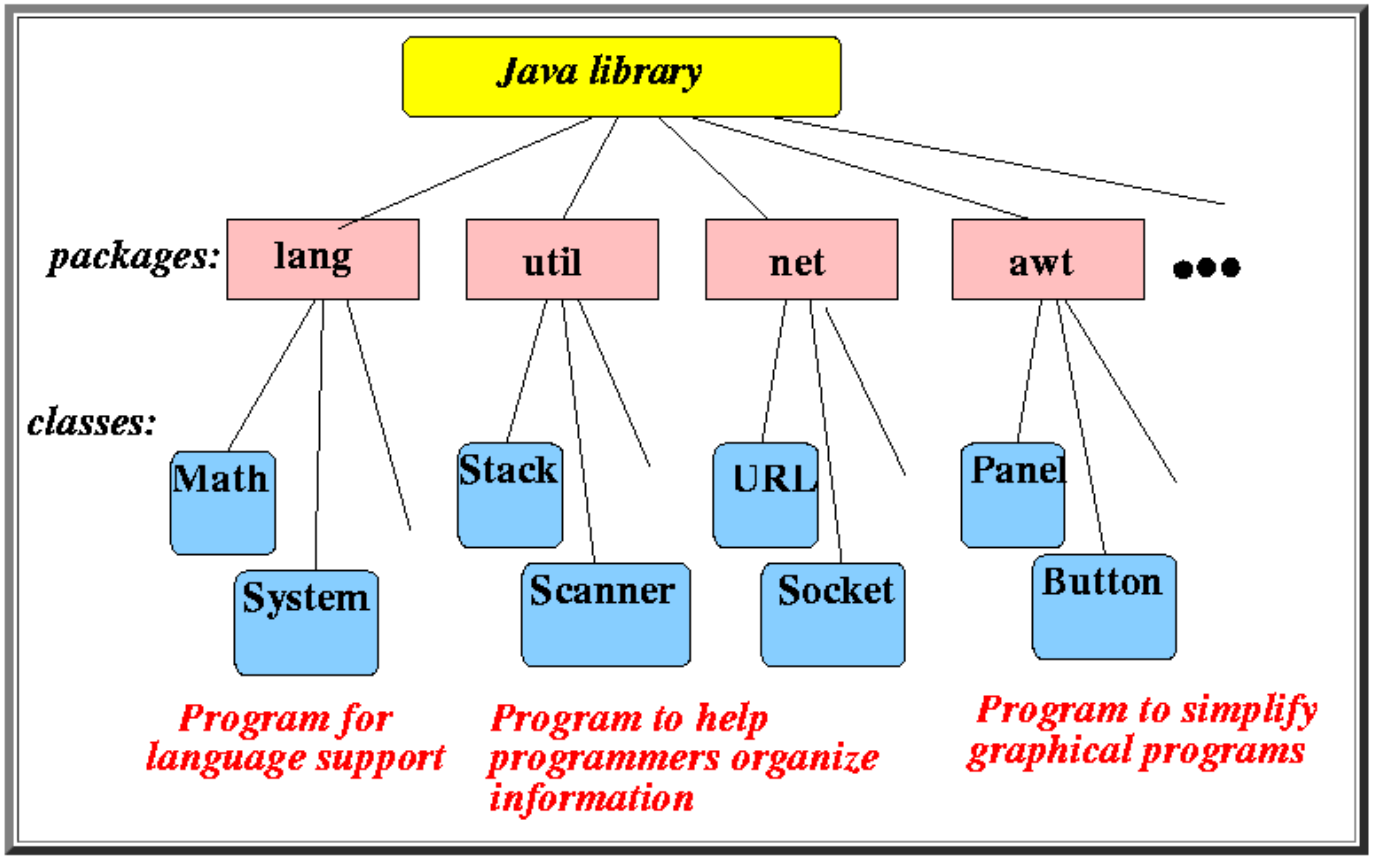
- It is used to achieve total abstraction.
- Since java does not support multiple inheritance in case of class, but by using interface it can achieve multiple inheritance .

- Interfaces are used to implement abstraction. So the question arises why use interfaces when we have abstract classes?

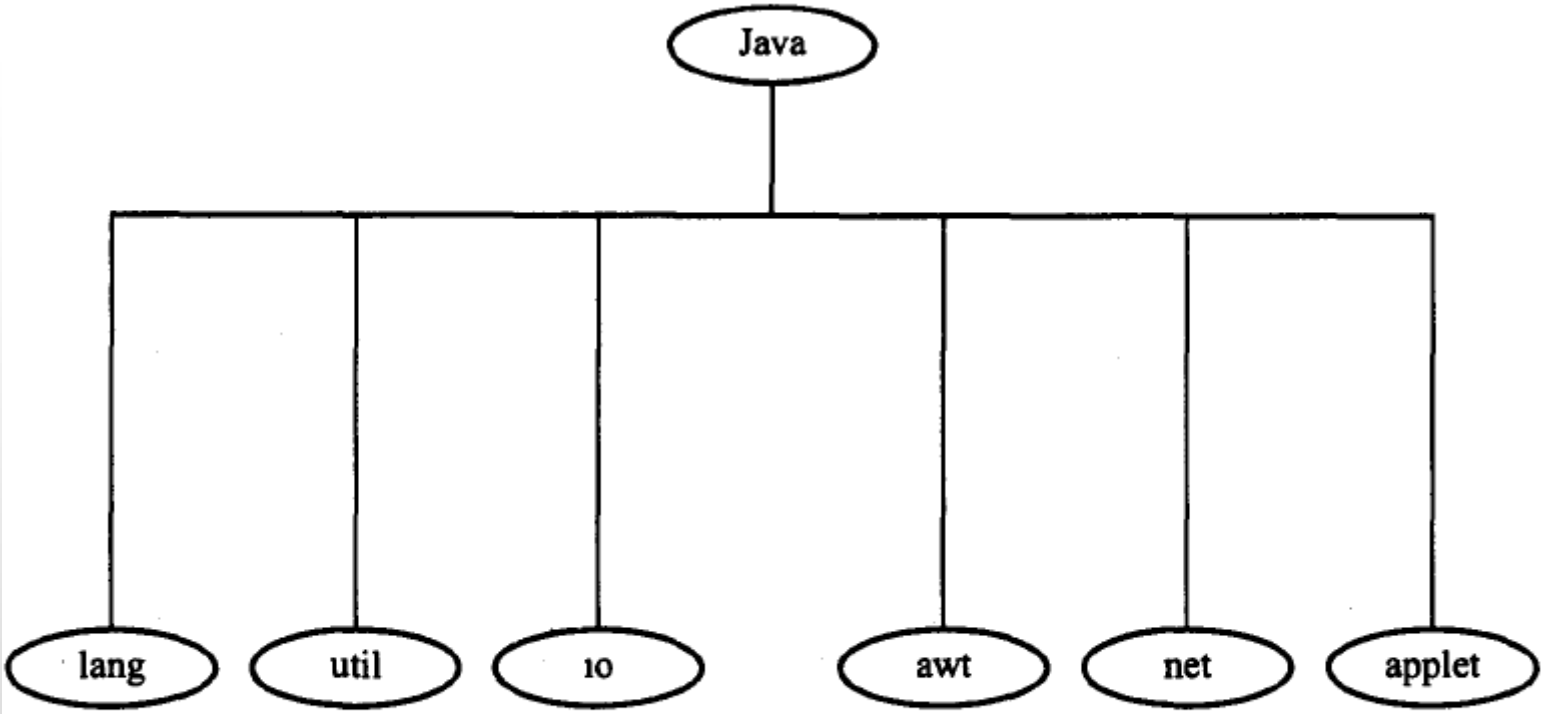
- The reason is, abstract classes may contain non-final variables, whereas variables in interface are final, public and static.

Packages

- packages are container for classes.
- used to keep the class name compartmentalized.
- package is both a naming and visibility control mechanism.



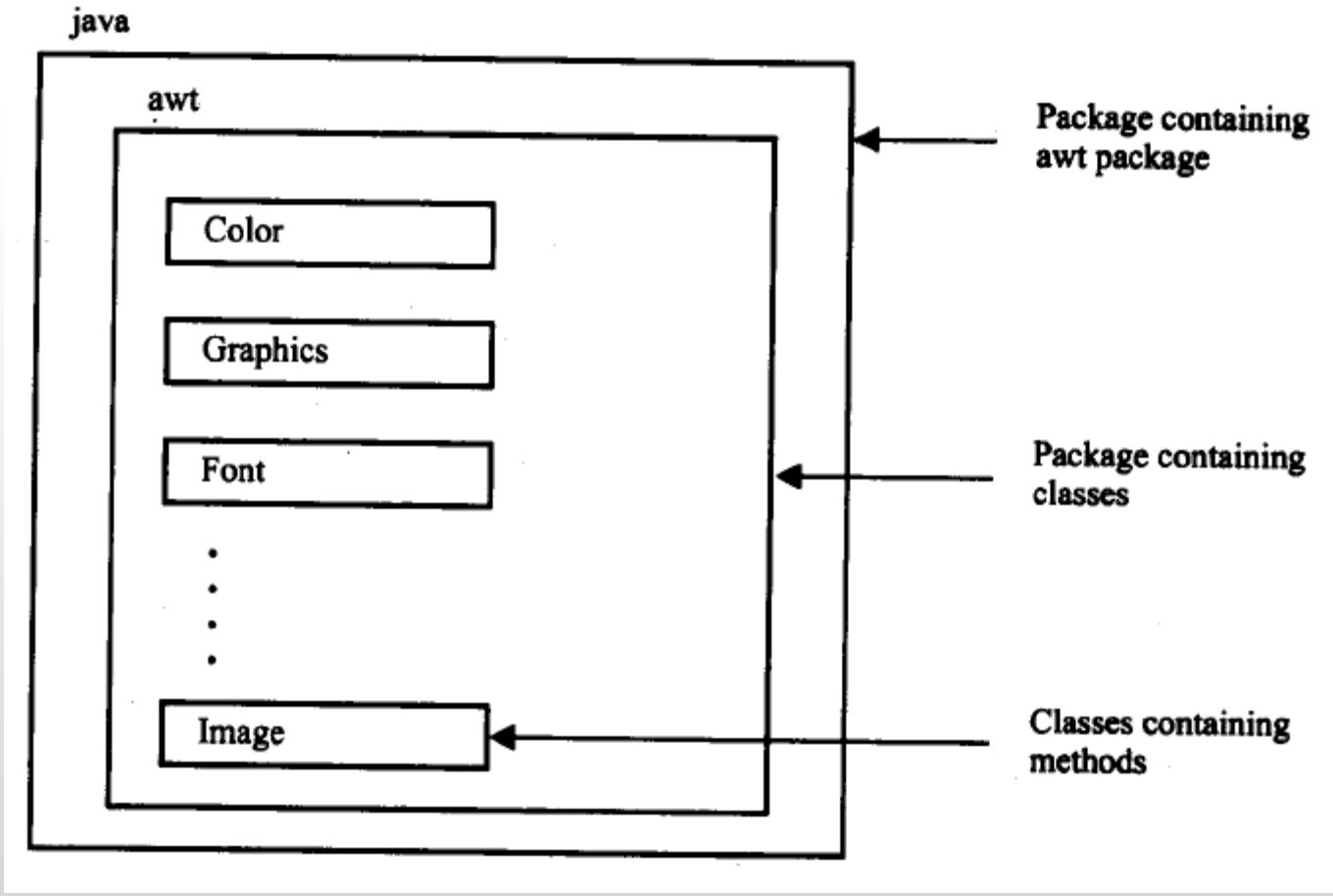
C:\Program Files (x86)\Java\jdk1.8.0_101\jre\lib\rt\java

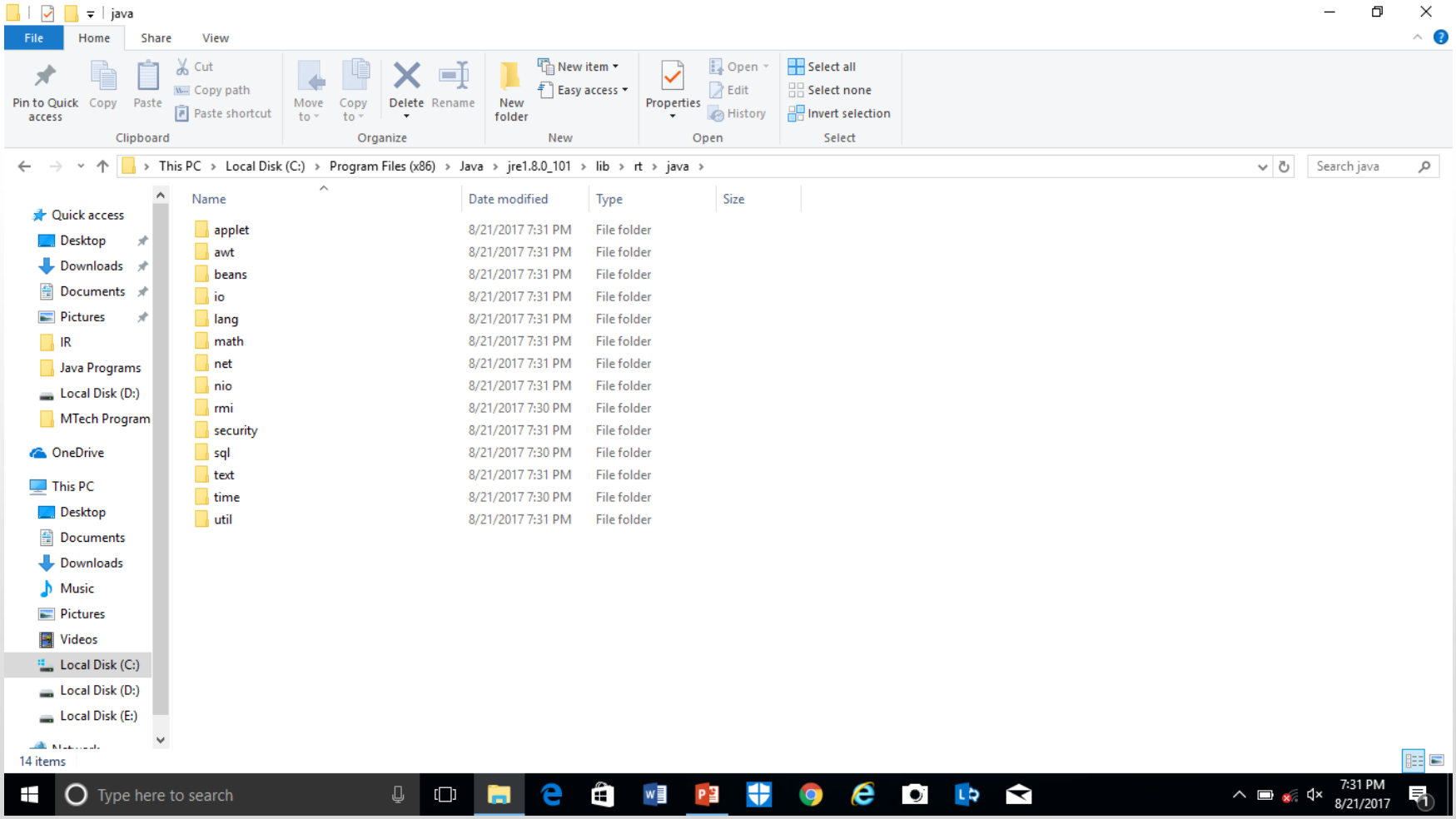


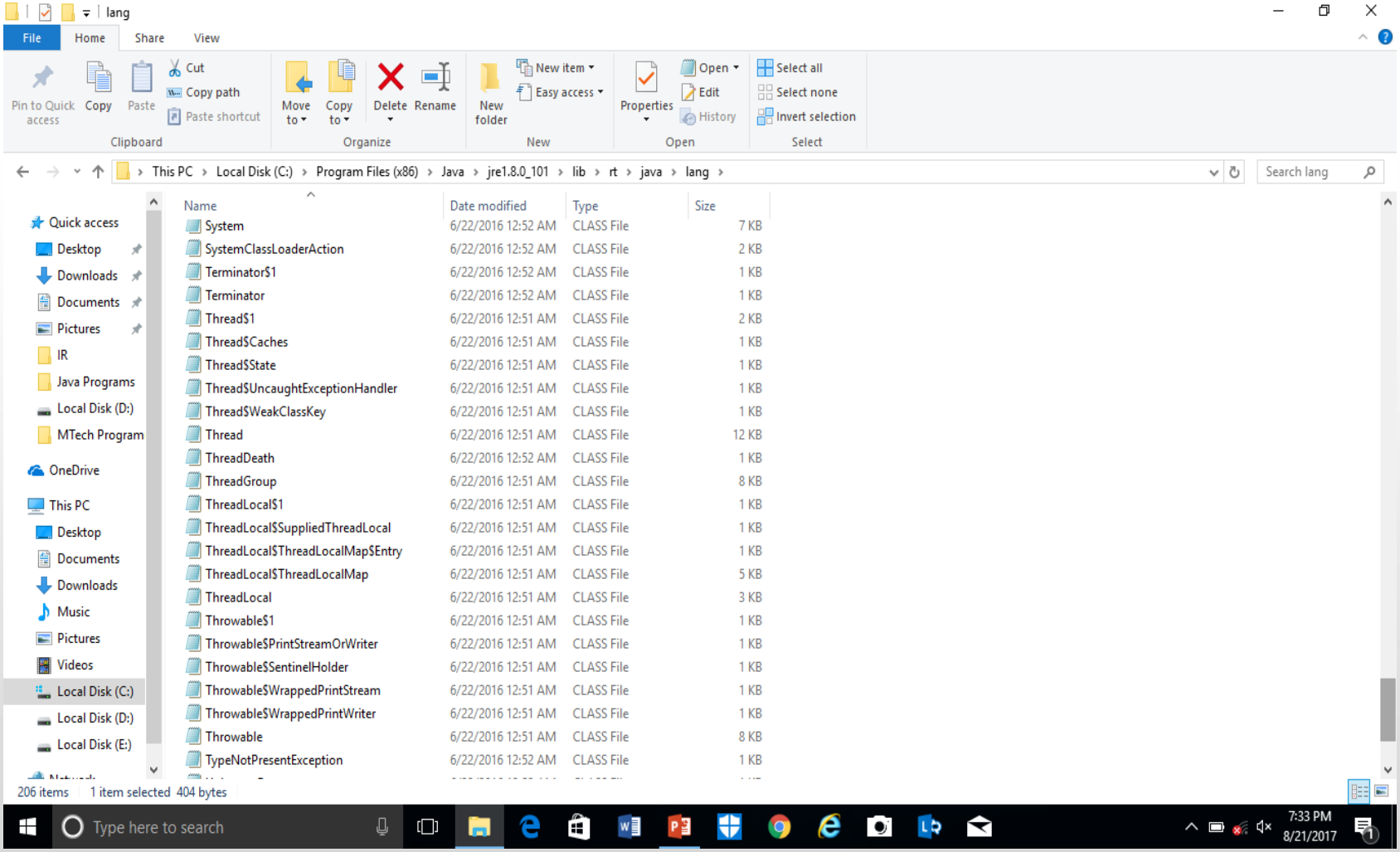
Java system packages

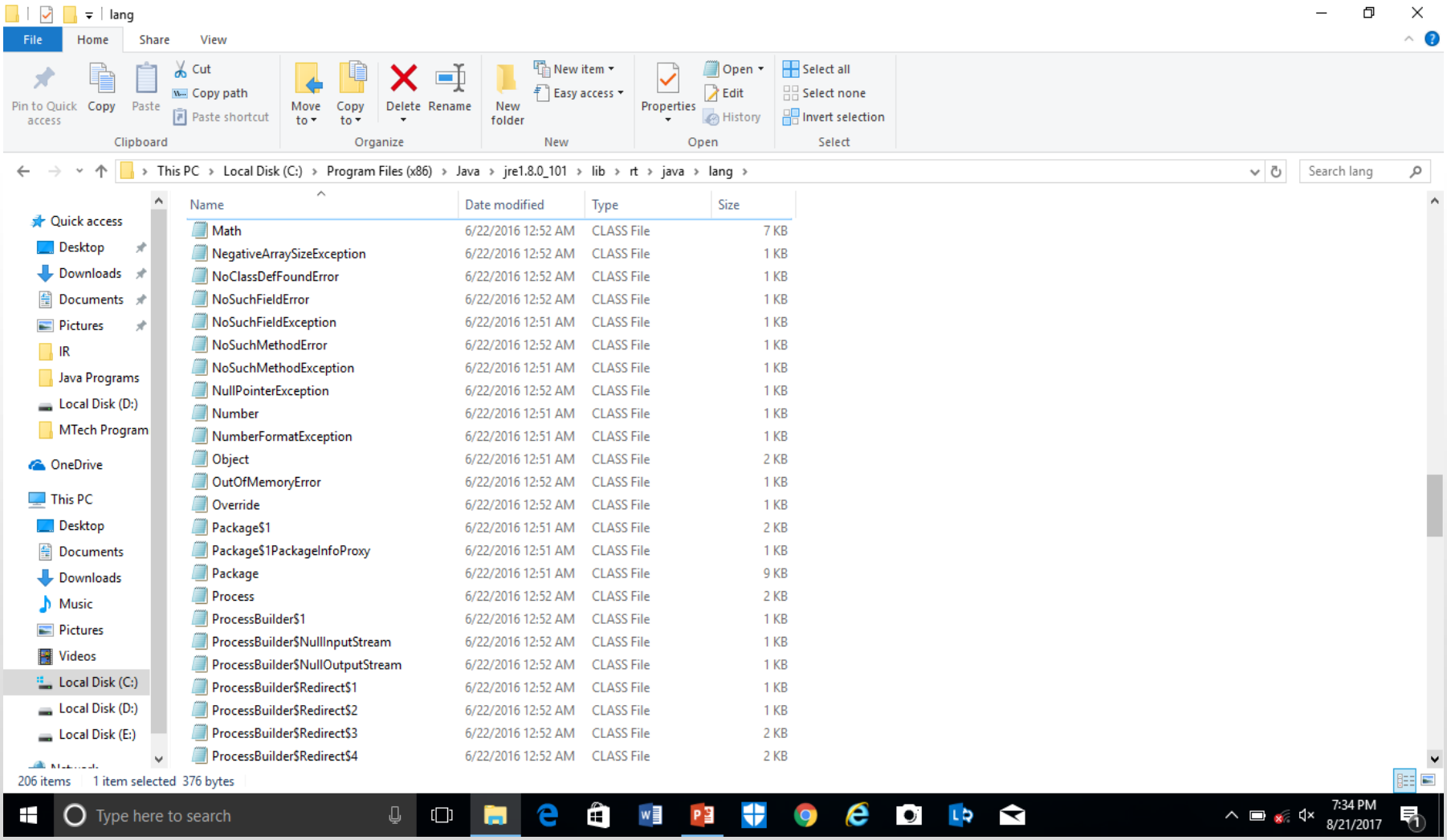
Table 11.1 Java System Packages and Their Classes

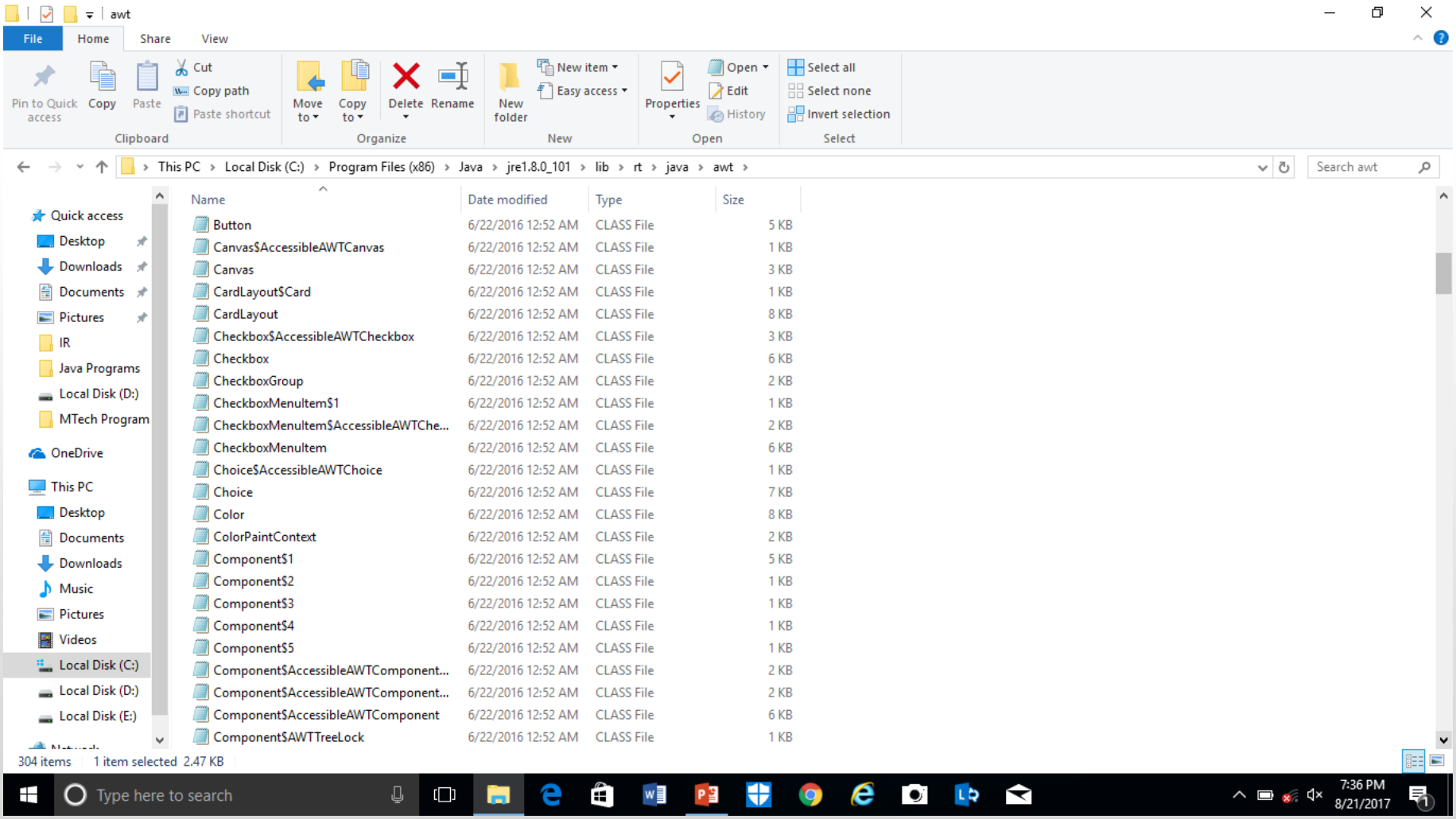
Package name	Contents
java.lang	Language support classes. These are classes that Java compiler itself uses and therefore they are automatically imported. They include classes for primitive types, strings, math functions, threads and exceptioos.
java.util	Language utility classes such as ors, hash tables. random numbers, date, etc.
java.io	Input/output support classes. They provide &lcilities for the input and output of data.
java.awt	Set of classes for implementing graphical user interface. They include classes for win'dows,buttons, lists, menus and so on.
java.net	Classes for networking. They include classes for communicating with local computers as well as with internet servers.
java.applet	Classes for creating and implementing applets.


java
awt
Color
Graphics
Font
Image
Package containing awt package
Package containing classes
Classes containing methods


java
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New item
Easy access
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
This PC › Local Disk (C:) › Program Files (x86) › Java › jre1.8.0_101 › lib › rt › java ›
Search java
Quick access
Desktop
Downloads
Documents
Pictures
IR
Java Programs
Local Disk (D:)
MTech Program
OneDrive
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
Local Disk (E:)
Name
Date modified
Type
Size
applet 8/21/2017 7:31 PM File folder
awt 8/21/2017 7:31 PM File folder
beans 8/21/2017 7:31 PM File folder
io 8/21/2017 7:31 PM File folder
lang 8/21/2017 7:31 PM File folder
math 8/21/2017 7:31 PM File folder
net 8/21/2017 7:31 PM File folder
nio 8/21/2017 7:31 PM File folder
rmi 8/21/2017 7:30 PM File folder
security 8/21/2017 7:31 PM File folder
sql 8/21/2017 7:30 PM File folder
text 8/21/2017 7:31 PM File folder
time 8/21/2017 7:30 PM File folder
util 8/21/2017 7:31 PM File folder
14 items
Type here to search
7:31 PM
8/21/2017


lang
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New item
Easy access
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
This PC › Local Disk (C:) › Program Files (x86) › Java › jre1.8.0_101 › lib › rt › java › lang ›
Search lang
Quick access
Desktop
Downloads
Documents
Pictures
IR
Java Programs
Local Disk (D:)
MTech Program
OneDrive
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
Local Disk (E:)
Name
Date modified
Type
Size
System 6/22/2016 12:52 AM CLASS File 7 KB
SystemClassLoaderAction 6/22/2016 12:52 AM CLASS File 2 KB
Terminator$1 6/22/2016 12:52 AM CLASS File 1 KB
Terminator 6/22/2016 12:52 AM CLASS File 1 KB
Thread$1 6/22/2016 12:51 AM CLASS File 2 KB
Thread$Caches 6/22/2016 12:51 AM CLASS File 1 KB
Thread$State 6/22/2016 12:51 AM CLASS File 1 KB
Thread$UncaughtExceptionHandler 6/22/2016 12:51 AM CLASS File 1 KB
Thread$WeakClassKey 6/22/2016 12:51 AM CLASS File 1 KB
Thread 6/22/2016 12:51 AM CLASS File 12 KB
ThreadDeath 6/22/2016 12:52 AM CLASS File 1 KB
ThreadGroup 6/22/2016 12:51 AM CLASS File 8 KB
ThreadLocal$1 6/22/2016 12:51 AM CLASS File 1 KB
ThreadLocal$SuppliedThreadLocal 6/22/2016 12:51 AM CLASS File 1 KB
ThreadLocal$ThreadLocalMap$Entry 6/22/2016 12:51 AM CLASS File 1 KB
ThreadLocal$ThreadLocalMap 6/22/2016 12:51 AM CLASS File 5 KB
ThreadLocal 6/22/2016 12:51 AM CLASS File 3 KB
Throwable$1 6/22/2016 12:51 AM CLASS File 1 KB
Throwable$PrintStreamOrWriter 6/22/2016 12:51 AM CLASS File 1 KB
Throwable$SentinelHolder 6/22/2016 12:51 AM CLASS File 1 KB
Throwable$WrappedPrintStream 6/22/2016 12:51 AM CLASS File 1 KB
Throwable$WrappedPrintWriter 6/22/2016 12:51 AM CLASS File 1 KB
Throwable 6/22/2016 12:51 AM CLASS File 8 KB
TypeNotPresentException 6/22/2016 12:52 AM CLASS File 1 KB
206 items
1 item selected 404 bytes
Type here to search
7:33 PM
8/21/2017


lang
File
Home
Share
View
Cut
Copy path
Paste shortcut
Pin to Quick access
Copy
Paste
Move to
Copy to
Delete
Rename
New folder
New item
Easy access
Properties
Open
Edit
History
Select all
Select none
Invert selection
Clipboard
Organize
New
Open
Select
This PC › Local Disk (C:) › Program Files (x86) › Java › jre1.8.0_101 › lib › rt › java › lang ›
Search lang
Quick access
Desktop
Downloads
Documents
Pictures
IR
Java Programs
Local Disk (D:)
MTech Program
OneDrive
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
Local Disk (E:)
Name
Date modified
Type
Size
Math 6/22/2016 12:52 AM CLASS File 7 KB
NegativeArraySizeException 6/22/2016 12:52 AM CLASS File 1 KB
NoClassDefFoundError 6/22/2016 12:52 AM CLASS File 1 KB
NoSuchFieldError 6/22/2016 12:52 AM CLASS File 1 KB
NoSuchFieldException 6/22/2016 12:51 AM CLASS File 1 KB
NoSuchMethodError 6/22/2016 12:52 AM CLASS File 1 KB
NoSuchMethodException 6/22/2016 12:51 AM CLASS File 1 KB
NullPointerException 6/22/2016 12:52 AM CLASS File 1 KB
Number 6/22/2016 12:51 AM CLASS File 1 KB
NumberFormatException 6/22/2016 12:51 AM CLASS File 1 KB
Object 6/22/2016 12:51 AM CLASS File 2 KB
OutOfMemoryError 6/22/2016 12:52 AM CLASS File 1 KB
Override 6/22/2016 12:52 AM CLASS File 1 KB
Package$1 6/22/2016 12:51 AM CLASS File 2 KB
Package$1PackageInfoProxy 6/22/2016 12:51 AM CLASS File 1 KB
Package 6/22/2016 12:51 AM CLASS File 9 KB
Process 6/22/2016 12:52 AM CLASS File 2 KB
ProcessBuilder$1 6/22/2016 12:52 AM CLASS File 1 KB
ProcessBuilder$NullInputStream 6/22/2016 12:52 AM CLASS File 1 KB
ProcessBuilder$NullOutputStream 6/22/2016 12:52 AM CLASS File 1 KB
ProcessBuilder$Redirect$1 6/22/2016 12:52 AM CLASS File 1 KB
ProcessBuilder$Redirect$2 6/22/2016 12:52 AM CLASS File 1 KB
ProcessBuilder$Redirect$3 6/22/2016 12:52 AM CLASS File 2 KB
ProcessBuilder$Redirect$4 6/22/2016 12:52 AM CLASS File 2 KB
206 items
1 item selected 376 bytes
Type here to search
7:34 PM
8/21/2017


awt
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New item
Easy access
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
This PC › Local Disk (C:) › Program Files (x86) › Java › jre1.8.0_101 › lib › rt › java › awt ›
Search awt
Quick access
Desktop
Downloads
Documents
Pictures
IR
Java Programs
Local Disk (D:)
MTech Program
OneDrive
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
Local Disk (E:)
Name
Date modified
Type
Size
Button 6/22/2016 12:52 AM CLASS File 5 KB
Canvas$AccessibleAWTCanvas 6/22/2016 12:52 AM CLASS File 1 KB
Canvas 6/22/2016 12:52 AM CLASS File 3 KB
CardLayout$Card 6/22/2016 12:52 AM CLASS File 1 KB
CardLayout 6/22/2016 12:52 AM CLASS File 8 KB
Checkbox$AccessibleAWTCheckbox 6/22/2016 12:52 AM CLASS File 3 KB
Checkbox 6/22/2016 12:52 AM CLASS File 6 KB
CheckboxGroup 6/22/2016 12:52 AM CLASS File 2 KB
CheckboxMenuItem$1 6/22/2016 12:52 AM CLASS File 1 KB
CheckboxMenuItem$AccessibleAWTChe... 6/22/2016 12:52 AM CLASS File 2 KB
CheckboxMenuItem 6/22/2016 12:52 AM CLASS File 6 KB
Choice$AccessibleAWTChoice 6/22/2016 12:52 AM CLASS File 1 KB
Choice 6/22/2016 12:52 AM CLASS File 7 KB
Color 6/22/2016 12:52 AM CLASS File 8 KB
ColorPaintContext 6/22/2016 12:52 AM CLASS File 2 KB
Component$1 6/22/2016 12:52 AM CLASS File 5 KB
Component$2 6/22/2016 12:52 AM CLASS File 1 KB
Component$3 6/22/2016 12:52 AM CLASS File 1 KB
Component$4 6/22/2016 12:52 AM CLASS File 1 KB
Component$5 6/22/2016 12:52 AM CLASS File 1 KB
Component$AccessibleAWTComponent... 6/22/2016 12:52 AM CLASS File 2 KB
Component$AccessibleAWTComponent... 6/22/2016 12:52 AM CLASS File 2 KB
Component$AccessibleAWTComponent 6/22/2016 12:52 AM CLASS File 6 KB
Component$AWTTreeLock 6/22/2016 12:52 AM CLASS File 1 KB
304 items
1 item selected 2.47 KB
Type here to search
7:36 PM
8/21/2017

Defining a package

- To create a package include a package command as the first statement.
- Any classes declared within that file will belong to the specified package.
- If package statement is not included, then class names are put into the default package, which has no name.

General form :

package pkg;

Ex: package MyPackage;

We can create a hierarchy of package.

General form

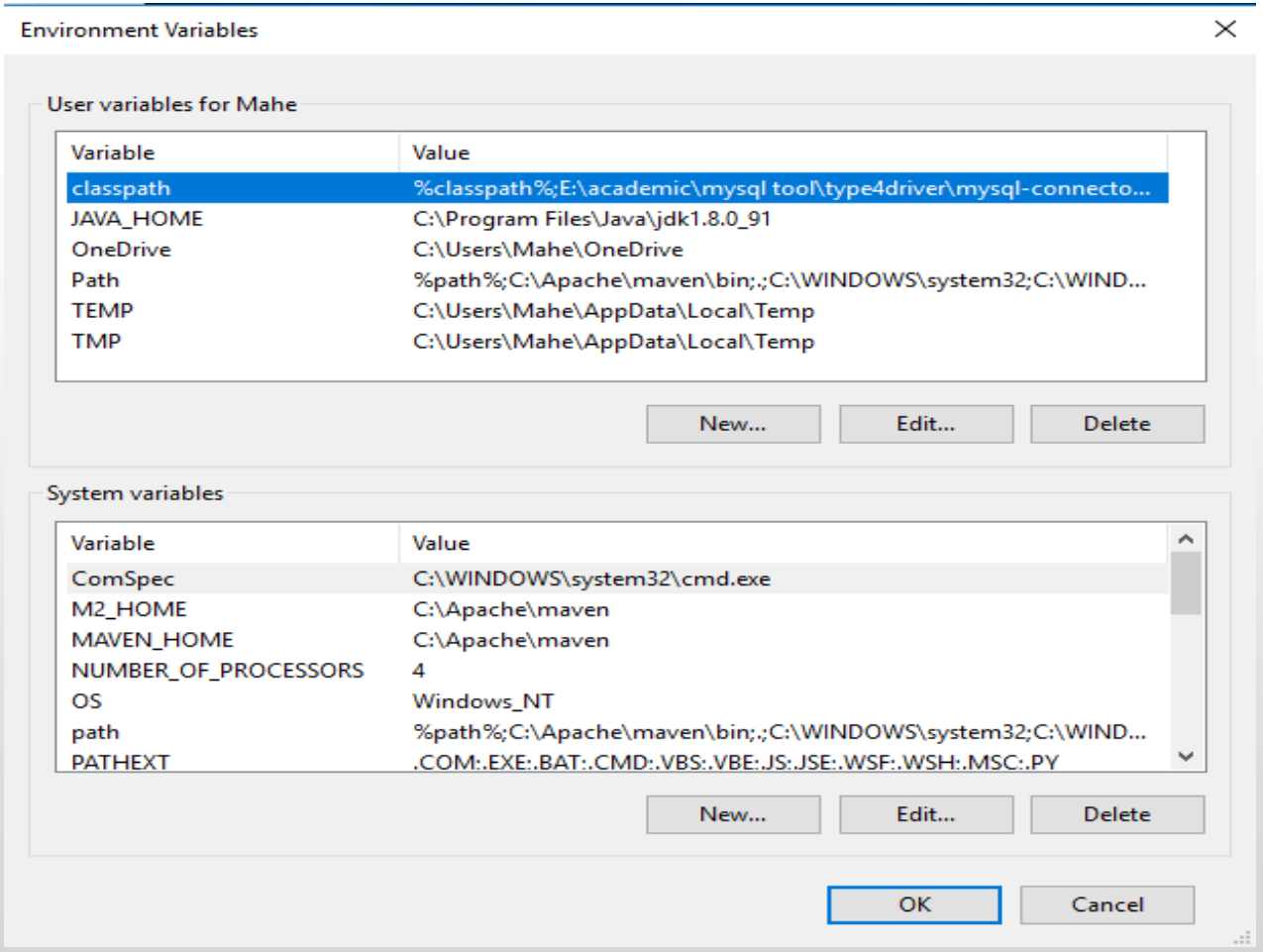
package pkg1[.pkg2[.pkg3]];

Ex: package java.awt.image.

Finding packages and CLASSPATH

- First, by default, the Java run-time system uses the current working directory as its starting point.
- if our package is in the current directory, or a subdirectory of the current directory, it will be found.
- Second, we can specify a directory path or paths by setting the CLASSPATH environmental variable.
- We can use –classpath option with java and javac to specify the path to our class.

Finding packages and CLASSPATH



Example

Consider the following package specification.

```
package MyPack;
```

In order for a program to find MyPack, one of the 3 things must be true.

- Either the program is executed from a directory immediately above **MyPack.**
- CLASSPATH must be set to include the path to MyPack.
- The –classpath option must specify the path to MyPack when the program is run via java.

When second two options are used, the classpath must not include MyPack itself.

It must specify the path to MyPack.

For ex: in windows if the path is to MyPack is

C:\MyPrograms\java\MyPack

Then the classpath to MyPack is

C:\MyPrograms\java

A Short Package Example

```
package MyPack;
class prg
{
    public static void main(String args[])
    {
        System.out.println("first  program to illustrate package");
    }
}
```

Name the above file as prg.java and put it in a directory called MyPack.

Next compile the file, make sure that resulting class file is also in MyPack.

Run the program using

```
java MyPack.prg
```

Access Protection

	Private	No modifier	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package subclass	No	Yes	Yes	Yes
Same package non-subclass	No	Yes	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Table 9-1. *Class Member Access*

Example has two packages p1 , p2 and five classes.

- first package defines three classes: Protection, Derived and SamePackage.
- The first class defines four int variables in each of the legal protection modes.
- The variable n is declared with the default protection, n_pri is private, n_pro is protected, and n_pub is public.

- The second class, Derived, is a subclass of Protection in the same package p1.
  - This grants Derived access to every variable in Protection except for n_pri.

- The third class, SamePackage, is not a subclass of Protection, but is in the same package and also has access to all but n_pri.

```
package  p1
class protection
{


}
class derived extends protection
{


}
class samepackage
{


}
```

```
package  p2
class  derived2 extends p1.protection
{


}

class otherpackage
{


}
```

Protection.java

```
package p1;
public class Protection
 {
         int n = 1;
         private int n_pri = 2;
         protected int n_pro = 3;
          public int n_pub = 4;
          public Protection()
        {
              System.out.println("n = " + n);
              System.out.println("n_pri = " + n_pri);
              System.out.println("n_pro = " + n_pro);
              System.out.println("n_pub = " + n_pub);
     }
}
```

Derived.java

```
package p1;

class Derived extends Protection
{
    Derived()
    {
            System.out.println("n = " + n);
            // System.out.println("n_pri = " + n_pri);
            System.out.println("n_pro = " + n_pro);
            System.out.println("n_pub = " + n_pub);
    }
}
```

SamePackage.java

```
package p1;
class SamePackage
{
        SamePackage()
        {
            Protection p = new Protection();
            System.out.println("n = " + p.n);
            // System.out.println("n_pri = " + p.n_pri);
            System.out.println("n_pro = " + p.n_pro);
            System.out.println("n_pub = " + p.n_pub);
   }
}
```

Protection2.java

```
package p2;
class derived2 extends p1.Protection
{
        Protection2()
        {
                // System.out.println("n = " + n);
                // System.out.println("n_pri = " + n_pri);
                 System.out.println("n_pro = " + n_pro);
                 System.out.println("n_pub = " + n_pub);
         }
}
```

OtherPackage.java

```
package p2;
class OtherPackage
{
   OtherPackage()
   {
            p1.Protection p = new p1.Protection();

            // System.out.println("n = " + p.n);
            // System.out.println("n_pri = " + p.n_pri);
            // System.out.println("n_pro = " + p.n_pro);
            System.out.println("n_pub = " + p.n_pub);
   }
}
```

Importing Packages

import statement to bring certain classes, or entire packages, into visibility.

Once imported, a class can be referred to directly, using only its name.

This is the general form of the import statement:

import *pkg1[.pkg2].(classname|*);*

```
import  java.util.Date;
import  java.io.*;
import  java.awt.*;
```

import statements occur immediately following the package statement (if it exists) and before any class definitions.

All of the standard Java classes included with Java are stored in a package called java.

The basic language functions are stored in a package inside of the java package called java.lang.

We have to import every package or class that we want to use, but since Java is useless without much of the functionality in java.lang, it is implicitly imported by the compiler for all programs.

This is equivalent to the following line being at the top of all of our programs:

```
import java.lang.*;
```

import statement is optional.

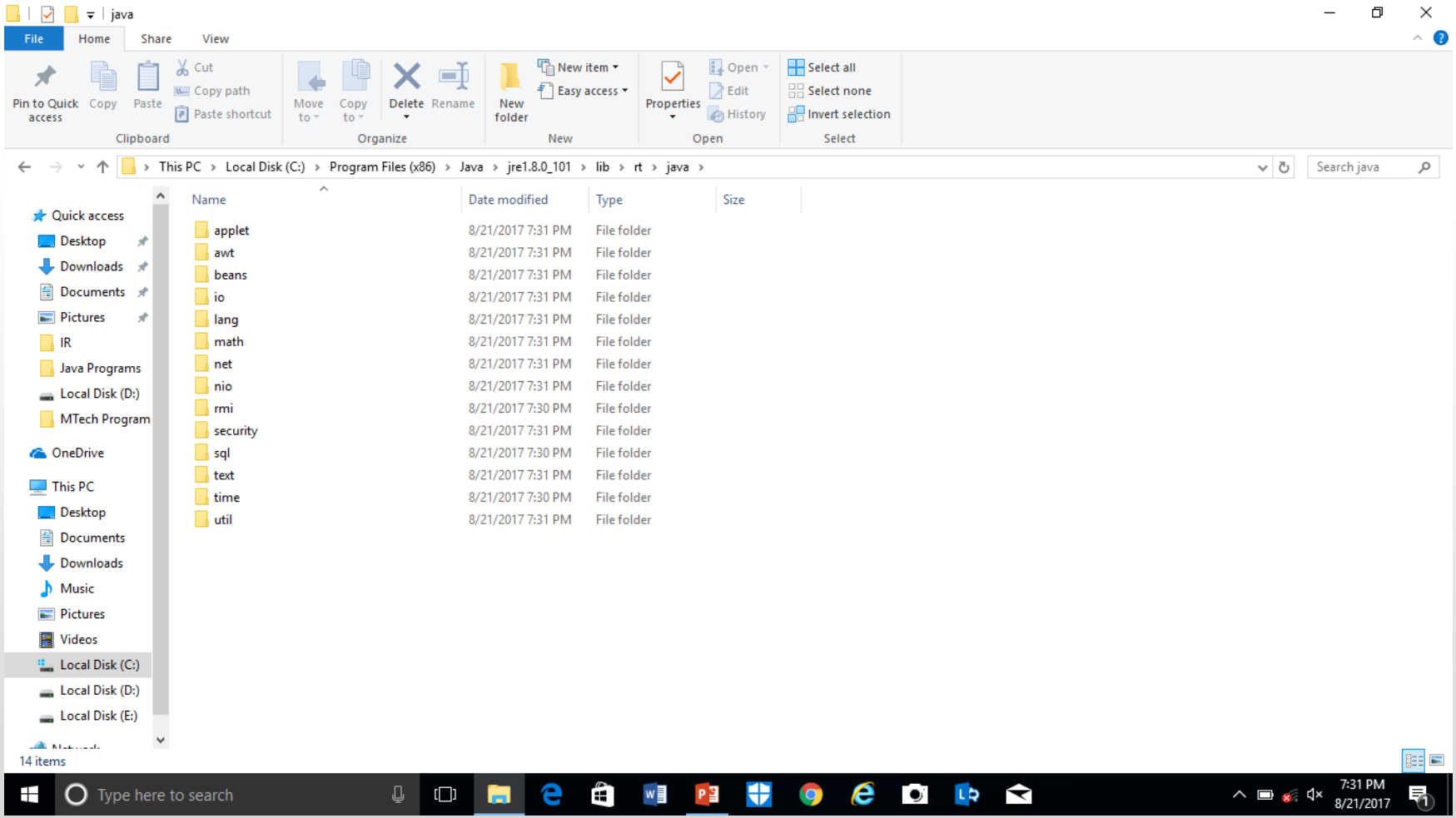
```
import java.util.*;
class MyDate extends Date
 {

}
```

The same example without the **import statement looks like this:**

```
class MyDate extends java.util.Date
{

}
```


java
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New item
Easy access
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
This PC › Local Disk (C:) › Program Files (x86) › Java › jre1.8.0_101 › lib › rt › java ›
Search java
Quick access
Desktop
Downloads
Documents
Pictures
IR
Java Programs
Local Disk (D:)
MTech Program
OneDrive
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
Local Disk (E:)
Name
Date modified
Type
Size
applet 8/21/2017 7:31 PM File folder
awt 8/21/2017 7:31 PM File folder
beans 8/21/2017 7:31 PM File folder
io 8/21/2017 7:31 PM File folder
lang 8/21/2017 7:31 PM File folder
math 8/21/2017 7:31 PM File folder
net 8/21/2017 7:31 PM File folder
nio 8/21/2017 7:31 PM File folder
rmi 8/21/2017 7:30 PM File folder
security 8/21/2017 7:31 PM File folder
sql 8/21/2017 7:30 PM File folder
text 8/21/2017 7:31 PM File folder
time 8/21/2017 7:30 PM File folder
util 8/21/2017 7:31 PM File folder
14 items
Type here to search
7:31 PM
8/21/2017

```
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.Vm;


  class prg
  {
          public static void main(String args[])
         {
                        Host  h = new Host();
                       Vm  v = new Vm();
         }
  }
```