Java Programming

TEK RAJ CHHETRI

Course Objective

- Explain the Java programming environment
- Describe the concepts of programming elements using Java and object-oriented
- programming concepts
- Apply the exception handling and input/output in Java programming
- Apply the event handling, GUI programming using swing, and Java database connectivity

Unit 3: Objects and Classes

- Introduction to Object-Oriented Programming
- Using pre-defined Classes
- Defining your own class
- Static fields and Methods
- Method Parameters
- Object Construction
- Packages

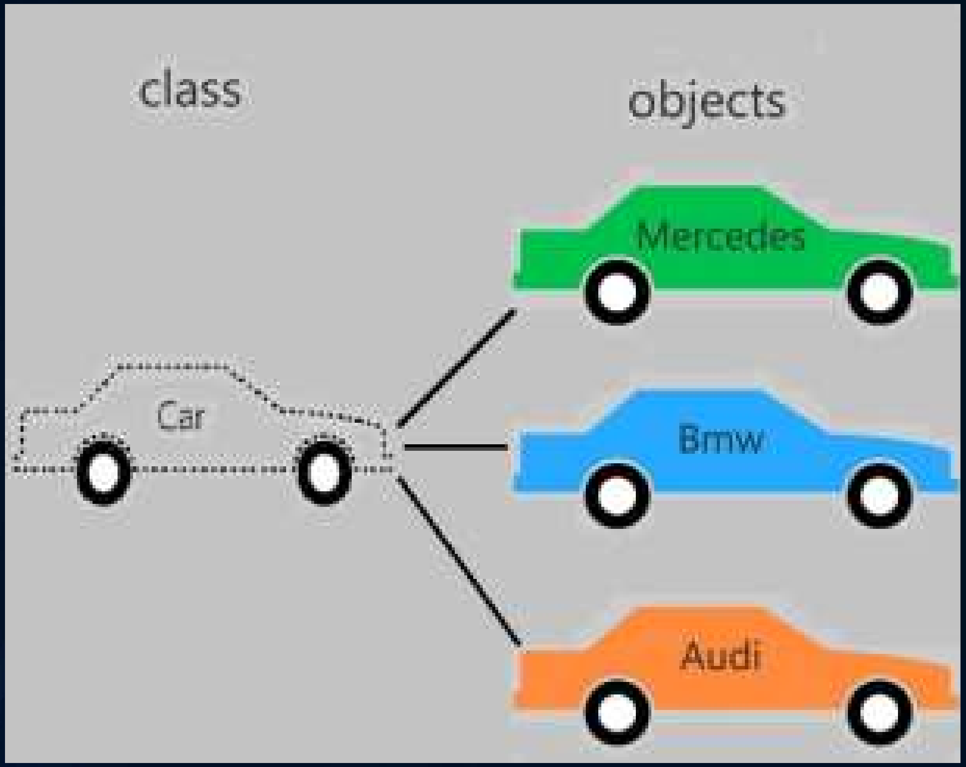
Learning Outcome (Unit 3)

- Develop understanding about object oriented programming
- Ability to use pre-defined class and create new user defined class
- Understanding about static fields, methods and ability to create and use it in programming
- Create objects and packages and use it to solve problem

Object Oriented Programming?

- Used to describe programming approach based on objects and class

- Why OOP?
- Modularization: application can be decomposed into modules
- Software re-use: application can be composed from existing and new module


class
objects
Car
Mercedes
Bmw
Audi

Using Pre-defined Class

```java
import java.io.*;
class Predefined{
        public static void main(String[] args) {
                PrintWriter printWriter = new PrintWriter(System.out, true);
                printWriter.println("Hello there!");
                printWriter.println(124982);
        }
}
```

```java
public class Integers
{
        public static void main(String[] args)
        {
                Integer i = new Integer(100);
                System.out.println(i.compareTo(70));
                System.out.println(i.compareTo(100));
                System.out.println(i.compareTo(101));

        }
}
```

- https://docs.oracle.com/javase/7/docs/api/allclasses-noframe.html

Defining your Own Class

- Class: a blue print / template describing behavior / state that object of its type support.

- Declared using the keyword **class**

- Data or variables within class are called instance variables

- Collectively methods and variables are called members of class

- **Syntax:**

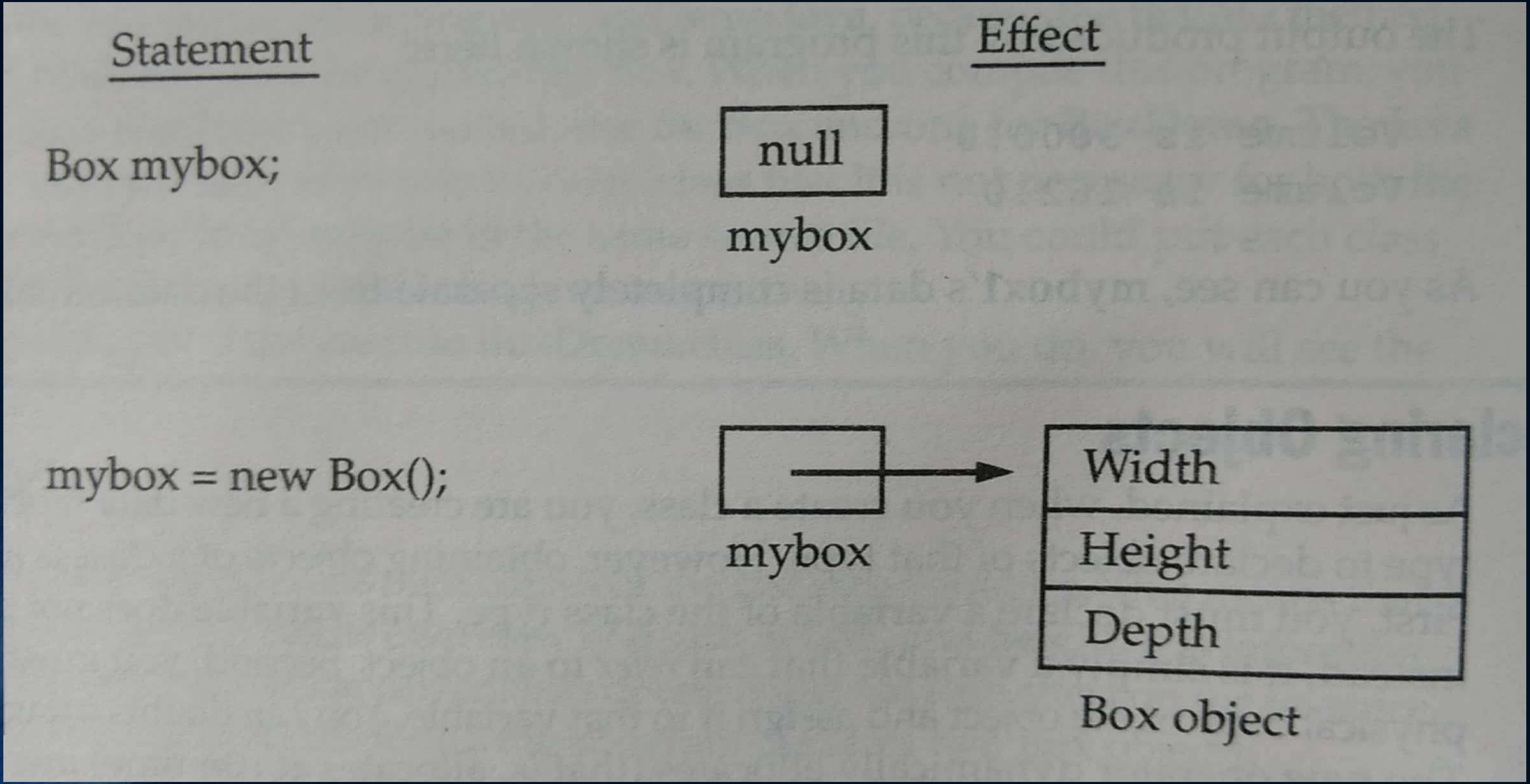
```
class classname{
        type instancevariable1;
        type instancevariable2;
        type instancevariablen;

        type methodName1(paramater-list){
                //method body
        }
        type methodNamen(paramater-list){
                //method body
        }
}
```

- **<u>Example</u>:**

```
class UserDefinedClass{
        public static void main(String [] args){
                System.out.println(“User-Defined");
        }
}
```

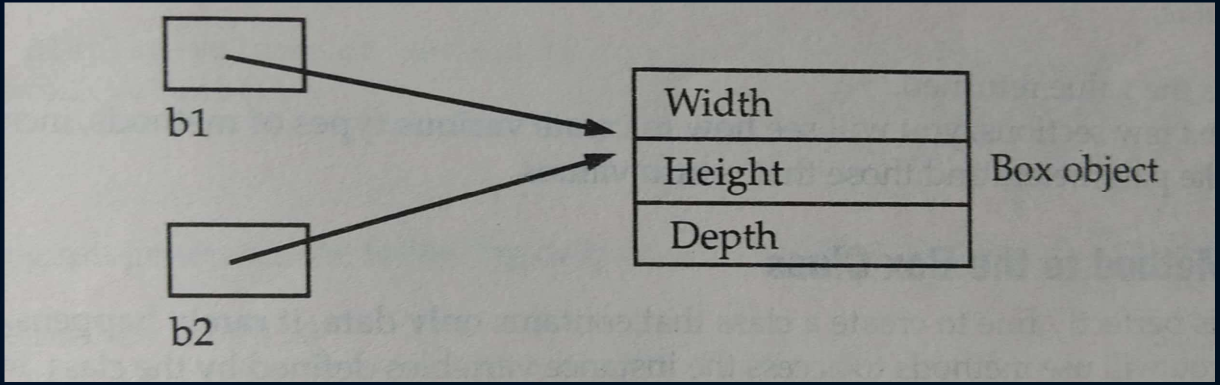
Object Construction

- **Declaring Object:**
  - Syntax: Classname objectname;
  - Eg: Box mybox;
- **Object Memory Allocation:**
  - The new keyword is used to allocate the memory to object dynamically.
  - Syntax: new Classname
  - Eg: new Box()
- Combining both, we create an object
  - Box mybox = new Box()



```
Box mybox;

mybox = new Box();
```

- **Object reference variables:**
  - Box b1 = new Box()
  - Box b2 = b1
  - Any changes through object b2 will affect object b1



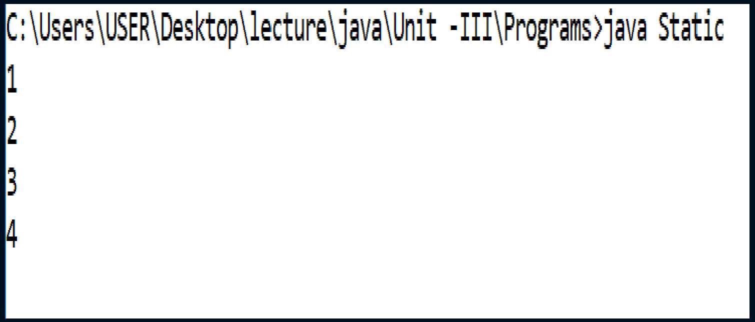
- b1 and b2 reference same object, not linked in any other way.
  - Box b1 = new Box()
  - Box b2 = b1
  - b1 = null
  - b1 is set to null, b2 still points to original object

Static fields and Methods

- **Static variables and methods:**
  - Declared using keyword static
  - Can be accessed without creating object, just by using class name if present in different class
  - Variable with keyword static are called static variables and methods are called static methods.
  - Static methods are the first to be executed.

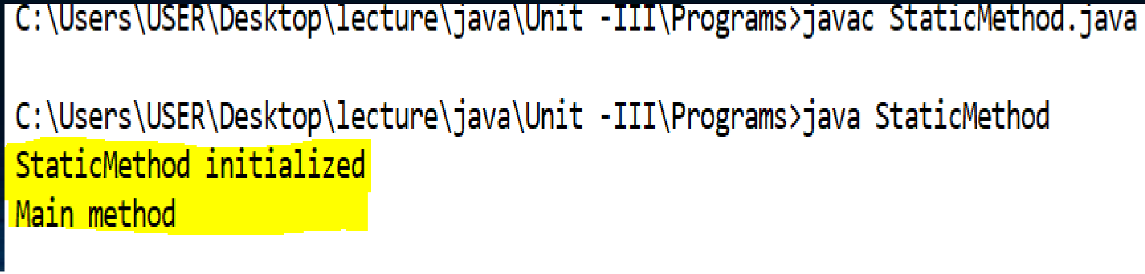
  - Syntax Static Variable:
    - Static data-type variable_name
    - Eg: static int counter
  - Syntax Static Method:
    - static return-type method-name{
      - //code
    - }
    - Eg: public static void main(String args[]){ }

```java
class Static{
    static int Number = 0;
    Static(){
            System.out.println(++Number);
    }
    public static void main(String[] args) {
            Static s1 = new Static();
            Static s2 = new Static();
            Static s3 = new Static();
            Static s4 = new Static();


    }
}
```



```
C:\Users\USER\Desktop\lecture\java\Unit -III\Programs>java Static
1
2
3
4
```

```java
class StaticMethod{
    static{
                System.out.println("StaticMethod initialized");
    }
    public static void main(String[] args) {
                System.out.println("Main method");


    }
}
```



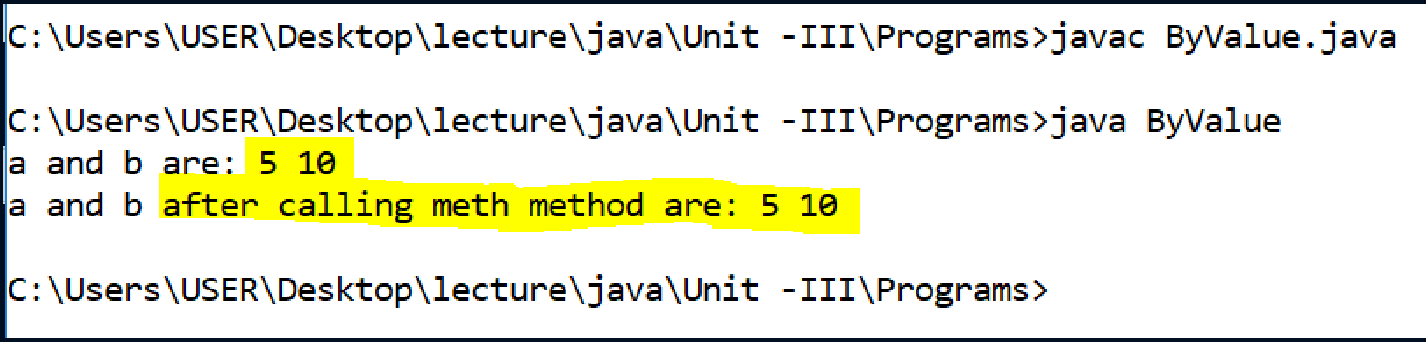
```
C:\Users\USER\Desktop\lecture\java\Unit -III\Programs>javac StaticMethod.java

C:\Users\USER\Desktop\lecture\java\Unit -III\Programs>java StaticMethod
StaticMethod initialized
Main method
```

Method Parameters

- **Method:**
- Syntax:
  - return type method-name(parameter1, parameter2,…, paramatern){
    - //code to execute
  - }
- While passing parameter, its type must be defined.
- Parameter can be passed by value and by reference.
- For reference method we pass object as parameter while for object we just pass value.

- **By Value:**

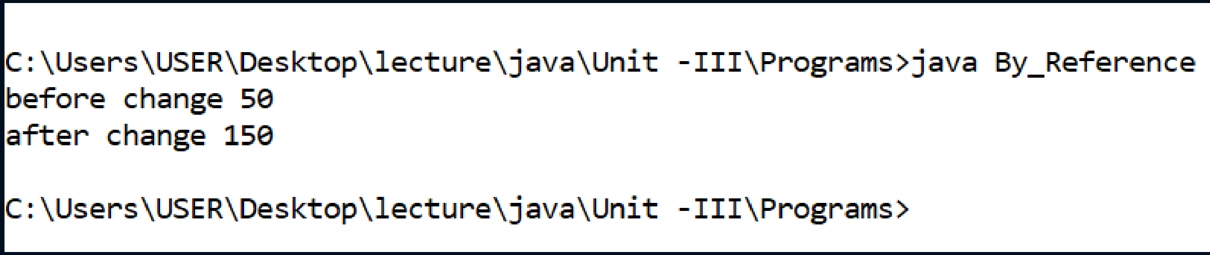
```
class Test{
        void meth(int i, int j){
                i *= 2;
                j /= 2;
        }
}
class ByValue{
        public static void main(String[] args) {
                int a = 5, b = 10;
                Test test = new Test();
                System.out.println("a and b are: "+a+" "+b);
                test.meth(a, b);
                System.out.println("a and b after calling meth method are: "+a+" "+b);
        }
}
```



```
C:\Users\USER\Desktop\lecture\java\Unit -III\Programs>javac ByValue.java

C:\Users\USER\Desktop\lecture\java\Unit -III\Programs>java ByValue
a and b are: 5 10
a and b after calling meth method are: 5 10

C:\Users\USER\Desktop\lecture\java\Unit -III\Programs>
```

- **By_Reference:**

```
class By_Reference{
 int data=50;
 void change(By_Reference op){
 op.data=op.data+100;//changes will be in the local variable only
 }
 public static void main(String args[]){
   By_Reference op=new By_Reference();
   System.out.println("before change "+op.data);
   op.change(op);//passing object
   System.out.println("after change "+op.data);
 }
}
```



```
C:\Users\USER\Desktop\lecture\java\Unit -III\Programs>java By_Reference
before change 50
after change 150

C:\Users\USER\Desktop\lecture\java\Unit -III\Programs>
```

Packages

- **Packages:**
- Packages are the containers for classes that are used to keep the class name space compartmentalized.

- Defining Package:
  - The package command is used to define the package placing it on the top.
  - package hello; //here hello is name of the package
  - Java uses file system directories to store packages. Meaning, package hello must be placed inside folder named hello.
  - Hierarchy of packages can be created. For this we use period to separate package name.
  - package pag1[.pkg2[.pkg3]];
  - Eg: package java.awt.image;

- **Executing package hello:**
  - First java runtime environment uses the current directory and looks into subdirectory of current directory.
  - Second is to specify directory path/s by setting CLASSPATH environment variable.
  - Third is to use –classpath option with java and javac command.

  - Eg:
  - package hello;

  - If package hello is in C:\Programs\Lecture\hello then classpath to package hello is C:\Programs\Lecture

- **Example:**

```
package bedict;
class Name{
      String name;
      Name(String n){
                        name = n;
      }
      void show(){
                        System.out.println(name);
      }
}
class Packages{
      public static void main(String[] args) {
                        Name [] names = new Name[3];
                        names[0] = new Name("OS");
                        names[1] = new Name("Java");
                        names[2] = new Name("Python");

                        for( int i = 0; i < names.length; i++){
                                    names[i].show();
                        }
      }
}
```

- Consider the location of above program is C:\Programs\Lecture\bedict then classpath to package hello is C:\Programs\Lecture
- Compile the program as usual.
- To run however, we need to be one step above where the current program is.

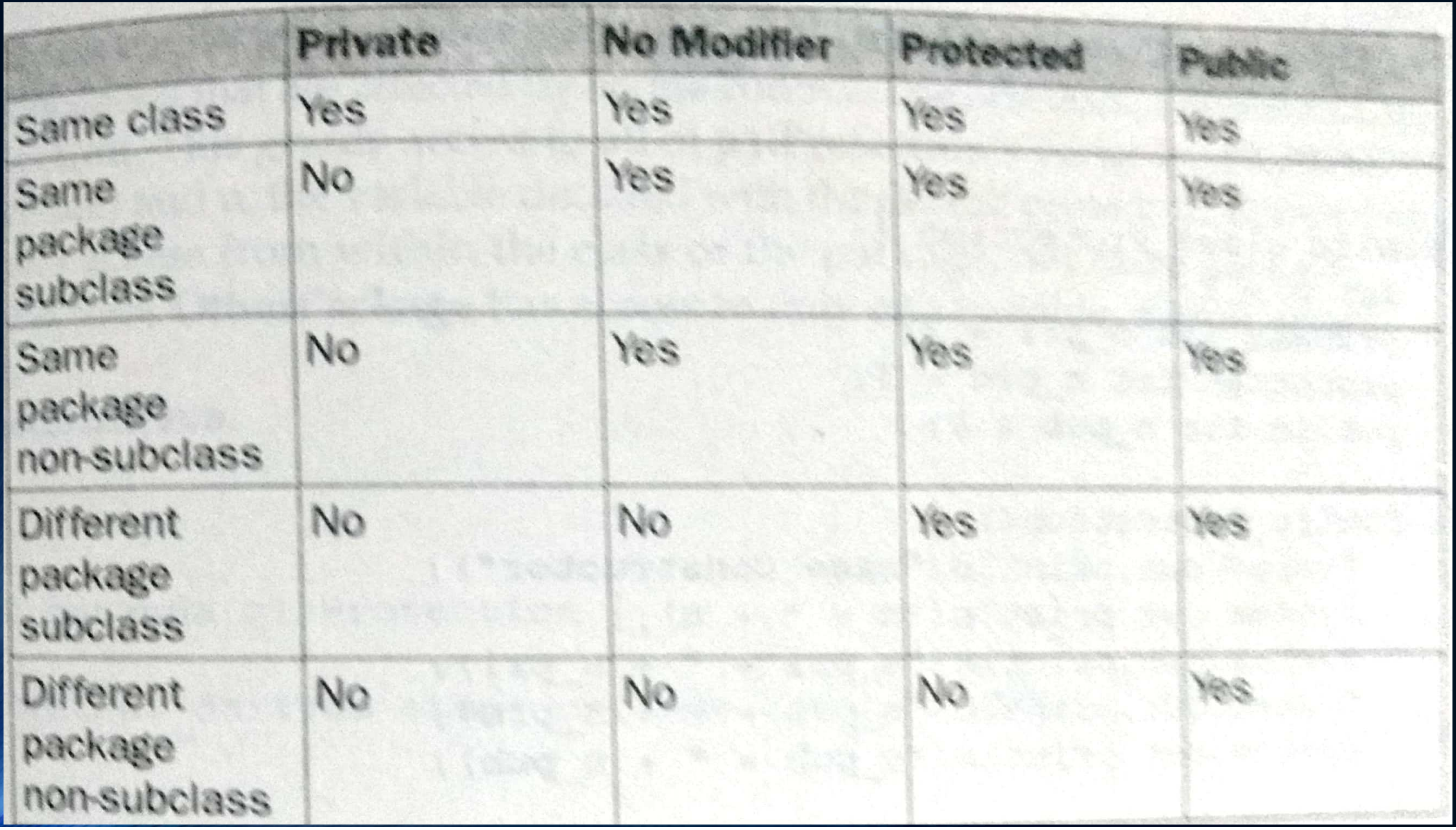
  Being in location C:\Programs\Lecture\bedict and running won't work.

  java Packages won't work

  we need to be in C:\Programs\Lecture and run the program to execute.

  java bedict.Packages → correct way of executing program

Access Modifier



	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

- **Importing packages:**
- import pkg1[.pkg2].(classname|*)
- Eg:
- import java.io.*;
- import java.util.Date;

Suggested Readings

- The respective topics in The complete Reference Java 7 (or any higher edition) by Hebert Schildt
- Oracle official java documentation

References

- The complete Reference Java 7 by Hebert Schildt
- Java 8 in Action by Dreamtech press.
- Mit Opencourseware
- http://ee402.eeng.dcu.ie/
- https://www.javatpoint.com/
- https://docs.oracle.com/javase/tutorial/?sess=16e492aba13789410194of7f88d9f51f
- https://images.google.com for Images