University of Bahri
Collage of Computer Sciences& Mathematic
2nd Year –Object Oriented Paradigms

Lab No (2)Classand object

Classes

- One Java class defined in each .java file
- File name must match the name of the class
  - Otherwise there will be compilation errors.
  - Class names start with an upper case letter.
- Compiler will generate a .class file with same name
  - Contains the *bytecode*.
- Classes defined using the **class** keyword.
- Class Definitions

```
public class ClassName
{
        // data member (variable) declarations
        // constructor definitions
        public ClassName(param list)
        {   ……..}
// method definitions
}
```

- **Access Specifiers (public/protected/ private)**
- **public**
  - Can be directly accessed from any other class.
  - Used for classes, constructors, and many methods.
- **protected**
  - Can be directly accessed from any other class in the same package (folder)
  - Used for inheritance related classes usually.
- **private**
  - Can only be directly accessed from within the same class.
  - Used to protect data.

Methods

- Define some behaviour of a class
- Method declarations have four basic sections, and a method body:
  - Visibility modifier (who can call the method)
  - Return type (what does it return)

- Method name
- Parameter list (what parameters does it accept)

```
public void printInf()
{
  //code goes here
}
private int addNumbers(int x, int y)
{
  //code goes here
}
```

- Calling a method
  - Methodname(Parameter list);

```
addNumbers(7, 12);
printInf();
```

Constructors

- new causes a constructor to be invoked
  - Constructor is a special method, used to initialise an object.
  - Class often specifies several constructors (for flexibility).
  - new operator chooses right constructor based on parameters (overloading).
- Constructors can only be invoked by the **new** operator.

Basic approach

1) Define class.
2) Declare objects.
3) Create objects.
4) Use objects.

Example (ClassBox)

1) A Simple Class (Define class**)**

```
// class Box.java with three attribute width,height,depth
class Box {
double width;
double height;
double depth;
}
```

2) Declare,Create and use objects

```
// A program that uses the Box class.Call this file BoxDemo.java
class BoxDemo1 {
public static void main(String args[]) {
Box mybox = new Box();//declares an object of type Box.
double vol;
// assign values to mybox's instance variables
mybox.width = 10;
mybox.height = 20;
mybox.depth = 15;
// compute volume of box
vol = mybox.width * mybox.height * mybox.depth;
System.out.println("Volume is " + vol);
}// end of main
}// end of class
```

3) Declares two(more than one) Box objects

```
// This program declares two Box objects.
// this file named BoxDemo2.java
class BoxDemo2 {
   public static void main(String args[]) {
   Box mybox1 = new Box();
   Box mybox2 = new Box();
double vol;
// assign values to mybox1's instance variables
mybox1.width = 10;
mybox1.height = 20;
mybox1.depth = 15;
// assign different values to mybox2's   instance variable
mybox2.width = 3;
mybox2.height = 6;
mybox2.depth = 9;
// compute volume of first box
vol = mybox1.width * mybox1.height * mybox1.depth;
System.out.println("Volume is " + vol);
// compute volume of second box
```

```
vol = mybox2.width * mybox2.height * mybox2.depth;
System.out.println("Volume is " + vol);
}// end of main
}// end of class
```

4) Adding a Method to the Box Class

```
// This program includes a method inside the box class.
class Box {
   double width;
   double height;
   double depth;
// display volume of a box
   double  volume() {
   returnwidth * height * depth);
}
}
```

5) Call method

```
// BoxDemo3.java
class BoxDemo3 {
   public static void main(String args[]) {
   Box mybox1 = new Box();
   Box mybox2 = new Box();
      Double vol;
   // assign values to mybox1's instance variables
   mybox1.width = 10;
   mybox1.height = 20;
   mybox1.depth = 15;
   // assign different values to mybox2's instance variables
   mybox2.width = 3;
   mybox2.height = 6;
   mybox2.depth = 9;
   // display volume of first box
vol = mybox1.volume();
   System.out.println("Volume is " + vol);
```

```
  // display volume of second box
  vol = mybox2.volume();
  System.out.println("Volume is " + vol);
  }// end of main
}
```

6) Constructors

```
  /* Here, Box uses a constructor to initialize thedimensions of a
box.*/
  class Box {
  double width;
  double height;
  double depth;
  // This is the constructor for Box.
  void Box() {
  System.out.println("Constructing Box");
  width = 10;
  height = 10;
  depth = 10;
  }
  // This is the constructor with Parameter for Box.
  void Box(double w, double h, double d) {
  width = w;
  height = h;
  depth = d;
  }
  // compute and return volume
  double volume() {
  return width * height * depth;
  }// endvolume method
  }// end of class box
```

```
  class BoxDemo4 {
  public static void main(String args[]) {
  // declare, allocate, and initialize Box objects
```

```
    Box mybox1 = new Box();
    Box mybox2 = new Box(3, 6, 9);
    double vol;
    // get volume of first box
    vol = mybox1.volume();
    System.out.println("Volume is " + vol);
    // get volume of second box
    vol = mybox2.volume();
    System.out.println("Volume is " + vol);
    }
    }
```

7) Access or Visibility Specifiers (public/protected/ private)

```
public class Box {
privatedouble width;
privatedouble height;
privatedouble depth;
          // This is the constructor for Box.
public  void Box() {
      System.out.println("Constructing Box");
     width = 10;
     height = 10;
     depth = 10;
   }
          // This is the constructor with Parameter for Box.
public  void Box(double w, double h, double d) {
          width = w;
          height = h;
          depth = d;
  }
             // compute and return volume
public double volume() {
         return width * height * depth;
    }// end volume method
}// end of class box
```

8) **Kinds of class's methods**

- The *get* and *set* methods are used to read and modify private properties
- Get/accessor: A method that lets clients examine object state.
  - Examples: display,getName()
- Set/mutator: A method that modifies an object's state.
  - Examples: setName(), Salary()
  - often has a non-void return type

```
public class Box {
    private double width;
    private double height;
    private double depth;
            // This is the constructor for Box.
   public  void Box() {
      width = 10;
   height = 10;
depth = 10;
   }
            // This is the constructor with Parameter for Box.
    public  void Box(double w, double h, double d) {
          width = w;
          height = h;
          depth = d;
  }
// the get method retuen value
public  double getWidth()  {
 return width;
}
public  double getHigth()  {
return height;   }
public  double getDepth(){
return depth;   }
// the set method receive and change  values
public  void setWidth(double w) {
width = w;   }
```

```
public  void setHigth(double h) {
height=h;
}
public  void setDepth(double d) {
depth=d;
}
            // compute and return volume
   public double volume() {
         return width * height * depth;
    }// end volume method
}// end of class box
```

```
class BoxDemo5 {
public static void main(String args[]) {
// declare, allocate, and initialize Box objects
Box mybox1 = new Box();
Box mybox2 = new Box(3, 6, 9);
double vol;
// get volume of first box
vol = mybox1.volume();
System.out.println("Volume is " + vol);
// get volume of second box
vol = mybox2.volume();
System.out.println("Volume is " + vol);
                // change  depth of box1 from 10 to 15
    mybox1.setDepth(15);
                  // print the new depth
     double  dep =   mybox1.getDepth();
System.out.println("the new depth is  is " + dep);
// get volume of first box
vol = mybox1.volume();
System.out.println("The new Volume is " + vol);
}// end of main
}// end of class
```