Department of Computer Science
Amrita School of Engineering
Amritapuri Campus

Lab 2
Classes and Object

19CSE204
Object Oriented Paradigm 2-0-3-3

Java User Input

- The **Scanner class** is used to get user input, and it is found in the **java.util package**.
- To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner class documentation.

To read String input

- As our intention is to read input string from user, we will use the **nextLine()** method, which is used to **read Strings**:
- To get the instance of Java Scanner which reads input from the user, we need to pass the input stream (**System.in)** in the constructor of Scanner class. For Example:
- Scanner in = new Scanner(System.in);
- To get the instance of Java Scanner which parses the strings, we need to pass the strings in the constructor of Scanner class. For Example:
- Scanner in = new Scanner("Amrita University");

Method	Description
nextBoolean()	Reads a boolean value from the user
nextByte()	Reads a byte value from the user
nextDouble()	Reads a double value from the user
nextFloat()	Reads a float value from the user
nextInt()	Reads a int value from the user
nextLine()	Reads a String value from the user
nextLong()	Reads a long value from the user
nextShort()	Reads a short value from the user

```
package Pack;
import java.util.Scanner;
public class Driver {
        public static void main(String[] args) {
          Scanner myObj = new Scanner(System.in);

          System.out.println("Enter name, age and salary:");

          // String input
          String name = myObj.nextLine();

          // Numerical input
          int age = myObj.nextInt();
          double salary = myObj.nextDouble();

          // Output input by user
          System.out.println("Name: " + name);
          System.out.println("Age: " + age);
          System.out.println("Salary: " + salary);
        }
      }
```

Class in Java

- **A class defines a new data type. Once defined, this new type can be used to create objects of that type.**
- Thus, a class is a template for an object, and an object is an instance of a class

- **The General Form of a Class**
- You define a class, you declare its exact form and nature. You do this by specifying the data that it contains and the code that operates on that data
    - **class** Classname {
      type instance-variable1;
      type instance-variable2;
      // ...
      type instance-variableN;
      type methodname1(parameter-list) {
      // body of method
      }
      type methodname2(parameter-list) {
      // body of method
    - }
      // ...
      type methodnameN(parameter-list) {
      // body of method
      }
      }

- Variables, defined within a class are called **instance variables**
- The code is contained within **methods**
- Collectively, the methods and variables defined within a class are called **members of the class**

A Simple Class Creation

- Here is a class called Box that defines three instance variables: **width, height, and depth**. Currently, Box does not contain any methods (but some will be added soon).
- Class definition
- **class** Box
- **{**
  double width;
  double height;
  double depth;
  }

Create a Box object

- **Box** mybox **= new Box(); // create a Box object called mybox**

every Box object will contain its own copies of the instance variables width, height, and depth. To **access these variables, you will use the dot (.) operator.**

mybox.width = 100; **//This statement tells the compiler to assign the copy of width that is contained within the mybox object the value of 100**

- **Lab Exercise I**
- Create a Java Project in Eclipse
- Create a Default Package
- Create a class names Box in the default package- > In this class define the class Box as seen on the right side
- Create a Driver class where you have the main() function
- In the main() create instance of Box class, assign values to the height width and depth. Calculate the volume from the height width and depth

2. Adding a Method to the Box Class

- You can add a method (function) to the class for accessing the instance variables defined by the class .
- void volume()
- {
- return(width*height*depth)
- }
- Note that the above method volume() defined inside the class can access the data members width, height and depth without the (dot) operator.

- **Lab Exercise II**
- Add the function volume() in the class Box
- Create two instances of the class in the main() function
- Assign values to the data members of the both the instances in the main()
- Invoke the function volume() in the main() to compute the volume of both the boxes and display on the screen

- **Set the dimension using SetDim(w,h,d) function**
- void setDim(double w, double h, double d)
- {
  width = w;
  height = h;
  depth = d;
  }

- **Lab Exercise III**
- Modify the **Box** class and driver class to include **setDim(w,h,d)**

4.Constructor

- A constructor **initializes an object immediately upon creation**. Once defined, the constructor is automatically called immediately after the object is created, before the new operator completes.
- **It has the same name as the class i**n which it resides and is syntactically **similar to a method**
- **Constructors have no return type, not even void.**

1) Default Constructor

- **a) Default Constructor Definition**
- For the class Box the default constructor defintion
- Box() {
  System.out.println("Constructing Box");
  width = 10;
  height = 10;
  depth = 10;
  }
- **b) Default Constructor Invocation**
- Box mybox1 = new Box();
- default constructor automatically initializes all instance variables to zero.

2. Parameterized Constructors

- While the Box( ) constructor in the preceding example does initialize a Box object, it is not very useful—all boxes have the same dimensions. What is needed is a way to construct Box objects of various dimensions. The easy solution is to add parameters to the constructor.
- the following version of Box defines a parameterized constructor which sets the dimensions of a box as specified by those parameters

- **(a) Parameterized constructor Definition**
- // This is the constructor for Box.
  Box(double w, double h, double d) {
  width = w;
  height = h;
  depth = d;
  }
- (b) **Parameterized constructor Invocation**
- object is initialized as specified in the parameters to its constructor.
- // declare, allocate, and initialize Box objects
  Box mybox1 = new Box(10, 20, 15);
- the values 10, 20, and 15 are passed to the Box( ) constructor when new creates the object. Thus, mybox1's copy of width, height, and depth will contain the values 10, 20, and 15, respectively.

- **Lab Exercise IV**
- In the class definition of Box give both default and parameterized constructors
- In the driver.java create an instance of class (Object) with default constructor
- In the driver.java create an instance of class (Object) with parameterized constructor
- Compute the volume of both the objects

5. Introducing Access Control

- Encapsulation provides another important attribute: **access control**. Through encapsulation, you can control what parts of a program can access the members of a class. By controlling access, you can prevent misuse
- For example, **allowing access to data only through a well-defined set of methods, you can prevent the misuse of that data.**
- Java's access specifiers are
- **public,**
- **private,**
- **protected.**
- **default**

- When a member of a class is modified by the **public** specifier, then
  - that member can be accessed by any other code
- When a member of a class is specified as **private**, then
  - that member can only be accessed by other members of its class.
- The members of a class have used the **default** access mode,
  - which is essentially public.
- When a member of a class is specified as **protected**
  - protected applies only when inheritance is involved (We will see later)

- **(a) Giving access specifiers to data members**
- public int b; // public access
  private int c; // private access
- **Example of having private and public data members in a class**
- If you have a private data member in a class, it cannot be accessed in another class. Even in the driver class main(). Hence we need to access it through a member function

Box.java

```
public class Box {

    private double width;
    double height;
    public double depth;
double volume() //Method for computing Volume
{
    return(width*height*depth);
}
void getwidth(int w)
{
    width=w;
}
}
```

Driver.java

```
public class Driver {
    public static void main(String[] args)
{
        double vol1;

        Box mybox1 = new Box();// Object Creation by invoking default constructor

        mybox1.depth=10; // This is correct.We can access public data members in another class
        mybox1.height=20; // This is correct. By default data members access specifier is public
        //mybox1.width=30; // This is not correct. We cannot access private members outside a clas

        mybox1.getwidth(30);
        vol1=mybox1.volume();
        System.out.println("Volume is " + vol1);

    }
}
```

Namah Shivaya!