Class Fundamentals

A class is a user-defined data type with a template that serves to define its properties. Once the class type has been defined, we can create “variables” of that type using declarations that are similar to the basic type declarations. These variables are termed as instances of classes, which are the actual objects.

The basic form of class definitions is:

```
Class classname [extends superclassname]
{
        [Fields declaration:]
        [Methods declaration:]
}
```

Everything inside the square brackets is optional.

Classname and superclassname are any valid Java Identifier. The keyword **extends** indicates that the properties of the superclassname class are extended to the classname class. Fields and methods are declared inside the body.

Fields declaration

We can declare the instance variables exactly same way as we declare local variables. Example:

```
Class Rectangle

{

        int length;

        int width;

}
```

Method declaration

Methods are declared inside the body of the class but immediately after the declaration of instance variables. The general form of a method declaration is

```
Type methodname (parameter list)

{

        Method body
```

}

Method declarations have four basic parts:

- The name of the method (method name)
- The type of the value the method returns (type)
- A list of parameters (parameter- list)
- The body of the method.

The methodname is a valid identifier. The parameter list is always enclosed in parenthesis. The body actually describes the operations to be performed on the data.

A class to compute area of rectangle can be defined as:

```
Class Rectangle
{
        int length, width;
        void getdata (int x, int y)
        {
                length=x;
                width=y;
        }
        int rectArea()
        {
                int area=length* width;
                return(area);
        }
}
```

Creating objects

An object in java is essentially a block of memory that contains space to store all the instance variables. Creating an object is also referred to as instantiating an object.

Objects in Java are created using the **new** operator. The **new** operator creates an object of the specified class and returns a reference to that object. Example of creating an object of type **Rectangle** as follows:

```
        Rectangle rect1;                // declare the object
        rect1= new Rectangle()          //instantiate object
```

The first statement declares a variable to hold the object reference and the second one actually assigns the object reference to the variable. The variable **rect1** is now an object of the **Rectangle** class.

Both statements can be combined into one as shown below:

Rectangle rect1=new Rectangle ();

Accessing class members

In order to access class members, we must use the concerned object and the *dot* operator as shown below:

Objectname.variablename=value;
Objectname.methodname(parameter-list);

Here, object name is the name of the object, variablename is the name of the instance variable inside the object that we wish to access, methodname is the method that we wish to call, and parameter-list is a comma separated list of actual values that must match in type and number with the parameter list of the methodname declared in the class.

The instance variables of the Rectangle class may be accessed and assigned values as follows:

```
        rect1.length=15;
        rect1.width=10;
```

Java program to calculate the area of rectangle

```
Class Rectangle
{
    int length, width;
    void getdata (int x, int y)
    {
        length=x;
        width=y;
    }
    int rectArea()
    {
        int area=length* width;
        return(area);
    }
}
Class RectArea
{
Public static void main( String args[])
{
    int area1, area2;
    Rectangle rect1=new Rectangle()
    Rectangle rect2=new Rectangle();
    rect1.length=15;
```

```
        rect1.width=10;
        area1=rect1.length*rect1.width;
        rect2.getdata(20,12);
        area2=rect2.rectArea();
        System.out.println(“Area1= “ +area1);
        System.out.println(“Area1= “ +area2);
}
}
```

OUTPUT

```
Area1=150
Area2=240
```