Java Fundamentals

Siddharth Sharma

CLASSES

- Defines the **shape** and **nature** of the object.
- It defines a **new data type**.
- Once defined, this new type can be used to create objects of that type.
- A class is a ***template*** for an **object**, and an **object** is an ***instance*** of a class.
- Because an object is an instance of a class, you will often see the two words ***object*** and ***instance*** used **interchangeably**.

- **The General Form of a Class:**

When 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. A class' code defines the interface to its data. A class is declared by use of the **class** keyword.

A simplified general form of a **class** definition is:

```
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
        }
```

}

- The data, or 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.
- **Variables defined within a class are called instance variables because each instance of the class (that is, each object of the class) contains its own copy of these variables. Thus, the data for one object is separate and unique from the data for another.**

NOTE:

```
class box
{
   int height;
   int length;
   int breadth;
}
```

As stated, a class defines a **new type of dat**a. In this case, the **new data type** is called **Box**. You will use this name to declare objects of type **Box**. It is important to remember that a **class** declaration only creates a template; it does not create an actual object. **Thus, the preceding code does not cause any objects of type Box to come into existence.**

To actually create a Box object, you will use a statement like the following:

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

After this statement executes, **mybox** will be an instance of **Box**. Thus, it will have "physical" reality.

Declaring Objects:

Obtaining objects of a class is a two-step process. First, you must declare a variable of the class type. This variable does not define an object. Instead, it is simply a variable that can *refer* to an object. Second, you must acquire an actual, physical copy of the object and assign it to that variable. You can do this using the **new** operator. The **new** operator dynamically allocates (that is, allocates at run time) memory for an object and returns a reference to it. This reference is, more or less, the address in memory of the object allocated by **new**. This reference is then stored in the variable. Thus, in Java, all class objects must be dynamically allocated.

```
Box mybox; // declare reference to object
mybox = new Box(); // allocate a Box object
```

Combining both the statements we can write it as:

```
Box mybox = new Box();
```

The first line declares **mybox** as a reference to an object of type **Box**. After this line executes, **mybox** contains the value **null**, which indicates that it does not yet point to an actual object. Any attempt to use **mybox** at this point will result in a compile-time error. The next line allocates an actual object and assigns a reference to it to **mybox**. After the second line executes, you can use **mybox** as if it were a **Box** object. But in reality, **mybox** simply holds the memory address of the actual **Box** object.

Introducing Methods :

This is the general form of a method:

type name(*parameter-list*)
{
    / / body of method
}

- *Type* specifies the type of data returned by the method. This can be any valid type, including class types that you create.
- If the method does not return a value, its return type must be **void**.
- The name of the method is specified by *name.* This can be any legal identifier other than those already used by other items within the current scope.
- The *parameter-list* is a sequence of type and identifier pairs separated by commas. Parameters are essentially variables that receive the value of the *arguments* passed to the method when it is called.

Adding a Method to the Box Class :

```
class box
{
   int height;
   int width;
   int length;
   void volume()
   {
      System.out.print("Volume is ");
      System.out.println(height*width*length);
   }
}
class example
{
   public static void main(String args[])
   {
      box box1= new box();
      box box2= new box();

      box1.height=10;
      box1.width=20;
      box1.length= 30;
      box2.height=10;
```

```
        box2.width=10;
        box2.length=10;

        box1.volume();
        box2.volume();
    }
}
```

OUTPUT:

```
Volume is 6000
Volume is 1000
```

Returning a Value:

```
class box
{
    int height;
    int width;
    int length;
    int volume()
    {
        int volume;
        volume=height*width*length;
        return volume;
    }
}
class example
{
    public static void main(String args[])
    {
        box box1= new box();
        box box2= new box();
        int volume1, volume2;
        box1.height=10;
        box1.width=20;
        box1.length=30;
        box2.height=10;
        box2.width=10;
        box2.length=10;
        volume1= box1.volume();
        volume2= box2.volume();
        System.out.println("Volume is "+volume1);
        System.out.println("Volume is "+volume2);
    }
}
```

OUTPUT:

```
Volume is 6000
```

```
Volume is 1000
```

Adding a Method That Takes Parameters:

```
class box
{
   int height;
   int width;
   int length;
   int volume()
   {
       return height*width*length;
   }
   void setdimension(int a, int b, int c)
   {
      height=a;
      width=b;
      length=c;
   }
}
class example
{
   public static void main(String args[])
   {
      box box1= new box();
      box box2= new box();

      box1.setdimension(10,20,30);
      box2.setdimension(10,10,10);

      System.out.println("Volume is "+box1.volume());
      System.out.println("Volume is "+box2.volume());
   }
}
```

OUTPUT:

```
Volume is 6000
Volume is 1000
```

NOTE:

It is important to keep the two terms *parameter* and *argument* straight. **A *parameter* is a variable defined by a method that receives a value when the method is called.** For example, in square( ), i is a parameter. **An *argument* is a value that is passed to a**

method when it is invoked. For example, square(100) passes 100 as an argument. Inside square( ), the parameter i receives that value.