CS205 Object Oriented Programming in Java

Module 2 - **Core Java Fundamentals (Part 5)**

Prepared by Renetha J.B.

Topics

- Core Java Fundamentals:

✓ **Object Oriented Programming in Java**

- **Class Fundamentals**
- **Declaring Objects**
- **Object Reference**
- **Introduction to Methods**.

Class Fundamentals

- The **class** is the core of Java.
  - The class forms the basis for object-oriented programming in Java.
- A class is a "blueprint" for creating objects
- A **class** is a ***template for an object.***
  - An **object** is an ***instance of a class.***
- A class defines a *new type of data*.
- A class creates a ***logical framework*** that defines the relationship between its members.

Example

Student

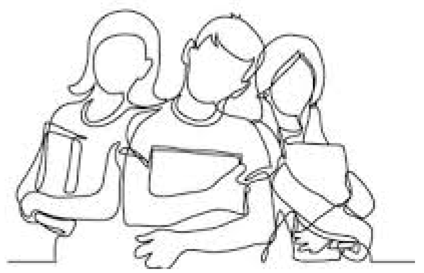
(class)

Rollno
Name

properties
(instance variable

read()
write()

behaviour
(methods)

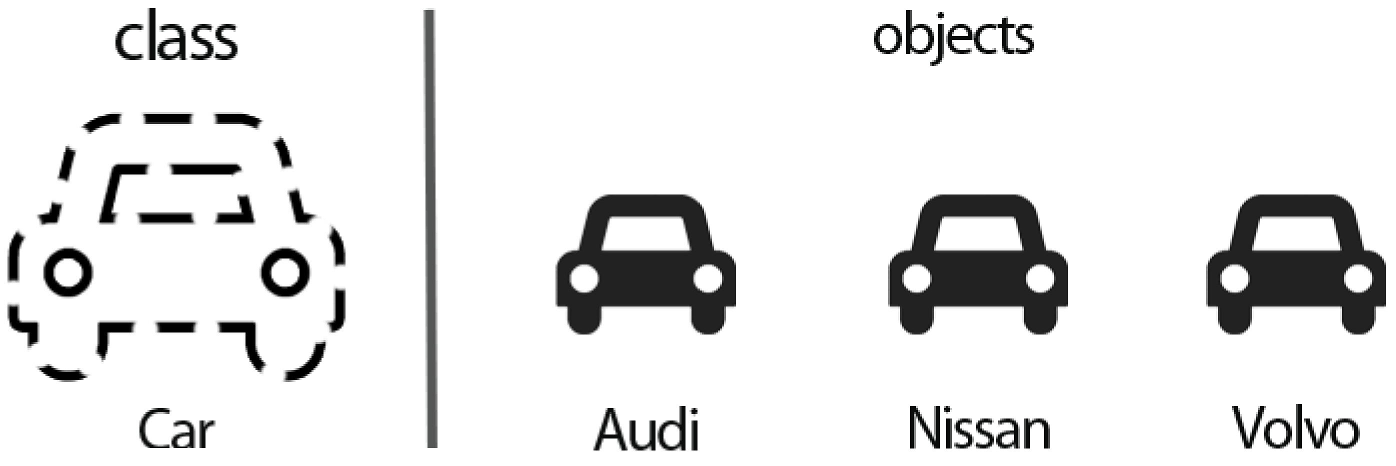


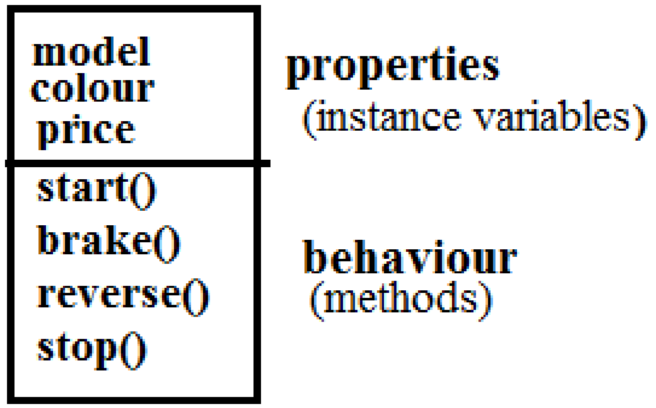
12
Smith

object

2
Susan

object


class
objects
Car
Audi
Nissan
Volvo


model
colour
price
start()
brake()
reverse()
stop()
properties
(instance variables)
behaviour
(methods)

Class(continued.)

- A class is declared using the keyword **class**
- The data or 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.
  - the data for each object is separate and unique.
- Functions inside class are called **methods**.
- The methods and variables defined within a class are called **members of the class.**

The General Form of a Class

- **A** general form of a **class definition is**

```
class classname
{
     type instance-variable1;
     type instance-variableN;
     type methodname1(parameter-list)
     {
     // body of method
     }
     type methodnameN(parameter-list)
     { // body of method
     }
}
```

Properties **(instance variables)**

Behaviour **or Method** or **function**

Behaviour or **Method** or **function**

Members of class

A Simple Class

```
class Box
{
double width;
double height;
double depth;
}
```

$\}$ **Properties (instance variables)** $\leftarrow$ **Member** of class

A Simple Class

```
class Box
{
double width;
double height;
double depth;

void  volume()
{
//statements
}


}
```

Properties
(instance variables)

Behaviour or
Method or
Function

Members
of class

Declaring Objects

- When we create a class, we are creating a new data type.
  - We can use this type to declare objects of that type.
- Obtaining objects of a class is a two-step process.
  - *First*, we must **declare a variable of the class type.**
    - This variable does not define an object.
    - It is simply a variable that can *refer to an object.*
  - *Second*, we must **acquire an actual, physical copy of the object and assign it to that variable** (using **new** operator)

Declaring Objects(contd.)

Classname objectname ; *// declare reference to object*

objectname = **new** Classname(); *// allocate an object*

- We can write this in a single statement

Classname objectname = **new** Classname();

```
class Box
{double Width;
double Height;
double Depth;
}
Box mybox;
```

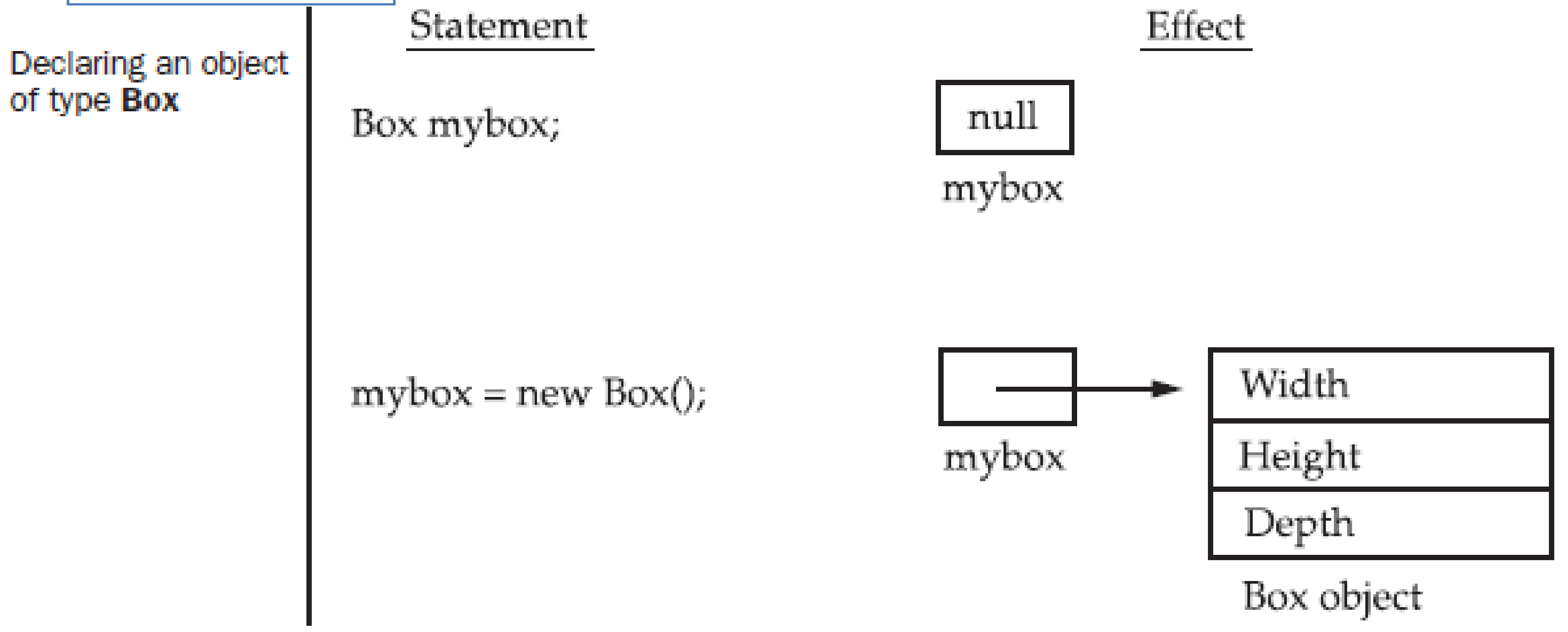
- This line declares **mybox** <u>as a **reference to** an object of type **Box**.</u>
- Here **mybox** contains the value **null**, which indicates that it does not yet point to an actual object

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

- This line allocates an actual object and assigns a reference to it to **mybox.**
- **mybox** holds the memory address of the actual Box object.

Declaring Objects(contd.)

```
class Box
{double Width;
double Height;
double Depth;
}
```



Declaring Objects(contd.)

- The class name followed by parentheses specifies the *constructor* for the class.

Box mybox=new Box();

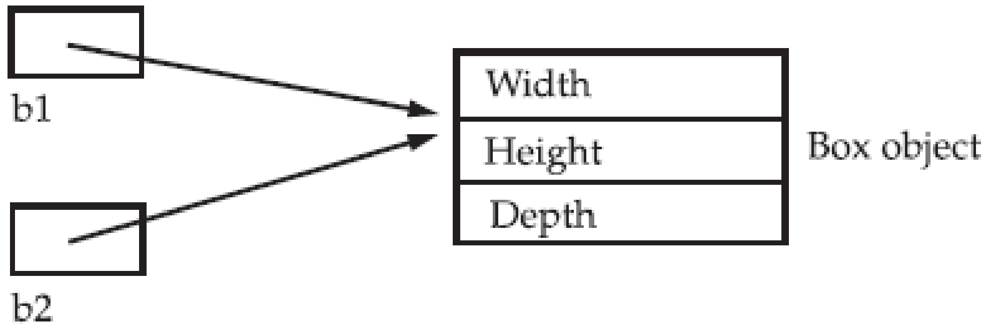
- Here Box is the class. Box()is the constructor.
- A constructor defines what occurs when an object of a class is created.

Assigning Object Reference Variables

- Object reference variables act differently when an assignment takes place
- E.g.

Box b1 = new Box();

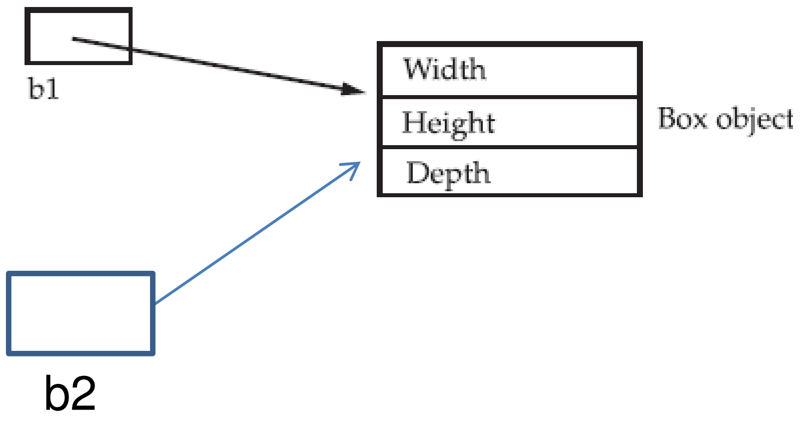
Box b2 = b1;

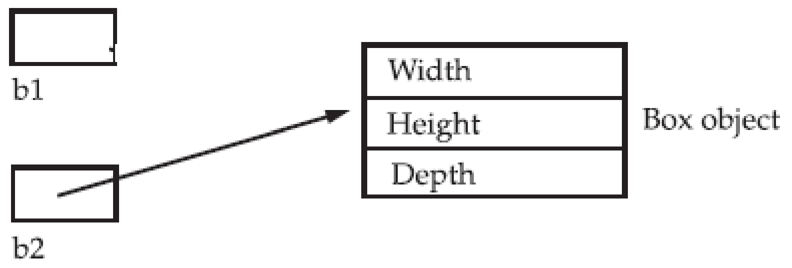


- Here b1 and b2 will both refer to the ***same object.***
- Any changes made to the object through b2 will affect the object which is referred by b1, because they are the same object.

Assigning Object Reference Variables (contd.)

```
Box b1 = new Box();
Box b2 = b1;
// ...
b1 = null;
```





- Here at the end b1 has been set to null, but b2 still points to the original object.

Class vs object

Class	Object
Template for creating objects	**Instance** of class
Logical entity	**Physical** entity
Declared using **class** keyword	Created using **new** operator.
Class **does not get any memory** when it is created.	Object **gets memory** when it is created using new operator.
A class is **declared only once**	**Many objects** can be created from a class

Introducing Methods

- **Classes** usually consist of two things:
  - Instance variables
  - Methods or functions.
- The general form of a method:

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

- The *type* specifies the type of data returned by the method.
  - any valid type, including class types, void
- The *parameter-list or argument list is a* sequence of type and identifier pairs separated by commas.

Introducing Methods(contd.)

- Methods that have a return type other than void return a value to the calling routine using the following form of the return statement:

  return value;

- Method of one class can be invoked by functions of other classes through objects of former class.

Objectname.method(parameters);

```
// EXAMPLE
class Box {
        double width;
        double length;
        double depth;

        void volume()
        {
        System.out.print("Volume is ");
        System.out.println(width * height * depth);
        }
}
class BoxDemo {
public static void main(String args[]) {
Box mybox1 = new Box();
mybox1.width = 10;
mybox1. length = 30;
mybox1.depth = 15;
mybox1.volume();
} }
```

Properties
(**instance variables**)

Behaviour or
Method or
Function

Box class

Behaviour or
Method or
Function
MAIN FUNCTION

Object of class **Box**

BoxDemo class

Example

- Create a class Box with instance variables length, width and height. Include a method volume to compute the volume of the box,
- Create another class BoxDemo with main function that creates an object of class Box named mybox1 and set the values for instance variables(length, width and height). Invoke the function volume in Box to compute the volume of the created object mybox1

```
class Box {
        double width;
        double length;
        double depth;

        void volume()
        {
        System.out.print("Volume is ");
        System.out.println(width * length * depth);
        }
}
class BoxDemo {
public static void main(String args[]) {
Box mybox1 = new Box();
mybox1.width = 10;
mybox1. length = 30;
mybox1.depth = 15;
mybox1.volume();
}
}
```

OUTPUT

```
Volume is 3000.0
```

```
// program using return statement
class Box {
        double width;
        double height;
        double depth;

        void volume()
        {
        return(width * length * depth);
        }
}
class BoxDemo {
public static void main(String args[]) {
Box mybox1 = new Box();
mybox1.width = 10;
mybox1.height = 20;
mybox1.depth = 15;
int v=mybox1.volume();
System.out.println(”Volume=“+v);
} }
```

OUTPUT

```
Volume is 3000.0
```

Reference

- Herbert Schildt, Java: The Complete Reference, 8/e, Tata McGraw Hill, 2011.