Object Oriented Programming in Java

Class

- Classes are powerfull in java, they not only encapsulate main() function but are also used to make objects, which are backbone of Object oriented programming.
- Class can be thought of as it defines new data type. Once defined it can be used to create any number of objects.
- Class is a template for an object (or blueprint)

General form of class

```
class classname{
    type instance-variable1
    type instance-variable2
    .....
    type instance-variableN
    //constructor()
    type methodname1(parameter list){ //body of method }
    .....
    type methodname1(parameter list){ //body of method }
    }
```

Members: Collectively the methods and variables defined within a class are called members of a class.

Instance Variable: The data or variables defined within a class are called iinstance-variables.

- These are called instance variables because each instance (object) of a class will have its own copy of these variables.
- Data for each object is seperate and unique(different from each other)

Java class does not need a main() method, we only specify main() method in java class if that class is a starting point of our program.

- Ex:

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

new type of data Box has 3 instance variables -> width, height, length and no methods.

Template(**Not an actual object**)

Box mybox = new Box(); //createas a Box object

- Dot operator is used to access the copies of instance variables of each object.

  Example: mybox.width = 100;

- Refer Box class examples

Object

- Objects are runtime entities in object oreinted system.
- A JAVA object is called an instance of a class.
- A Problem can be thought of in terms of Objects and type of communication between these objects.
- Example: A User Class is used to make objects then different users are treated as objects each having unique properties like firstname, lastname, DOB, phone, ID, etc.

Object Creation

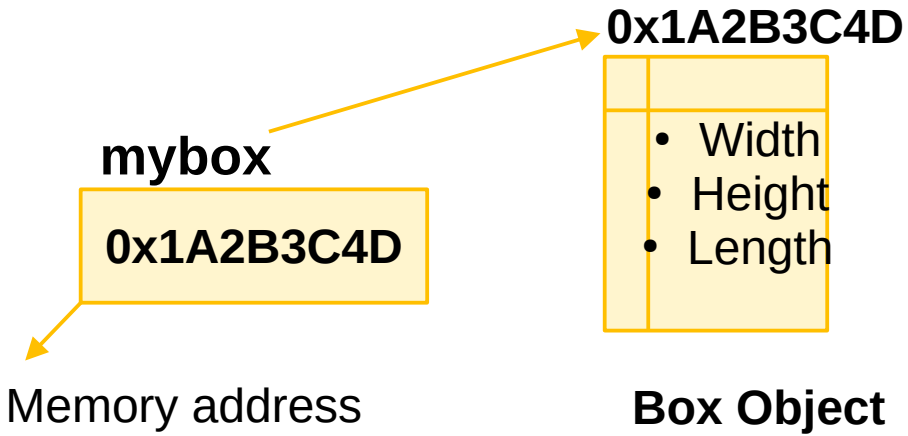
- Declaration:

  Syntax: type ObjectName;

  - Example: Box mybox;
- Defining an object:

  Syntax: ObjectName = new ClassName(); //i.e. constructor

  - Example: mybox = new Box();



mybox is simply a variable of type Box That can refer to an object

new keyword dynamically allotcates memory(at runtime) i.e. it acquires actual physical Copy of object and assigns it to mybox variable.

method

- Method is a block of code in java which performs a certain task or operation.
- Methods are used to achieve reusability of code.
- Provides easy modification and reusability of code.
- A method gets executed only when we call or invoke it.
- We can use methods to access instance variables defined by the class.
- Methods defines interface to most classes.

General form of method

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

Sequence of type and identifier pairs
Seperated by commas
Ex: int x, int y, float z

Type of data returned by method
i.e. any primitive or non primitive type

- Example: double vol = box1.volume()
- The type of data returned by method must be compatible with return type specified by method
- Variable receiving the value returned by a method must also be compatible with return type.

Parameter vs Argument

- **Parameter:** is a variable defined by method that receives a value when the method is called

  example: int square(int i){

  return i*i

  }

  Here i is parameter

- **Argument:** is a value that is passed to a method when it is invoked.

  Example: square(5); //5 has been passed as an argument and parameter i receives the argument 5

Methods can also be parameterized

- w, h, l are parameters and the instance variables width, height and length are being set by the arguments they will recieve respectively.

```
void setDim(double w, double h, double l){
    width = w;
    height = h;
    length = l;
}
```

Constructor

- A constructor intializes an object immediately upon creation.
- It has the same name as the class in which it resides and is syntactically very similar to a method.
- Once a constructor is defined, it automatically gets called when the object is created.
- Constructor do not ahave a primitive return type.
- It has an implicit return type of class type of the same class it belongs to.

- A constructor’s job is to initialize the internal state of an object so that the code fcreating an instance (object) will have a fully initialized, usable object immediately.
- Constructor can be simple (refer to code: ConstructorDemo1)
- Constructor can also be parameterized (refer to code: ConstructorDemo2)
- When no constructor is defined java automatically creates a default constructor with default values:
  - numeric types -> 0
  - Boolean -> false;
  - Reference Types -> null

Garbage Collection

- When no reference to an object exists after the program execution is over and objects are assumed to be no longer needed, the memory has to be reclaimed for other purposes.
- So java has Garbage collection for this purpose. And JVM automatically runs it.
- C++ has delete operator and C has free() function.
- Object creation in Java is done useing new keyword and it dynamically allocates space in memory but at some time when they are no longer needed they need to be destroyed and memory has to be reclaimed, hence Garbage collection plays an important role by taking care of this memory recycling.

Finalize method

- Sometimes object need to perform some action, or get collected before it is destoryed.
- Example: File Handling- A non java resource could be reserved and have to be freed before object is destroyed,
- This is achieved by finalization.
- To add finalizer to a class define finalize() method.
- JRE calls the finalize method when it is about to recycle the object.

- finalize () -> actions to be carried before destroction of object.
- Garbage collector runs periodically so if you need some action to be performed before object is destroyed then you have to define finalize()
- Note*:- At program level there are other methods to release control of resources instead of depending on Garbage collector.

- Finalize() is considered outdated and has been largely deprecated. It is not recommended to rely on or use finalize() for resource management or cleanup in modern Java applications.
- Reasons: Unpredictable Execution, Performance Overhead, Limited Guarantee

packages

- Packages are containers for classes.
- They are used to resolve the problem of namespace collision.
- To create a package is quite easy: simply include a package command as the first statement in a Java source file.
- If you omit the package statement, the class names are put into the default package, which has no name.
- There are already many packages which have utility functions for input output and other functions such as: java.util
- package util has following directory path in Windows OS: C:\<path to JDK>\java\util\