Unit-II

Objects and Classes

A class is a group of objects which have common properties. It is a template from which objects are created. It is a logical entity. It can't be physical.

A class in Java can contain:

Fields

Methods

Constructors

Blocks

Nested class and interface

```
class <class_name>{
    field;
    method;
}
```

Defining Classes

- Class defines a new data type
- A class define data and method

```
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
  }
}
```

Members

Declaring Objects

- Declare reference to object

```
ClassName objectName;
```

- Allocate an actual object

```
ObjectName = new ClassName();
```

e.g.

```
Value obj;
obj = new Value();
```

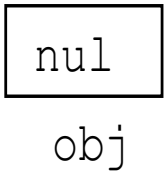
You can combine both the statement also:

```
Value obj = new Value();
```

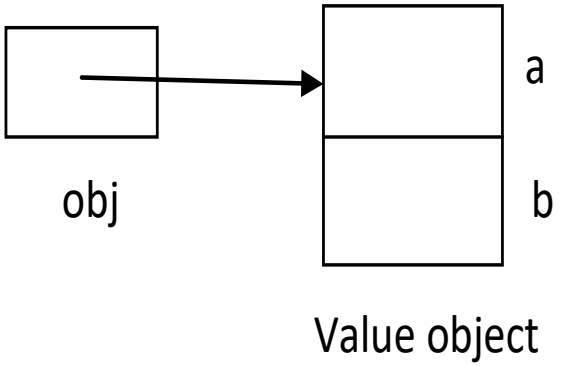
Statement

Effect

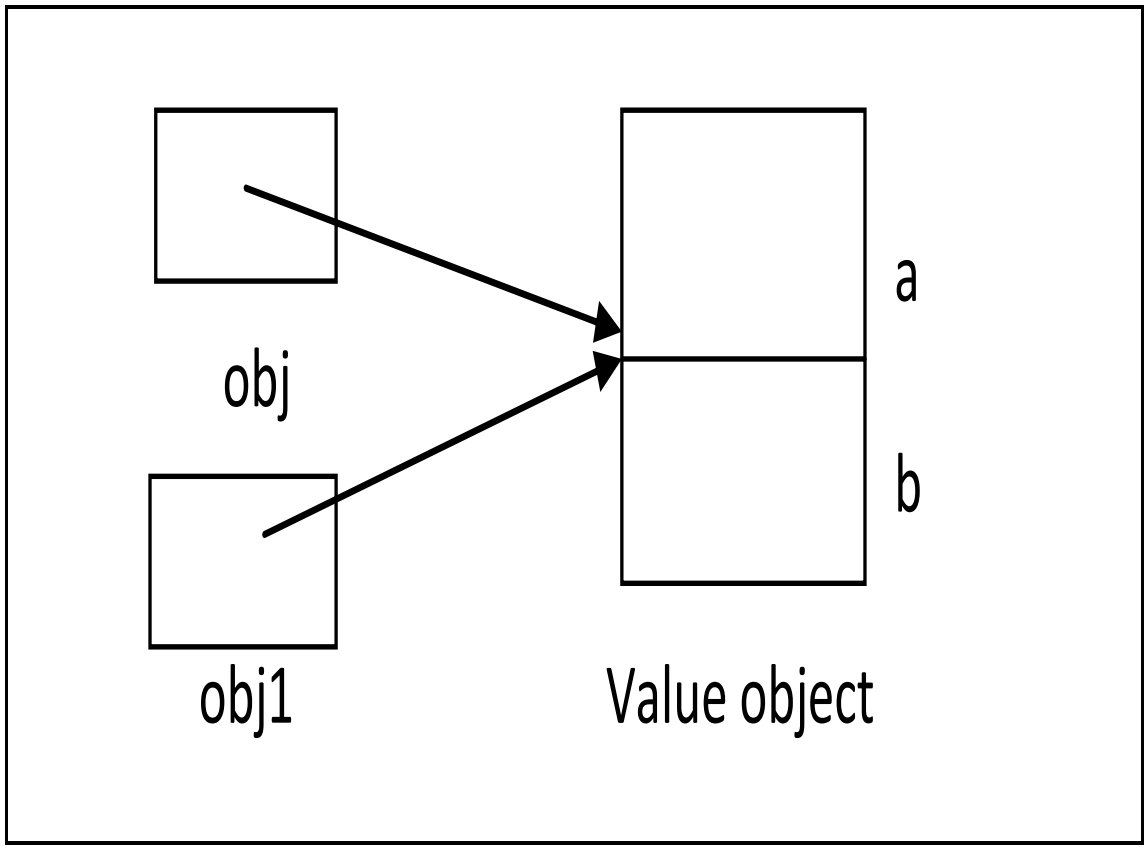
```
Value obj;
```




```
obj = new Value();
```



If obj1 = obj;



new keyword in java

- The Java new keyword is used to create an instance of the class. In other words, it instantiates a class by allocating memory for a new object and returning a reference to that memory. We can also use the new keyword to create the array object.

NewExample obj=**new** NewExample();

- It is used to create the object.
- It allocates the memory at runtime.
- All objects occupy memory in the heap area.
- It invokes the object constructor.
- It requires a single, postfix argument to call the constructor

Access Specifies

- **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
- **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
- **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
- **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

Constructors

- Construction is a method in a class which is used to initialize objects when it is created.
- **Overloading Constructors**
  - In Java a class can have more than one constructor.

- A constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.
- It is a special type of method which is used to initialize the object.
- Every time an object is created using the new() keyword, at least one constructor is called. It calls a default constructor if there is no constructor available in the class. In such case, Java compiler provides a default constructor by default.

There are rules to defined for the constructor.

- Constructor name must be the same as its class name
- A Constructor must have no explicit return type
- A Java constructor cannot be abstract, static, final, and synchronized

There are two types of constructors in Java:

- Default constructor (no-arg constructor)
- Parameterized constructor

```
class Bike{
//creating a default constructor
Bike(){System.out.println("Bike is created");}
//main method
public static void main(String args[]){
//calling a default constructor
Bike b=new Bike();
}
}
```

Purpose of a default constructor?

- The default constructor is used to provide the default values to the object like 0, null, etc., depending on the type.

```
class Student{
   int id;
   String name;
   //creating a parameterized constructor
   Student(int i,String n){
   id = i;
   name = n;
   }
     void display(){System.out.println(id+" "+name);}
      public static void main(String args[]){
      Student s1 = new Student4(101,"Karan");   //creating objects and passing valu
    es
   Student s2 = new Student4(212,"Arjun");
   //calling method to display the values of object
   s1.display();
   s2.display();
  }
}
```

```
class StudentS{
    int id;
    String name;
    int age;
    //creating two arg constructor
    StudentS(int i,String n){
    id = i;
    name = n;
    }
    //creating three arg constructor
    StudentS(int i,String n,int a){
    id = i;
    name = n;
    age=a;
    }
    void display(){System.out.println(id+" "+name+" "+age);}
      public static void main(String args[]){
    StudentS s1 = new Student5(101,"Karan");
    StudentS s2 = new Student5(202,"Aryan",25);
    s1.display();
    s2.display();
   }
}
```

```
class StudentS{
    int id;
    String name;
    //constructor to initialize integer and string
    StudentS(int i,String n){
    id = i;
    name = n;
    }
    //constructor to initialize another object  (copy constructor)
    StudentS(StudentS s){
    id = s.id;
    name =s.name;
    }
    void display(){System.out.println(id+" "+name);}
      public static void main(String args[]){
    StudentS s1 = new StudentS(111,"Karan");
    StudentS s2 = new StudentS(s1);
    s1.display();
    s2.display();
   }
}
```

- Difference between constructor and method

Constructor	**Method**
A constructor is used to initialize the state of an object.	A method is used to expose the behavior of an object.
A constructor must not have a return type.	A method must have a return type.
The constructor is invoked implicitly.	The method is invoked explicitly.
The constructor name must be same as the class name.	The method name may or may not be same as the class name.

static keyword

- The static keyword in Java is used for memory management mainly. We can apply static keyword with variables, methods, blocks and nested classes. The static keyword belongs to the class than an instance of the class.

The static can be:

- Variable (also known as a class variable)
- Method (also known as a class method)
- Block
- Nested class

static variable

- If you declare any variable as static, it is known as a static variable.
- The static variable can be used to refer to the common property of all objects (which is not unique for each object), for example, the company name of employees, college name of students, etc.
- The static variable gets memory only once in the class area at the time of class loading.

```
class Counter{
int count=0;//will get memory each time when the instance is
    created

Counter(){
count++;//incrementing value
System.out.println(count);
}

public static void main(String args[]){
//Creating objects
Counter c1=new Counter();
Counter c2=new Counter();
Counter c3=new Counter();
}
}
```

```
class Counter2{
static int count=0;//will get memory only once and retain its v
    alue

Counter2(){
count++;//incrementing the value of static variable
System.out.println(count);
}

public static void main(String args[]){
//creating objects
Counter2 c1=new Counter2();
Counter2 c2=new Counter2();
Counter2 c3=new Counter2();
}
}
```

static method

- If you apply static keyword with any method, it is known as static method.
- A static method belongs to the class rather than the object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- A static method can access static data member and can change the value of it.

static block

Is used to initialize the static data member.

It is executed before the main method at the time of classloading.

```
class A2{
  static{System.out.println("static block is invoked");
    }
  public static void main(String args[]){
   System.out.println("Hello main");
  }
}
```

- Why is the Java main method static?

➢ It is because the object is not required to call a static method. If it were a non-static method, JVM creates an object first then call main() method that will lead the problem of extra memory allocation.

Use of this keyword

- reference to the object on which the method was called
- **this** is used to resolve ambiguity between a local and instance variable.

- this(keyword) can be used to refer current class instance variable.
- this can be used to invoke current class method (implicitly)
- this() can be used to invoke current class constructor.
- this can be passed as an argument in the method call.
- this can be passed as argument in the constructor call.
- this can be used to return the current class instance from the method.

```
class Student{
int rollno;
String name;
float fee;
Student(int rollno,String name,float fee){
this.rollno=rollno;
this.name=name;
this.fee=fee;
}
void display(){System.out.println(rollno+" "+name+" "+fee);}
}

class TestThis2{
public static void main(String args[]){
Student s1=new Student(111,"ankit",5000f);
Student s2=new Student(112,"sumit",6000f);
s1.display();
s2.display();
}}
```

Predefined Classes

- The Java API provides a rich set of classes and interfaces organize in **packages**.
- The most important predefined class is the **Object** class.
- **Object** is the top of the class hierarchy in java.
- All classes in Java derived from **Object** class.
- The **Object** class defines the basic state and behaviour that all objects must have.

Method	Purpose
Object clone( )	Creates a new object that is the same as the object being cloned.
boolean equals(Object object)	Determines whether one object is equal to another.
void finalize( )	Called before an unused object is recycled.
Class getClass( )	Obtains the class of an object at run time.
int hashCode( )	Returns the hash code associated with the invoking object.
void notify( )	Resumes execution of a thread waiting on the invoking object.
void notifyAll( )	Resumes execution of all threads waiting on the invoking object.
String toString( )	Returns a string that describes the object.
void wait( ) void wait(long milliseconds) void wait(long milliseconds, int nanoseconds)	Waits on another thread of execution.

```
public class EqualsExample{
public static void main(String args[]){
String s1="javaty";
String s2="javaty";
String s3="JAVATY";
String s4="python";
System.out.println(s1.equals(s2));//true because cont
ent and case is same
System.out.println(s1.equals(s3));//false because cas
e is not same
System.out.println(s1.equals(s4));//false because con
tent is not same
}}
```

```
public class JavaObjectgetClassExample {
   public static void main(String[] args)
   {
      Object obj = new String("Java programming");
      Class a = obj.getClass();
      System.out.println("Class of Object obj is : " + a
.getName());
   }
}
```

o/p:-

Class of Object obj is : java.lang.String

Wrapper Class

- Primitive types are not part of object hierarchy and they do not inherit the Object class.
- In some situation these primitive types needs to be work as an object.
  - For example, a method returns object or primitive type value are to be used in a Collection
- As a solution to this problem, Java provides Wrapper classes,
- Wrapper classes allow us to “wrap” primitive data type into an object.
- Wrapper classes are in the lang package.

Primitive Type	**Wrapper classes**
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

- **Creating wrapper class Objects**

  – using there corresponding class constructors

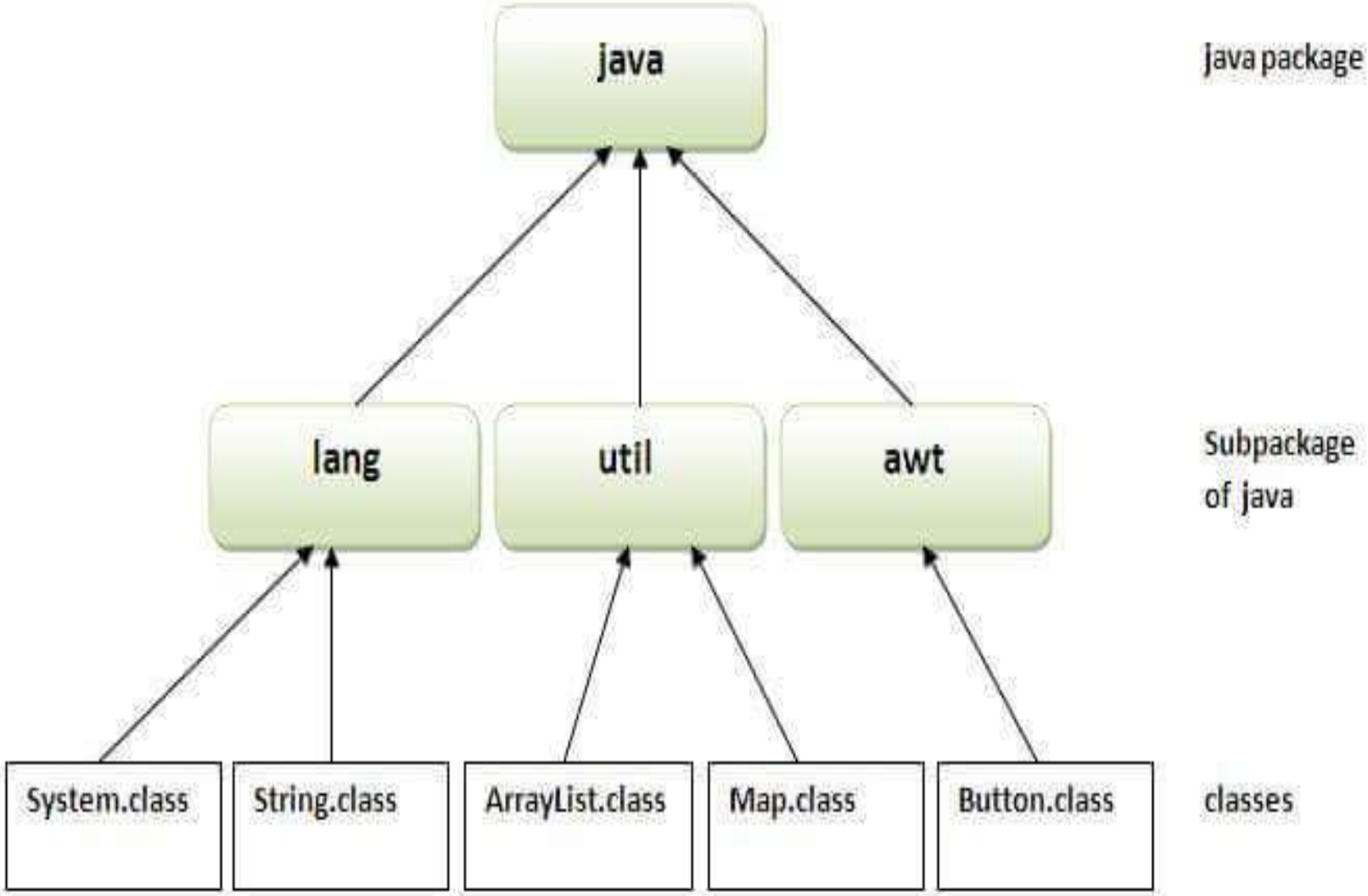
```
int i = 20;

Integer In = new Integer(i);

Float x = new Float(6.5);
```

Introduction to Packages

- Packages contain classes.
- used to keep the class name space compartmentalized.
- The package is both a naming and a visibility control mechanism.
- **package** is a group of similar types of classes, interfaces and sub-packages.
- can be categorized in two form, built-in package and user-defined package.
- many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Advantage of Java Package

- package is used to categorize the classes and interfaces so that they can be easily maintained.
- package provides access protection.
- package removes naming collision


java
Java package
lang
util
awt
Subpackage
of java
System.class
String.class
ArrayList.class
Map.class
Button.class
classes

Predefined Packages

Package	**Purpose**
java.lang	Language related (default package)
java.io	Input Output classes
java.net	Networking and web related classes
java.util	Utility like date etc.
java.awt	AWT library (GUI)
javax.swing	Swing library (GUI)
java.awt.event	Event handling

Garbage Collection

- In C++, dynamically allocated objects must be manually released by use of a **delete** operator.
- Java handles deallocation automatically.
- This technique is called *garbage collection.*
  - when no references to an object exist, that object is assumed to be no longer needed, and the memory occupied by the object can be reclaimed.

The finalize( ) Method

- An object will need to perform some action when it is destroyed.
  - holding some non-Java resource such as a file handle
  - these resources are freed before an object is destroyed.
- Java provides a mechanism called *finalization.*
- Using finalization, you can define specific actions that will occur when an object is just about to be reclaimed by the garbage collector.

- The Java run time calls **finalize()** method whenever it is about to recycle an object of that class.

```
protected void finalize( )
{
   // finalization code here
}
```

Date and time processing

- The java.util.Date class represents date and time in java.
- It provides constructors and methods to deal with date and time in java.

1)	Date()	Creates a date object representing current date and time.
2)	Date(long milliseconds)	Creates a date object for the given milliseconds since January 1, 1970, 00:00:00 GMT.

1	boolean after(Date date)	tests if current date is after the given date.
2	boolean before(Date date)	tests if current date is before the given date.
3	int compareTo(Date date)	compares current date with given date.
4	static Date from(Instant instant)	returns an instance of Date object from Instant date.
5	long getTime()	returns the time represented by this date object.
6	void setTime(long time)	changes the current date and time to given time.
7	Instant toInstant()	converts current date into Instant object.