JAVA CHAPTER 2:- CLASS,INTERFACES AND PACKAGES

```
//2.1 = define class
class point_1{
    public static void main(String[] args) {
        /*
           class:= 1.In object-oriented programming, a class is a
basic building block
                  2.class is a collection of object and it don't take
any space on
                    memory
                  3.class is also called as blueprint

                  4. syntay:=   class class_name {
                   //data
                   //method
                   //code
                  }


                 5.example :=  class Noob{
                   int a=43;
                  }
         */
    }
}
```

Point :-2

```
//2.2= define member fields
```

```
class point_2{
    public static void main(String[] args) {
        /*
            what is member fields- in object oriented programming a
member variable (sometimes called a
                member field)is a variable that is associated with a
specific object and accessible for all its method
                (member function)
                 refer the   chapter 1 point:-2.3-->blocks & variable
scope
         */
    }
}
```

Point:- 3

```
//2.3=constructor & instantion
class point_3{
    public static void main(String[] args) {
        /*
            what is constructor:= 1.constructor is a speciale type of
method whose name is same as class name
                                  2.the main purpose of constructor is
initialize the object
                                  3.every java class has a default
constructor
                                  4.a constructor is outometically
called at the time of object creation
                                  5.a constructor never contain any
return-type including void
```

```
                                6. thre are 4 constructor in java
1.private 2.default 3.parameterized 5.copy
                                7. syntax -  class class_name{
                                                    class_name{


                                                    }

 */
/*
 //default constructor:- program

        class Noob{
             int a; String b; boolean c;
             Noob{
               a=100;
               b="noob";
               c=false;
             }
        }
        void show {
            System.out.println(a+""+b+""+c)
        }
        class Default_constructor{
            public static void main(String args[]){
                Noob n=new Noob();
                n.show();
```

```
            }
        }
*/
/*
//parametrized constructor:- program

        class Noob{
            int o;int z;
             Noob(int a ,int b){
              o=a;
              z=b;
             }
        }
        void show {
            System.out.println(o+""+y)
        }
        class Default_constructor{
            public static void main(String args[]){
                Noob n=new Noob(45,79);
                n.show();
            }
        }
*/
 /*
   // copy constructor:- program
        //not IMP
```

```
         */
        /*
            // private constructor  :- program
                //not IMP
         */



    }
}
```

Point:- 4

```
//2.4=Defining and Invoking Member Methods
class point_4{
      public static void main(String[] args) {
        /*
         Method :=
            1.In general, a method is a way to perform some task
            2.Similarly, the method in Java is a collection of
instructions that performs a specific task.
            3.It provides the reusability of code. We can also easily
modify code using methods.
            4.There are two types of methods in Java: Predefined
Method,User-defined Method
            5.Predefined Method-In Java, predefined methods are the
method that is already defined in the
             Java class libraries is known as predefined methods. It
is also known as the standard library
             method or built-in method
```

```
            example -println();

            6.User-defined-The method written by the user or
programmer is known as a user-defined method.

            These methods are modified according to the requirement.

            example -void show(){};



         */

        /*

            How do you invoke a method?

                1.Create a class object that corresponds to the
object whose method you want to invoke.

                 For more information, see the Retrieving Class
Objects section.

                2.Create a method. object by calling getMethod on the
class object.

                3.Call the method by calling invoke .

        */

        /*




        */
}
}
```

Point:-5

```
//2.5= Inheriting Members from Another Class (superclass)
class point_5{
```

```
    public static void main(String[] args) {
        /*
         Inheritance:-1.Inheritance in Java is a mechanism in which one object acquires all the properties
                       and behaviors of a parent object. It is an important part of OOPs
                     2.a new class access all the feature & properties of existing class called inheritance
                     3.in java extends keyword is use to perform inheritans
                     4.it provid code reusablity
                     5.we can't access private member of class through inheritance
                     6.types of inheritance Single,Multilevel,Hierarchical,multiple
                     7.syntax:=     class Subclass-name extends Superclass-name{
                                        //methods and fields
                                    }
         */
        /*
         Single inheritance program:=
          class Animal{
            void eat(){
                System.out.println("eating...");
           }
            }
            class Dog extends Animal{
            void bark(){
```

```
        System.out.println("barking...");
    }
    }
    class TestInheritance{
    public static void main(String args[]){
    Dog d=new Dog();
     d.bark();
     d.eat();
 }}
 */

/*
// Multilevel Inheritance program:=
 class Animal{
    void eat(){
        System.out.println("eating...");}
    }
    class Dog extends Animal{
    void bark(){
        System.out.println("barking...");}
    }
    class BabyDog extends Dog{
    void weep(){
        System.out.println("weeping...");}
    }
    class TestInheritance2{
```

```
    public static void main(String args[]){
    BabyDog d=new BabyDog();
     d.weep();
     d.bark();
     d.eat();
     }}
 */
/*
 Hierarchical Inheritance program:=
 class Animal{
        void eat(){
            System.out.println("eating...");}
        }
        class Dog extends Animal{
        void bark(){
            System.out.println("barking...");}
        }
        class Cat extends Animal{
            void meow(){
                System.out.println("meowing...");}
        }
        class TestInheritance3{
        public static void main(String args[]){
        Cat c=new Cat();
         c.meow();
         c.eat();
```

```
                }}
         */
        /*
         super  keyword in java:-
                1.super keyword refer to the object of super class it
is used when we want to call the supwe class
                 variable ,method,& constructor through sub class
object
                2.whenever the super class & sub class variable and
method name both are same than it can be used
                 only
                3.to avoid the confusion between super class and sub
classes variable and method  that have same
                 name we should use super keyword

                super program:=


                class A{
                   int a=10;
                }
                class B Extends A{
                   int a=20;
                   void show(){
                       System.out.println(a);
                       System.out.println(super.a);
                   }
                }
```

```
                class Main_A{
                    public static void main(String argd[]){
                        B obj = new B();
                        obj.show();
                    }
                }
        */

    }
}
```

Point:-6

```
//2.6 = Defining and Implementing Interfaces
class point_6{
    public static void main(String[] args) {
        /*
         Interfaces:=1.Interfaces is just like a class which contain
only abstract method
                    2.to provide Interfaces java provide a keyword
called implement
                    3.Interfaces methods are bydefault public &
abstract
                    4.Interfaces method must be overriden inside the
implimenting classes
                    5.Interfaces nothing but deals between client &
developer
                    6. syntax :- Interfaces Interfaces_name{
```

```
   //code
}
```

7. Interface program:=

```
inport java.util.Scanner;
      Interface client{
       void input();
       void output();
      }
      class test implements client{
       String name;double sal;
       public void input(){
           Scanner sc = Scanner (System.in);
           System.out.println("Enter the name");
           name=sc.nextLine();
           System.out.println("Enter the
salary");
           name=sc.nextDouble();
       }
       public void output(){
           System.out.println(name+""+salary)
       }
       client c = new test();
       c.input();
       c.output();
      }
```

```
        */


    }
}
```

Point:-7

```
//2.7= Overriding and Overloading methods
class point_7{
    public static void main(String[] args) {
     /*
      method overriding -1.If a subclass provides the specific
implementation of the method that has been
                          declared by one of its parent class, it is
known as method overriding.
                        2.Method overriding is used to provide the
specific implementation of a method which
                          is already provided by its superclass
                        3.Method overriding is used for runtime
polymorphism
                        4.syntax -
                            class A{
                             void show(){


                             }
                            }
                            class B Extends A{
                             void show(){
```

```
    }
   }
```

```
5.program
class A{
 void shape(){
   System.out.println("Shape A ");
 }
}
class B Extends A{
 @Override
 void shape(){
 System.out.println("Shape B");
}
class test{
 public static void main(String args[]){
   A s = new A();
   s.shape();
 }
}
```

```
*/
/*
```

Overloading methods:=1.whenever a class contain more then one method with same name and different type

of parameters called method overloading

2.syntax:=

return_type methode_name(int a);

return_type methode_name(int a,int b);

3.program -

```
class A{
 void add(){
   int a=14,b=34,c;
   c=a+b;
   System.out.println(c);
 }
  void add(int x,int y){
   int c;
   c=a-b;
   System.out.println(c);
 }
  void add(double c,int y){
   double c;
   c=a-b;
   System.out.println(c);
 }
 public static void main(String
args[]){
   A r = new A();
   r.add();
   r.add(28,87);
   r.add(45.56,67);
```

```
                                        }
                                    }
        */
    }
}
```

Point:-8

2.8= Static Fields and Static Methods

Note :- 1.A static field can be accessed before any object of a class is created, without reference to any object.

2. We define as static when the behaviour or state does not depend on any particular instance of the class.

1|static variable=[

- Static variables are also known as class variables.
- Initialization of a static variable is not mandatory. Its default value is 0.
- static variables are declared using the static keyword within a class outside of any method, constructor or block.
- class main{

  public static int a=6; --> static variable

  public static void main(String args[]){

  }

]

2|static method=[

- static method access only static variable
- without object you can access static method
- program:-

  class A{

    static int a=34;

    public static void main String(String args []){

```
        A.show();
      }
      static void show(){
        System.out.println(a);
      }
    }
]
3|static block=[
    ●static block is such kind of block in java which gets executed at the time of loading the
    class file into JVM
    ●syntax- class A{
      static {
        //static block
      }
    }
    ●program-
      class A{
        public state void main(String args[]){

        }
        static{
          System.out.println("NOOB");
        }
      }

]

Point:-9
```

2.9= Defining a Package

Package[

1|A java package is a group of similar types of classes, interfaces and sub-packages.

2|Package in java can be categorized in two form, built-in package and user-defined package.

3|There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

4| syntax of creating user_defined package Package Package_name;

5|how to compile package file

use this command:- javac -d . file_name

6|example :- javac -d . Simple.java

Point:-10

2.11 Access Control

1|Private: The access level of a private modifier is only within the class. It cannot be accessed from outside

the class.

2|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.

3|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.

4|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.