(In java their are 68 reserve keywords)

Setters and Getters in Java ⇒

i) In Object Oriented programming we should not access the data (i.e Read, write, ~~access~~ edit) we should always access it by some behaviour.

ii) So to mutated (set / change) the data. we use setters and to read the data we use getters (accessor)

(right click → sourse → generate setters/getters) (In Eclipse)

Constructor ⇒

If we want to initialize the object with some default values at the time of that object creation then we use constructor. A constructor never contain any return type.

Defination →

It is a special function which has name same as that of class and gets call implicitly at the time of object creation

Their are 4 types of Constructor ⇒

i) Private Constructor
ii) Default Constructor
iii) Parameterize Constructor.
iv) Copy Constructor.

(ii) Default Constructor ⟹

Default constructor takes no any parameter and set default values.

Example →

```
public employee ()
{
    System.out.println("Default Constructor");
    this.id = 0;
    this.name = "Not Given";
    this.salary = 0;
}
```

In main

```
employee e1 = new employee ();
```
No parameter

(iii) Parameterize Constructor ⟹

When we want to initialize a object by passing some values at the time of object creation
(1 or more than 1 parameter)

Example →

```
public employee (int id, String name, int salary)
{
    this.id = id;
    this.name = name;
    this.salary = name;
}
```

① Private Constructor ⇒

In java, it is possible to write a constructor as a private but according to the rule we can't access private members outside of class.

Syntax ⇒

```
class class_name
{
    private class_name()
    {

    }
}
```

Example ⇒

(i)

```
Class A
{
  int a;
  String name;

  private A()
  {
    a = 10;
    name = "Ankit";
  }print(a,name);
  public static void main
      (String[] args){
  } A obj = new A();
```

Output ⇒
10
Ankit.

(ii)

```
Class A
{
    int a;
    String name;

    private A()
    {
      a = 10;
      name = "Ankit";
    } print(a,name);
}
Class B
{ public static void main
      (String[] args){
  } A obj = new A();
```

Output ⇒
error
A() has private access in A.

④ Copy Constructor ⇒

Whenever we pass object refference to the constructor then it is called copy constructor.

Syntax →

```
class class_name
{
    class_name (obj_ref)
    {
    }
}
```

(copy all the content from one constructor to another)

Example →

```
Class A
{
    int a;
    String b;

    A()
    {
    a=10;  b="Ankit";
    print (a,b);
    }
    A (A ref)
    {
     a = ref.a;
     b = ref.b;
    print (a,b);
    }
}
```

```
Class B
{
    public static void main
            (String[] arg) {

     A r1 = new A();
     A r2 = new A(r1);
    }
```

In main

```
employee e2 = new employee(102, "Ankit", 2000);
```

parameter
(At least 1 parameter required)

This keyword in java :→

i) The this keyword refer to the current object in a method or constructor

ii) The most common use of the this keyword is to eliminate the confusion between class attributes and parameters with the same name

Example →

```
public class Point
{
    private int x;
    private int y;

    public Point (int x, int y)
    {
        this.x = x;
    // this.x refers to instance variable
       while x refers to the parameter

        this.y = y;
    }// similarly here.
}
```

* What is Components of Java Architechure.

① JVM (Java Virtual Machine) :→

The main feature of java is "Write Once Run Anywhere" (platform independent). Because of JVM.

It is Java platform that gives us an environment to execute java program.
(JVM main task is to convert byte code into machine code)

② JRE (Java Runtime Environment) :→

It provides an environment in which Java programs are excute

The JRE contains libraries and software needed by your Java programs to run.
(JRE takes our Java code, integrates it with the required libraries, and then starts the JVM to execute it).

③ JDK (Java Development Kit) :→

JDK is a software development environment used to develop Java applications and applets.
(It contains JRE & several development tools an interpreter/loader (java), a compiler (javac), an archiever (jar), etc.).

* Input and Output in Java :→

① Input (Scanner Class) :→

Scanner is a predefined class in Java which is available in java.util. package

It is used to get user input.

Rules ⇒

i) If we use Scanner class, must have to create object of Scanner class.

```
Scanner object_name = new Scanner(System.in);
```

ii) Scanner class methods :→

```
① nextLine();      // string with space
② nextInt();       // Integer
③ nextFloat();     // floating
④ nextBoolean();   // True or false
⑤ nextDouble();    // Double
⑥ next();          // string
⑦ next().charAt(0); // character at index 0.
```

iii) import Scanner class package at the top line of program.

```
import java.util.Scanner;
```

iv) Wrong Input ⇒ If use method nextInt() and my ~~ta~~ eh take input string than it will show Input Mismatch Exception.

② Output ⇒ (System.out.print())

It is an output statement in java through which we can print the variables, expressions and many more content.

What is System →

i) Predefined class
ii) It is available in java.lang package (heart of java)

What is out →

Out is static member of System class

What is print →

It is predefined method used to print statement on console.

* Note ⇒

(i) In c language if(1) is allowed
{
}

because in c integer value is always consider as true.

(ii) But in Java we used to write boolean value in condition making statements.
i.e if(true)
{
}

(In Java their is <u>no Destructor</u> like C++.
Instead of that Java has an efficient Garbage collector that deallocates memory automatically.)

Constructor chaining ⇒

When we create a class which inherited by a superclass (In this when we create an object of derived class the Constructor of super class also called with derived class constructor)

Polymorphism ⇒ (Same object having diff behaviour)

① Compile time polymorphism :→
(Static Binding / Method Overloading)

A polymorphism which is exists at the time of compilation is called Compile time polymorphism or early binding or static polymorphism

we acheive it by method Overloading

Whenever a class contain more than one method with same name and different types of parameter it is called as Method Overloading.

Syntax →

return_type nameoffunction(Para){}

return_type nameoffunction(Para1, Para2){}

Example →

```
Class A
{
void add()          */  int add()
{                       {
  int a=10, b=20, c;      int a=10, b=20, c;
  c=a+b;                  c=a+b;
  print(c);               return c;
}                       }
                        /*
```

```
void add(int x, int y)
{
    int c;
    c = x+y;
    print(c);
}
void add(int x, double y)
{
    double c;
    c = x+y;
    print(c);
}

public static void main(String[] args){

    A r = new A();

    r.add();                    */ int add=r.add();
    r.add(100,200);                 print(add);
    r.add(50, 45.32);               /*
}
```

Output ⇒

30
300
95.32

② Runtime polymorphism :→
(Dynamic Binding) / Method Overriding)

override method of its subclass at runtime

* Package ⇒

A package arrange number of classes, interfaces and sub package of same type into a particular group.

Package is nothing but folder in window

Types :→

- Pre-defined
- User-defined

Pre-defined	User-defined
java.lang (by defaut)	Package demo
java.util (data structure)	Package P1
java.io (input output)	Package add
java.applet	Package myPack
java.awt (web related application)	
java.net (web related application)	
java.SQL (Database)	

* Access Modifier ⇒

	Access Modifier	within class	within package	Outside Package by Subclass	Outside package
①	Private	✓	✗	✗	✗
②	Default	✓	✓	✗	✗
③	Protected	✓	✓	✓	✗
④	Public	✓	✓	✓	✓

private cannot be accessed outside class
default cannot access by other package
Protected can be accessed by class & subclass but not access by outside package subclasses
Public can be access every where

Advantage of Package ⇒

i) Reusability
ii) Security
iii) Fast searching of packages
iv) Naming conflecting problem solved (Package 1 class A, Package 2 class A) Here in two diff. packages their is same name of class using pacg no naming conflect happens
v) Hiding (using package we can hide our source code and only share package by hiding source code)

Disadvantage of Package ⇒

We cannot pass parameter to package.

* System.out.println("Hello");

i) Here System is class from java.lang package
ii) . is a operator (member access operator)
iii) out is a object (static variable of PrintStream class)
iv) print is a method & ln is for next line.
(method inside PrintStream class)

* Types of variable in Java ⇒

Three types ⇒

variable

- local
- static (globle) {in java their is no globle variable static variable is treated as globle variable}
- Instance

① Local variable :→

A variable which is declared inside the body of the method or method parameter is called local variable.

Syntax :→

i) void fun ( )

```
{
    int x;    // local variable
}
```

ii) void fun (int a)

{ ← local variable.

}

what is property in java class. →

int[] arr = new int[10];

As we know when we use array to find its length we use arr.length, so here length is not a method it is variable ~~int~~ a using setter/getter it gets value.

② Instance variable :→

A variable which is declared inside the class but outside of all the methods is called instance variable.

Syntax :→

```
class A
{
    int a ;   // instance variable.
    public static void main (String[] arg)
    {

    }
}
```

③ Static variable :→

A variable which is declared with the help of static keyword called static variable.

Syntax :→

```
static int x ;
```

Note ⇒

i) Use for memory management

ii) Generate single copy storage for variable & method.

* Lets understand with code

** How to access Local variable, Static variable & Instance variable ⟹

```
Class A
{
  static int b=20 ; //static variable

  int c=30 ;        // instance variable

  public static void main(string[] args)
  {
     int a=10;      // local variable

     A ref = new A();

     System.out.println(a); (print it locally)
     System.out.println(b);
     System.out.println(A.b);
     System.out.println(ref.b);
     System.out.println(ref.c);
  }
```

It can be also access without class Name & Class object

static variable can be access by class name & class refference (object)
i.e A.b (with class)
ref.b (with object)

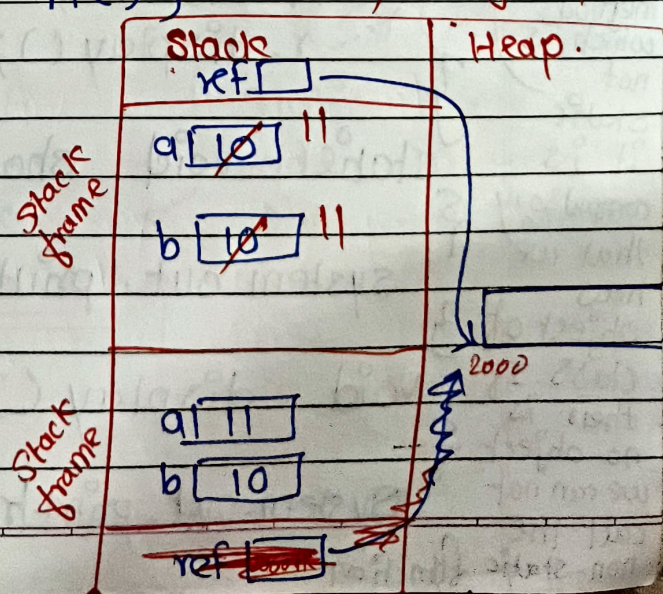
Instance variable can only access by refference of class i.e object of class

* Example of How static variable forms single copy & local variable form multiple copy.

```
Class A
{
    static int a =10;

    void fun()
    {
        int b=10;

        System.out.println (a+" "+b);

        ++a ;
        ++b ;
    }

    public static void main (String arg[])
    {
        A ref = new A();

        ref.fun();

        ref.fun();
    }
```

It is accessed without using class name & class object (ref)

Memory diagram



* Difference between static & non static Method ⇒

i) If we apply static keyword with any method it will be treated as static method. (static void Add ())

ii) While non-static method does not require any keyword (void Add ())

Lets understand with code →

```
Class A
{
    int a=10;
    static int b=20;

    public static void main (String [] args)
    {
        A r =new A();
        A.show();
        r.display();
    }
    static void show()
    {
        system.out.println("showing "+b);
    }
    void display()
    {
        System.out.println("Displaying "+a+" "+b);
```

but to call display method which is not static it is compulsory that we have object of class. If their is no object we can not call the non-static function

show method can be called with class name and without class name .i.e show();

Instance variable

Here we didnot access 'a' because static method can only access static variable

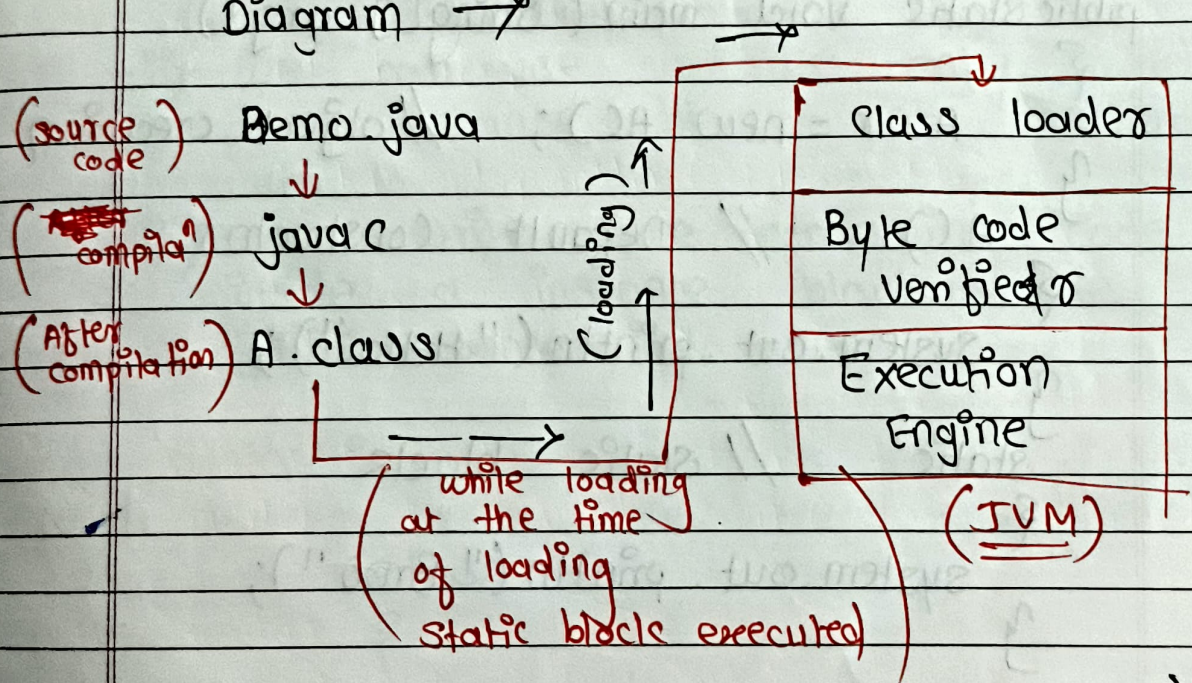
* Static block ⟹

Static block is a group of statement that gets executed when the class is loaded into memory by Java class loader.

OR

executed at the time of loading the .class file into JVM memory

Diagram →



Syntax ⟹

```
Class A
{
    static
    {
        ____
    }
}
```

*** (IMP)

Static block code is executed only once when class is loaded into memory before main method or any object creation

Static block is always get executed first because they get store into the memory at the time of class loading or before object creation.

Example →

```
Class A
{

public static void main ( String[] args)
{
    A r = new A();   // object creation
}
  A()   // Default Constructor
  {
    System.out.println("Hello");
  }
  static   // static block
  {
    system.out.println("Broo");
  }

}
```

Output ⇒

Broo
Hello.

static block alway execute first it doesn't matter where we have written it in code.

Instance block ⇒

Instance block is similar to method which has no name, it can be written inside a class but not inside a method.

Note :→

i) It always executed before the constructor

ii) We can use variable only inside the instance block not method

iii) We write time consuming code inside a instance block like JDBC connectivity.

Syntax :→

```
Class A
{
    {                    // Instance
                            block
    ————
    }
}
```

Output →

10 20
30 40

Example →

```
Class A
{
    int a,b;

    A()          // Default Constructor
    {
        a=30; b=40;

        System.out.println(a+" "+b);
    }

    {                          // Instance block
        a=310; b=20;

        System.out.println(a+" "+b);
    }
}

Class B

{
public static void main (String[] args)
{
        A r = new A();
}
```

After creating object of A class first Instance block executed then the defaut Constructor execute

** But with the help of static block & static method we can execute it where we want

* Lets understand with code

```
Class A
{
  int a,b;

  static void show()          // static method
  {
    system.out.println("Hello");
  }
  A()                         // Default constructor
  {
    System.out.println("Bro");
  }
  {                           // Instance block.
    System.out.println("welcome");
  }
}
Class B
{
  public static void main(String args[])
  {

    A.show();
    A r = new A();
  }
```

output →
Hello
welcome
Bro

* Why ?. Public static void main(String arg[])

In the starting of Java program, JVM calls the main() method

Public :→ In Java Public is a access modifier
Here main method is public because from outside of class we can access main method (i.e JVM can access main method outside the class)

Static :→

Here main method is static because JVM ~~we~~ can access the main method without creating object of main class.
methods

(when JVM calls main method it can be called without object creation)

Void :→

Void is Datatype which returns no value.

String :→
String is predefined class in Java
It is array of Strings, main method passes this argument so that we can send any information