Introduction to Java:

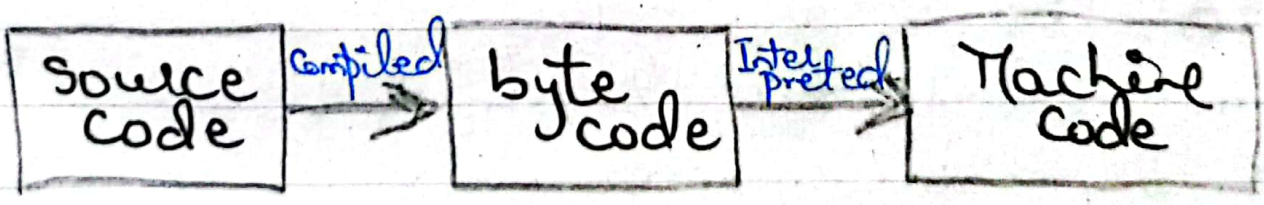
Java was developed by James Gosling at Sun Microsystems in the year 1991. It is a simple programming language. It makes writing, compiling, and debugging programming easy. It helps to create reusable code and modular programs.

It is class based, object oriented programming language. It's initial name was OAK.

Do

How Java Works:-

It is compiled into the byte code and then it is interpreted to machine code.



byte code:

It is the instruction set for the Java Virtual Machine. It acts similar to an assembler.

Scanned with CamScanner

- JDK → stands for Java Development Kit.
- Collection of tools used for developing and running Java programs.

- JRE → stands for Java Run Time Environment
- Helps in executing programs developed in Java.

2 Basic Structure:

package com.company:-

- A package in Java is used to group related classes. Think of it as a folder in a file directory. Encapsulate a group of classes.

main():

The main method is the entry point into the application

run without creating obj

entry point

```
public static void main(){
}
```

access modifier

no return value

Methods :-

- block of code which only runs when it is called.
- You can pass data, known as parameters, into a method.
- used to perform certain actions.
- It defines the code once, and use it many times.
- We use camelCase Convention.

Class :-

- basic building block.
- A logic template to create the objects that share common properties and methods.
- We use Pascal Convention.

3) Variables and Datatypes :

Data Types

- Primitive
  - int
  - long
  - double
  - short
  - byte
  - float
  - char
  - bool
- Non Primitive
  - class
    - string
    - array

Scanned with CamScanner

Variable:

A variable is a container that stores a value.

```
int num = 8;
```

- Variable name is case sensitive. num and Num both are diff.

Primitive Data Type:-

Java is statically typed means variables must be declared before use. 8 primitive data types define below:

- **byte** →
  - Value ranges from -128 to 127.
  - takes 1 byte
  - default value is 0.

- **short** →
  - Value ranges from $-(2^{16})/2$ to $2^{16}/2 - 1$
  - takes 2 bytes
  - default value is 0.

- **int** →
  - Value ranges from $-(2^{32})/2$ to $2^{32}/2 - 1$
  - takes 4 bytes
  - default value is 0.

Scanned with CamScanner

- float → • value ranges from decimal.
  - takes 4 bytes
  - Default value is 0.0f

- long → • Value ranges from $-2^{64}/2$ to $2^{64}/2 - 1$
  - takes 8 bytes
  - default value is 0.

- double → • value contains decimal nos.
  - takes 8 bytes.
  - default value is 0.0d.

- char → • Value ranges from 0 to 65535 ($2^{16}-1$)
  - takes 2 bytes bcz it supports unicode
  - default value is 0.

- boolean → • Value can be true or false.
  - Size depends on JVM..
  - default value is false.

JVM → Java Virtual Machine.

Scanned with CamScanner

(4) Literals in Java:

- A constant value which can be assigned to the variable is called as a literal.

```
byte a = 3;
char ch = 'A';
float f1 = 5.6f;
double d1 = 4.6d;
int b = 2;
Long k = 566666L;
Short s = 56;
String str = "faizeen";
```

(5) Taking User Inputs-

- In order to read data from the Keyboard Java has a Scanner class.
Scanner class has a lot of methods to read data from the Keyboard.

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

s → obj

System.in → Read from the Keyboard

Scanned with CamScanner

```
int a = sc.nextInt();
```

sc → obj of Scanner class

nextInt() → method for taking input value.

```
boolean b1 = sc.hasNextInt();
```

hasNext → to check input data according to the data type.

```
String str = sc.next();
```

→ just ek word ko input lene k liye.

```
String str2 = sc.nextLine();
```

→ poora sentence ko input lelega.

6 Calculate Total Marks and Percentage:-

```
package com.company;

import java.util.Scanner;

public class exercise_06{
```

Scanned with CamScanner

```
public static void main(String[] args)
{
System.out.println("Enter Your Subjects
                    Marks:");

Scanner sc = new Scanner(System.in);
float total_marks = 500;

System.out.println("Enter Your 5 sub
                    marks:");
float sub1 = sc.nextFloat();
"    "  2 =  "    "
"    "  3 =  "    "
"    "  4 =  "    "
"    "  5 =  "    "

float om = sub1 + Sub2 + Sub3 + Sub4 + Sub5;
System.out.println(om);

float percentage = (om/total marks) x 100;

System.out.println(percentage);
}
}
```

Scanned with CamScanner

7) Operators:-

Operators are used to perform operations on variables and values.

7 + 7 = 14

operands ← operator, result

Types of operators:

- Arithmetic: +, -, *, /, % (return decimal remainder as well), ++, --
- Assignment: =, +=, -=
- Comparison: ==, >=, <= → return true or false
- Logical: &&, ||, !
- Bitwise: &, |

8) Precedence:

The operators are applied and evaluated based on precedence. for example (+, -) has less precedence compared to (*, /). Hence + & |

Associativity of * and / left to right.

+ - Left to right.

++, = Right to left.

Scanned with CamScanner

(10)

Data Types:

Data Types specify the different sizes and values that can be stored in the variable.
following table summarizes the resulting data types after arithmetic operation on them.

b -> byte
s -> short
i -> int
f -> float
l -> long
c -> char
d -> double

R = b+s -> int
R = s+i -> int
R = l+f -> float
R = i+f -> float
R = c+i -> int
R = c+s -> int
R = l+d -> double
R = f+d -> double

Increment and Decrement:

a++ , ++a -> Increment
a-- , --a -> Decrement

These will operate on all data types except boolean.

Scanned with CamScanner

a++ → first use the value and then increment

++a → first increment the value then use it.

```
int b = 5;
System.out.println(++b); → 6
    "   "      (b++); → 6 then increment
```

```
char ch = 'a';
"   "   "   (++ch); → b
```

(13) Strings:

'A string is a sequence of characters'.
A string is initiated as follows:

```
String name;
name = new String("farzeen");
```

String is a class but can be used like a datatype.
Strings are immutable means can't be changed.

Scanned with CamScanner

%.2f → use for 2 decimal places.

String name = "farzeen";

System.out.printf("Value of a %d", a);

%d → int
%c → char
%f → float
%s → string.

System.out.format(); → same as printf.

(14) String Methods:

String Methods operate on Java Strings. They can be use to find length of the string, convert to lowercase, etc. Some of the commonly used String methods are:

1. name.length() → Returns the length of string name.

2. name.toLowerCase() → Returns the new string which has all the lowercase characters from the string name.

Scanned with CamScanner

3. <u>name.toUpperCase()</u> → Returns a new String which has all the lower case characters from the string name.

4. <u>name.trim()</u> → Returns a new String after removing all the leading and trailing spaces from the original String.

5. <u>name.substring()</u> → Returns a Substring from start to the end.
starting index hum as a parameter pass kate hain wahan se end tk print kedeta.

. <u>name.substring(startIndex, endIndex)</u> → Returns a substring, starting is included and the ending is excluded.

e.g:

Farzeen
0 1 2 3 4 5 6

(1 , 5) → the output will be 'arze'
↓ start Index
↓ end Index

Scanned with CamScanner

website.endsWith(".com"); → to check website ends.

7. name.replace('r', 'p') → Returns a new string after replacing 'r' with 'p'.

(char, char)

also we replace strings like this
("air", "een");
target, replacement

8. name.startsWith("Fa") → returns true if string starts with "fa".

9. name.endsWith("en") → return true if string ends with "en".

10. name.charAt(2) → returns character at a given index position.

11. name.indexOf(s) → returns the index of the given substring.
(substr)

e.g:-
name.indexof("ai") returns 1 which is the first occurence of ai in string Faizeen.
if no match is found it returns -1.

Scanned with CamScanner

2. name.lastIndexOf("r") → returns the last index of the given string.

3. name.lastIndexOf("r", 2) → returns the last index of the given string before index 2.

4. name.equals("faizeen") → returns true if the given string is equal to "faizeen" otherwise false.

5. name.equalsIgnoreCase("faizeen") → returns true if the given string is equal to faizeen. Case is lower or upper it doesnot matter in this method.

6. **Conditionals:**

In Java, conditional operators check the condition and decides the desired result on the basis of both conditions.

* If else Conditionals
* Switch Statement

Scanned with CamScanner

• If Else Statement

Java has a Simillar syntax like C and

```
if (condition to be checked)
  {
    statement - if condition true.
  }
else → if above if condition is false
  {
    statement
  }
```

• else is optional.

Relational and Logical Operators:

Relational operators are used to evalu conditions (true or false) inside the if statements.

e.g

== , >= , > , < , <= , !=

Sh

if no

"=" is used for assignment, where as "==" is used for equality check.

Logical Operators:-

&&, || and ! are most commonly used logical Operators in Java.

&& → AND
|| → OR
! → NOT

} used to provide logical.

• AND:

1 1 → 1
0 1 → 0
1 0 → 0
0 0 → 0

• OR

1 1 → 1
0 1 → 1
1 0 → 1
0 0 → 0

• NOT:

1 → 0
0 → 1

if a = true
then !a = false

(18) Switch Case :

Switch Case is used when we have to make a choice b/w number of alternatives for a given variable.

```
switch (val)
{
    case c1 :
        //code
        break;
    case c2 :
        // code
        break;
    default :
        // code
}
```

'val' can be int char or string.

Scanned with CamScanner

11 Loops:

Loop is a control flow statement that iterates a ~~program~~ part of the program multiple times.

- While Loops
- Do-while loops
- for Loops

- While loops:

```
While(boolean condition)
{
    //statement → this keeps
                executing as long
                as the condition is true.
}
```

2) do-While loop:

this loop is similar to a while loop except the fact that it is guaranteed to execute at least once.

```
do{
    //code
}while(condition);
```

Scanned with CamScanner

2n → even nos
2n+1 → odd nos

3) for Loops :-

The syntax of a for loop looks like

```
for(initialization, check expression, update)
{
    // code;
}
```

A for loop is usually used to execute a piece of code for specific number of times.

24) Break Statements

It is used to exit the loop irrespective of whether the condition is true or false.

```
break;
```

Continue Statements

It is used to continue the loop.

```
continue;
```

Scanned with CamScanner

26) Arrays in Java :-

↳ array is an object in Java.

Arrays are used to store multiple values in a single variable having same data type.

Accessing ~~are~~ elements in an array is faster.

Syntax:

- int [ ] marks; → Declaration
  marks = new int [5]; → Memory Allocation

- int[] marks = new int [5]; → Declaration + Memory Allocation

int [] marks = {10, 9, 7, 5, 1}; → Declare + initialize

Array indices starts from 0 and goes till (n-1) where n is the size of an array.

float[] marks = {98.5f, 72.2f, 87.5f};

String [ ] std = {"fauzeen", "Ali"};

Scanned with CamScanner

(27) for-each loop in Java

It is mainly used to traverse the array or collection ~~in Java~~ element. The advantage of the foreach loop is that it eliminates the possibility of bugs and makes the code more readable.

Syntax:

```
for(int element : marks)
  {                  ↳ array name
    Sout(element);
  }

}
```

Scanned with CamScanner

don't declare size at the time of declaration.

Array length:

Array have a length property which gives the length of the array.

```
marks.length;
```

Displaying an Array:

```
for(int i= 0; i<marks.length; i++)
{
System.out.println(marks[i]);
}
```

This is also known as array traversal.

For displaying in reverse:

```
for (int i= marks.length-1; i>=0; i--)
{
  Sout (marks[i]);
}
```

index

Note: Arrays are object in java and its super class is Object class. Its memory is located in Heap.

Scanned with CamScanner

(28) Multidimensional Array:

Multidimensional Arrays are Array of Arrays. Each element of a multi dimensional array is a array itself.

• 2-D Array:- → 2 loops using!

A 2-D Array can be created as follows:

```
int[][] floater = new int[2][3]
```

row column

We can add elements to this array as follows

```
floater [0] [0]  =  100
   "    [0] [1]  =  200
   "    [0] [2]  =  300
```

and so no!

This 2-D Array can be visualized as:

	Col 1	Col 2	Col 3
[0] Row 1	(0,0)	(0,1)	(0,2)
[1] Row 2	(1,0)	(1,1)	(1,2)

Scanned with CamScanner

31 # Java Methods:- → a func inside a class.

A method is a function written inside a class. Since Java is an Object Oriented Language, we need to write the method inside the same class.

→ without static error as we call it from main method but static here.

```
static int logic(int x, int y)
{
    " " code }

logic(a,b);
```

→ calling function.

Calling A Method:-

A method can be called by creating an object of the class in which the method exist followed by the method call:

```
calc obj = new Calc();
    obj.mySum(a,b);
```

→ obj creation.

→ Method call upon an obj

```
    int (obj);
```

object method using obj creation:

If we do not make a method static then we can create its own obj and then we will call the method like this:

```
int logic(int x, int y).
{

}
public static void main()
{
Methods obj = new Methods();
```

class name obj class obj creation

```
c = obj.logic();
```

calling a function.

(32) Method Overloading in Java

• Void return Type:

When we donot want our method to return anything. We use void as a return type.

Scanned with CamScanner

Static Keyword:-

It is used to associate a method of a given class with the class rather than the object. Static method in the class is shared by all the objects.

note:-

Incase of arrays, the reference is passed. Same is the case for object passing to methods.

```
main(){
 int [] marks = { 1, 2, 3, 4};
 change (marks);
 sout( marks [0]);
}

        static void change(int[] a){
        a[0] = 50;
        }
```

the value of array index 0 is 50.

Scanned with CamScanner

Method Overloading:

- Two or more methods can have same name but different parameters. Such methods are called overloaded methods:

```
void foo();
void foo(int a);
int foo(int x, int y);
```

⇒ overloaded func foo.

Method overloading cannot be performed by changing the return type of methods.

e.g:

```
void foo(int a)
{
}

int foo(int a)
{
}
```

error

```
void foo(int a)
{
}

int foo(int a, int b)
{
}
```

allow

33

Variable Arguments:

The variable allows the method to accept zero or multiple arguments. Before varags either we use overloaded method or take an array as the method parameter but it was not considered good because it leads to the maintenance problem. if we dont know how many arguments we will have to pass in the method varags is better approach.

```
return type methodname(datatype... variableName)
{
    {

static int sum(int... arr)
{ int result=0;          //Available as int[] arr;
  for(int a : arr)
  {
    result += a;
  }
  return result;
}
```

Scanned with CamScanner

(34) **Recursion:**

A method in java thats calls itself is called Recursive Method.

for Example:

Factorial of a number

factorial(n) = n * factorial(n-1)

Syntax:

```
returntype methodname()
{
    methodname();
}
```

A method calls itself directly is known as direct Recursion.

A method calls another function that in turns it calls it.

Scanned with CamScanner

35

Introduction to OOP:

Solving a problem by creating objects is one of the most popular approaches in programming. This is called OOP.

Class:

A class is a blueprint for creating objects. It contains information to create a valid object.

Object:

An object is an instantiation of a class. When a class is defined, a template(info) is defined. Memory is allocated after object instantiation.

How to model a problem in OOPs:

We identify the following:

Noun → Class → Employee

Adjective → Attributes → name, age, s.no

Verb → Methods → getSalary();

Scanned with CamScanner

3) OOPs Terminologies

1. Abstraction

Hiding internal details

e.g:-

using the phone without bothering about how it was made.

2. Encapsulation

The act of pulling various components together (in a capsule).

e.g:-

Laptop is a single entity with wifi + speaker + storage in a single box.

In Java, encapsulation simply means that the sensitive data can be hidden from the users.

3. Inheritance

The act of deriving new things from existing things.

e.g:-

Rickshaw → E-Rickshaw

Phone → Smart Phone

Scanned with CamScanner

1. **Polymorphism:**

One entity many forms.

Smartphone = Phone + Calculator.

Custom Class:

We can write a custom class as follows:

```
public class Employee {
    int id;          → Attribute 1
    String name;     → Attribute 2
}
```

Any real world object = Properties + Behaviour

Object in OOPs = Attributes + Method.

note:-

ek file me ek public class hoty hi agr ek file me do classes banny hain to ek public or doosry just class or class ka naam likhden aese:

```
class employee{
}
```

Scanned with CamScanner

Class with Methods:

1. We can add Methods to our class Employee as follows:

```
public class Employee
{
    public int id;
    public string name;

    public int getSalary( )
    { // code
    }

    public void getDetails( )
    { // code
    }
}
```

Farzeen

Scanned with CamScanner

(40)

Access Modifiers:-

Specifies where a property/method is accessible.

Four types of access modifiers are:

- Private
- Default
- Protected
- Public

Getters and Setters:-

Getters and Setters are methods use to protect your data and make your code more secure. Getters returns the value (accessors). it returns the value of data type int, String, double, Float, etc.

Setter sets or updates the value (mutators). It sets the value for any variable which is used in the programs of a class.

e.g:

```
public String GetName()
{ return name;
}
```

```
public void
SetName(String N)
{ this.name = N;
}
```

Scanned with CamScanner

Constructors:

A member function used to initialize an object while creating it.

```
Employee e = new Employee();
e.setName("fazeen");
```

Syntax:

```
public Employee()
{
    name = "Name";
}
```

Constructor Overloading:

Constructor can be overloaded just like method overloading.

```
public Employee(String n)
{
    name = n;
}
```

There can be more than two overloaded constructors.

Scanned with CamScanner

(45) Inheritance

It is used to borrow properties and methods from an existing class.

Phone → Smart Phone

Super Class → Sub Class

Declaring Inheritance in Java:
Inheritance in Java is declared using "extends" keyword.

Super Class → Parent

↓

Sub Class → Child. Subclass extends the super class.

(46) Constructors in Inheritance

When a derived class is extended from the Base class, the constructor of the Base class is executed first followed by the constructor of derived class.

Scanned with CamScanner

for the following inheritance hierarchy the constructors are executed in the order ① → ② → ③

C_1 → Parent
C_2 → Child
C_3 → Grand Child

Constructors executed in top to bottom order.

Note:

Agr Base class me 2 constructors overload ho rhe hon or derived class k object se ham Constructor call kiwaen to ~~defo~~ default by default first constructor call hojata hi or agr hamne apny marzy ka constructor call kiwana ho to ham derived k constructor me super (value) use kskte hain.

(47) this Keyword:

this is a way for use to reference an object of the class which is being created / referenced.

this.area = 20; → This is a reference to current obj.

Super Keyword:

A reference variable used to refer immediate parent class object.

* can be used to refer immediate parent class instance variable
* can be used to invoke parent class methods.
* can be used to invoke parent class constructors.

(48) Method Overriding:

If the child class implements the same method present in the parent class again, it is known as method overriding. Redefining method of super class! (in sub class)

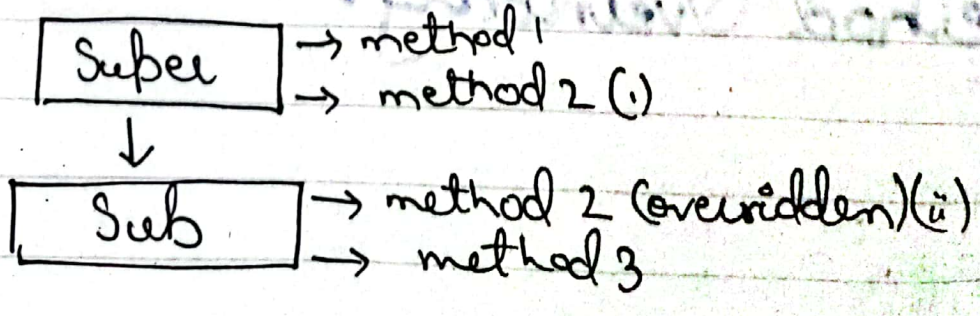
Scanned with CamScanner

When an object of subclass is created and the overrided method is called the method which has been implemented in the subclass is called and its code is executed.

(49) Dynamic Method Dispatch:-

It is a mechanism by which a call to an overridden method is resolved at runtime. It is also known as "Run Time Polymorphism". When an overridden method is called by a reference java determines which version of that method to execute based on the type of object it refer to.

Consider the following inheritance hierarchy:



Scanned with CamScanner

Scenario 1 → Super obj = new Sub(); //Allowed
obj.method2(); //Called (method of object)
obj.method3(); //not Allowed

Scenario 2 → Sub obj = new Super();
//not allowed → Error

(43) Abstract Classes and Methods:

Abstract → existing in thought or as an idea without concrete existance.

Abstract Method → A method that is declared without an implementation.
abstract void moveTo(double x, double y);
∵ ek abstract method poory class ko abstract bna deta hai!

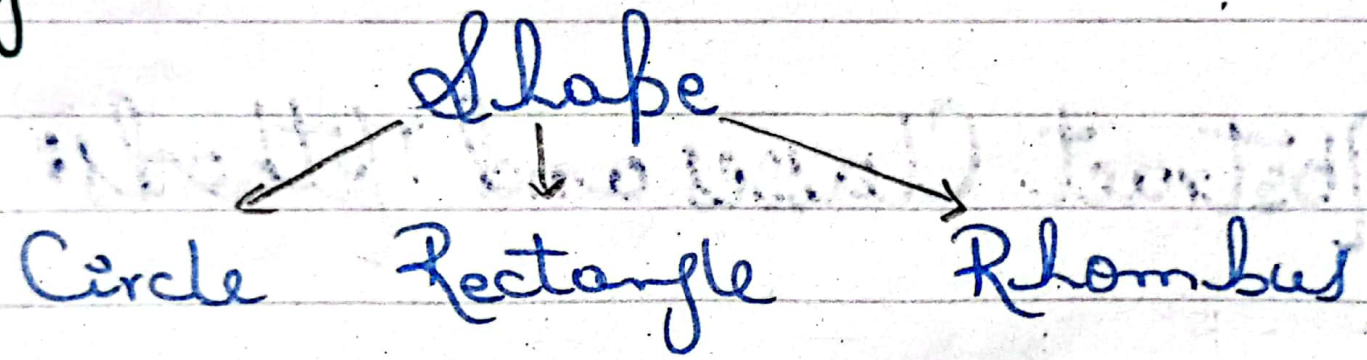
Abstract Class → If a class includes abstract methods, then the class itself must be declared abstract, as in:

```
public abstract class PhoneModel{
    abstract void switchOn();
    //more code
}
```

Scanned with CamScanner

When an abstract class is subclassed, the subclass usually provides implementations for all of the methods in parent class. If it does not, it must be declared abstract.

e.g:



- It is possible to create reference of an abstract class.
- It is not possible to create an object of an abstract class.

Scanned with CamScanner