COPYRIGHTED BY....

Instagram - @Codes learning

Youtube - Tech 21

Telegram - Codeslearningnow

Written By -

LAREKAR ARPITA RAMESH

SOFTWARE ENGINEER

Scanned by CamScanner

JAVA

PROGRAMMING

@ Codeslearning

Scanned by CamScanner

S. No.	~~Date~~	Title	Page No.	Teacher's Sign / Remarks
		Index		

	1.	Java Introduction		
	2.	Java documentation comments		
	3.	Data-types - Variables		
	4.	Java - Type - Casting		
	5.	Java - Operators.		
	6.	Control statements.		
	7.	Java Arrays.		
	8.	Java Methods		
	9.	Constructors in Java		
	10.	Java static keyword		
	11	Inheritance in Java		
	12.	Java File Handling.		
	13.	Exception Handling		
	14.	Java Project - 1		
	15.	Java Project - 2.		

Scanned by CamScanner

Java Programming.

Chapter 1. Java Introduction

- What is Java ?

Java is a popular programming language, created in 1995.
It is owned by Oracle, & more than 3 billion devices run java.

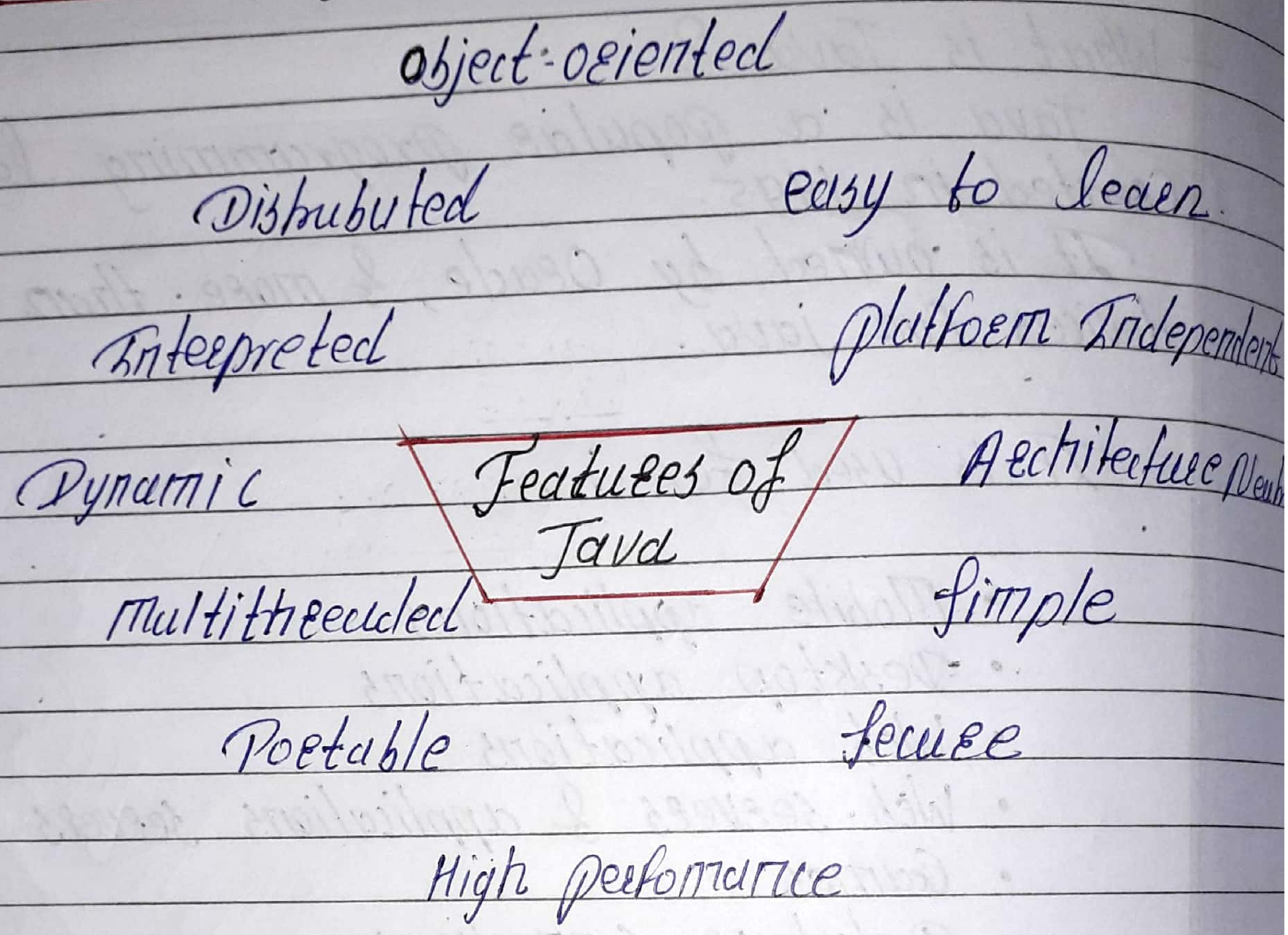
It is used for :

- Mobile applications
- Desktop applications
- Web applications
- Web servers & applications servers.
- Games
- Database connections.

- Why Use Java ?

Java works on different platforms.
(Windows, Mac, Linux, Raspberry Pi etc.)
It is easy to learn & simple to use.
It is open source & free
It is secure, fast & powerful
It has huge community support
Java is object oriented language which gives a clear structure to program.
As java is close to c++ & c# it makes it easy for programmers to switch to java.

Features of Java -



History of Java -

The history of java starts from Green team. java team members initiated a revolutionary task to develop a language for digital devices.

1] James Gosling, Mike Sheridan & Patrick Naughton initiated the Java language in 1991. The small team of sun engineers called Green team.

2] Firstly, it was called "Greentalk" by James Gosling & file extension was .gt.

Java Versions -

JDK Alpha & Beta (1995)

JDK 1.1 - 19th Feb (1997)

J2SE 1.3 - 8th May (2000)

J2SE 5.0 - 30 Sep. (2004)

Java SE 7 (28 July, 2011)

JAVA VERSIONS...

JDK 1.0 (23rd Jan, 1996)

J2SE 1.2 (8th Dec, 1998)

J2SE 1.4 (6th Feb, 2002)

Java SE 6 (11 th Dec, 2006)

Java SE 8 (18th March 2014)

Java Installation - choose the operating system -

① Windows download - • online download
• offline download

② Mac download - Oracle java can be installed on Mac versions 10.7.3 (64 bit browser)

③ linux download - Java on linux platform
Java on RPM - based linux platform

④ Solaris download - Solaris SPARC (32-bit)
Solaris SPARC (64-bit)

Note - Java comes in the form of self-executing binary file. Download the appropriate package for your computer & operating system.

Chapter-2- Java Documentation Comments.

— Java Comments —

The java comments are statements that are not executed by the compiler & interpreter. The comments can be used to provide information about the variable.
It can also be used to hide program code for specific time.

Java - Comments
- Single line Comments
- Multiline Comments.

— Java single line Comments -

The singleline comment is used to comment only one time.

• Example • Syntax - // This is a singleline Comment

```
Public class Comment Example 1 {
Public static Void main (String [] args) {
int i = 10; // Here 1 is a Variable
    System. out. Print (i);
}
}
```

Output : 10.

Java Multi-line Comment -

Syntax :

```
/*
This
is
multi line
comment
*/
```

Example :

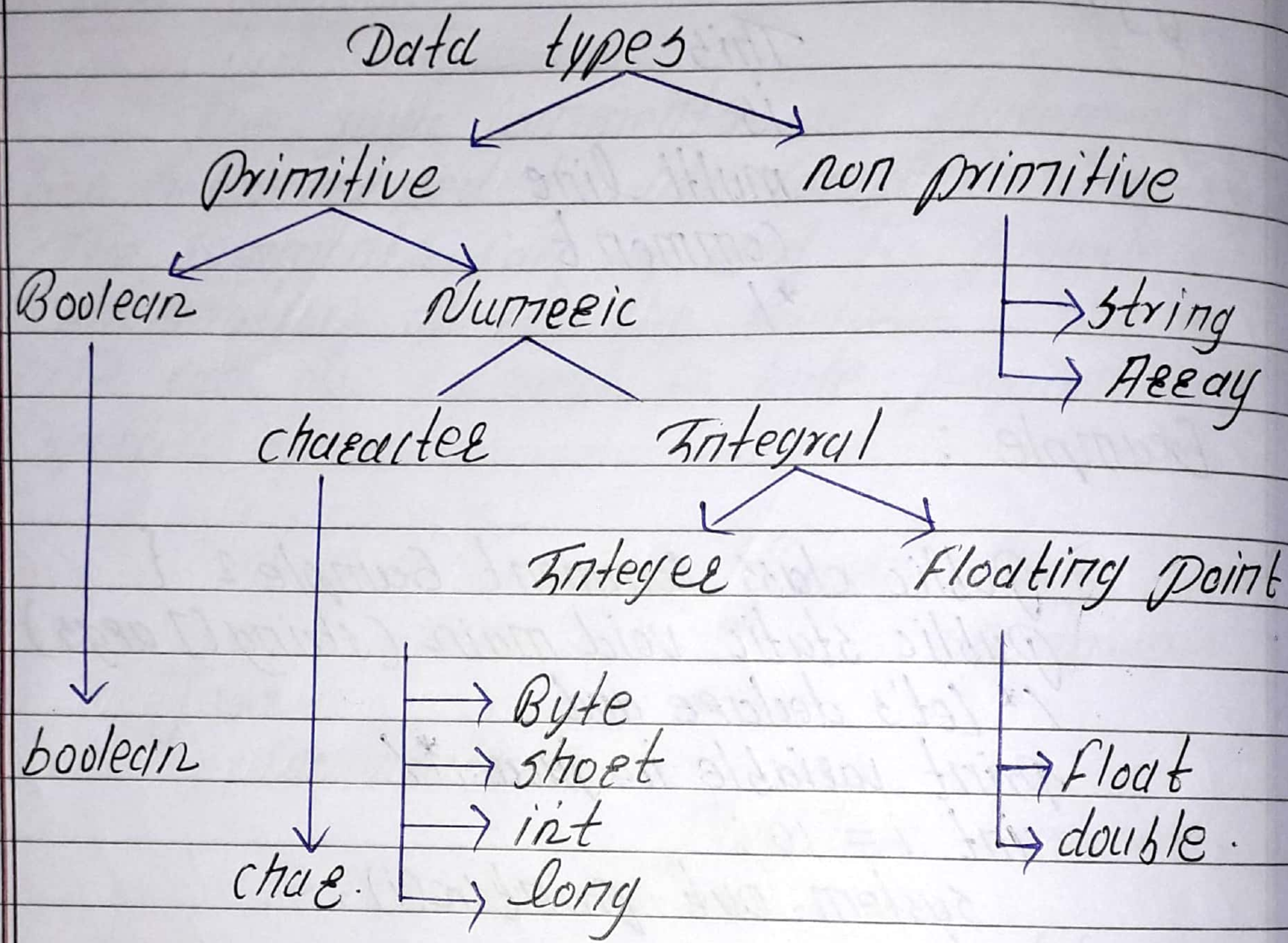
```
Public class Comment Example 2 {
Public static void main (string[] aegs) {
/* let's declaee and
print variable in java. */
int i = 10;
System. out. print ln(i);
}}
```

Output : 10

@Codes learning

Scanned by CamScanner

Chapter 3. Data types .... Variables



Example

```
Class simple {
public static void main (String[] args){
 int a =10;
 int b =10;
 int c = a+b;
 System out. println (c);
}}
```

Output = 20.

Data types	Default size
Boolean	1 bit
char	2 bytes
byte	1 byte
short	2 byte
int	4 byte
long	8 byte
Float	4 byte
double	8 byte.

— Variables -

local variable

Instance Variable

static Variable.

- **local variable** -
  A variable which is declared inside the method is called local variable.

- **Instance Variable** -
  A variable which is declared inside the class but outside the method

- **static Variable** -
  The variable that is declared as static.

Example:-

```
class A {
int data = 50; //instance variable
static int m = 100; // static variable
void method () {
int n = 90; // local variable
}
} // end of class.
```

Chapter 5 - Java Type Casting.

Type Casting is when you assign a value of one primitive data type to another type.

Java Type Casting -

- Widening Casting
- Narrowing casting.

- **Widening Casting (automatically) -**

Widening casting is done automatically when passing a smaller size type to a larger size type.

Converting a smaller type to a larger type size -

Byte -> short -> char -> int ->
long -> float -> double

- **Example -**

```
Public class Main {
 Public static void main (String[]) {
  int myInt = 9;
  double myDouble = myInt;

  System.out.println (myInt);
  System.out.println (myDouble);
```

Scanned by CamScanner

Narrowing Casting -

Narrowing Casting must be done manually by placing the type in parenthesis in front of the value.

Converting a ~~smaller~~ type larger type to a smaller size type.

double -> float -> long -> int ->
char -> short -> byte.

Example

```
Public main class main {
  public static void main (String []) {
    double my double = 9.78d;
    int myint = (int) myDouble;

    System.out.println (myDouble);
    System.out.println (myint);
  }

}
```

Java Enum -

Enum in java is a data type that contains fixed set of constants
It can be used for days of the week.
SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY
FRIDAY, SATURDAY.

The java enum constants are static & final implicitly. Its having classes that have fixed set of constants.

Simple example of java Enum -

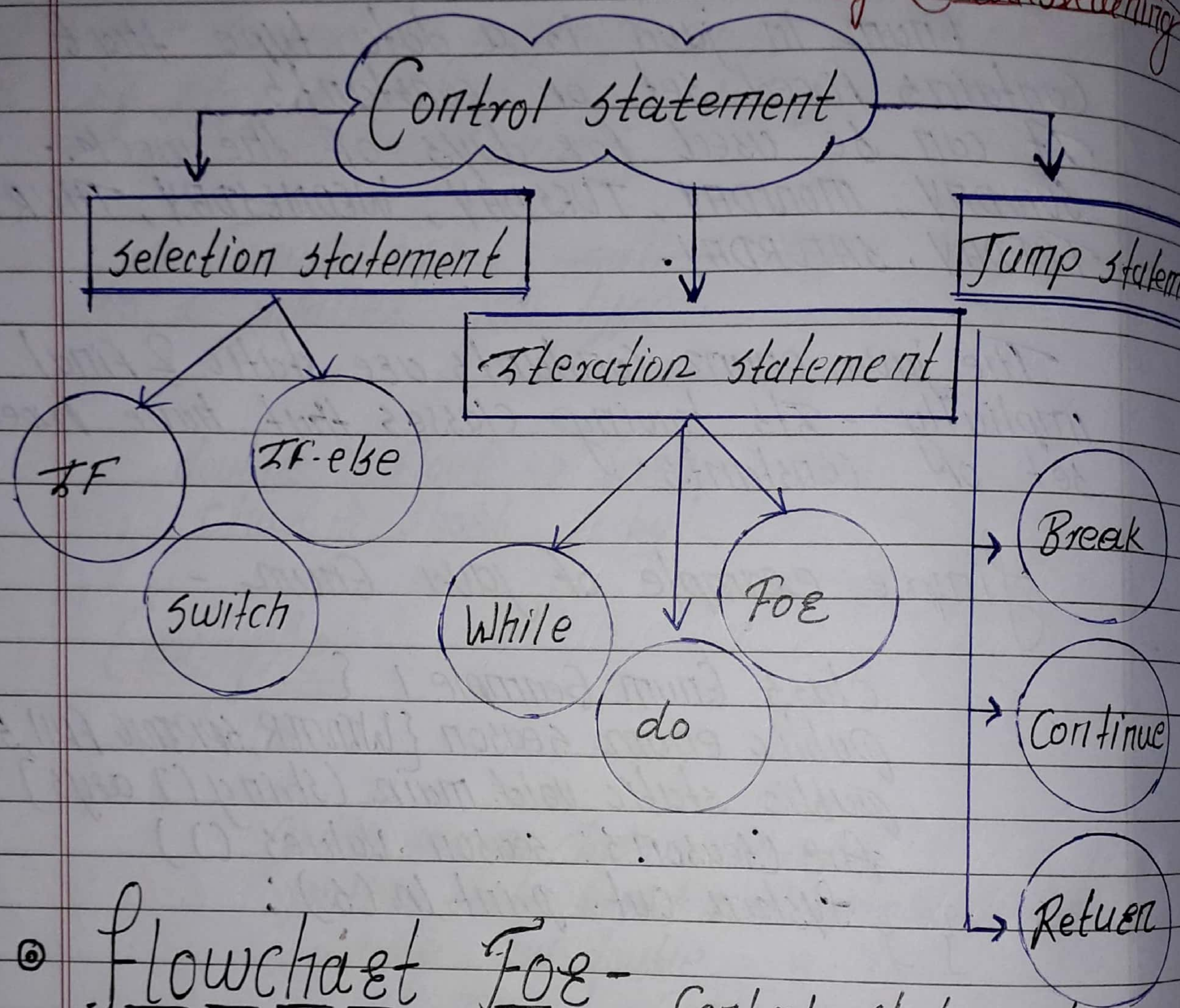
```
class EnumExample1 {
public enum season {WINTER,SPRING,FALL,SUMMER}
public static void main (String[] args) {
for (season s: season.values())
} } System.out.println(s);
```

- Output -

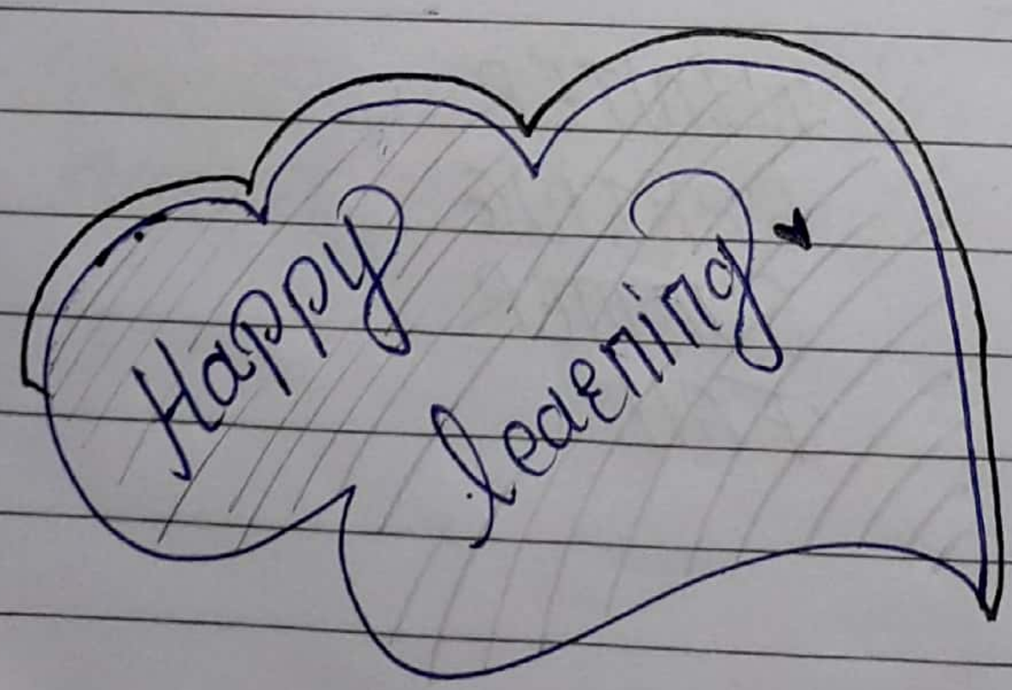
```
WINTER
SPRING
SUMMER
FALL
```

Chapter 6 - Control Statements..

by- @ Codeslearning



- **Flowchart For-** Control statement.



• Java Control statements

- Control statements are used in programing language to cause the flow of control to advance & branch based on changes to the state of program.

- Java control statements can be divided into three categories :

  - Selection statement
  - Iteration statement
  - Jump statement

• Selection Statement

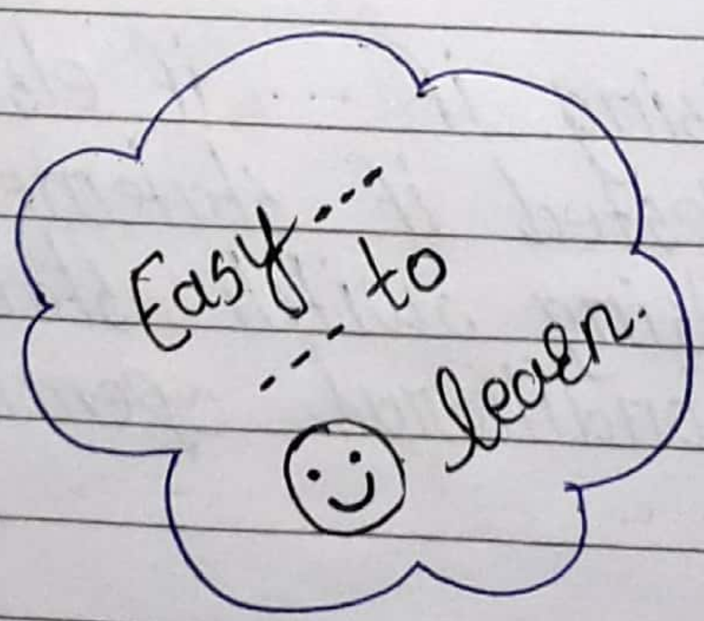
- selection statements are used in programing language to choose different paths of execution based upon the outcome of an expression or the state of a variable.

  - Using if ... if else
  - Nested if statements
  - Using switch statements.
  - Conditional operator.

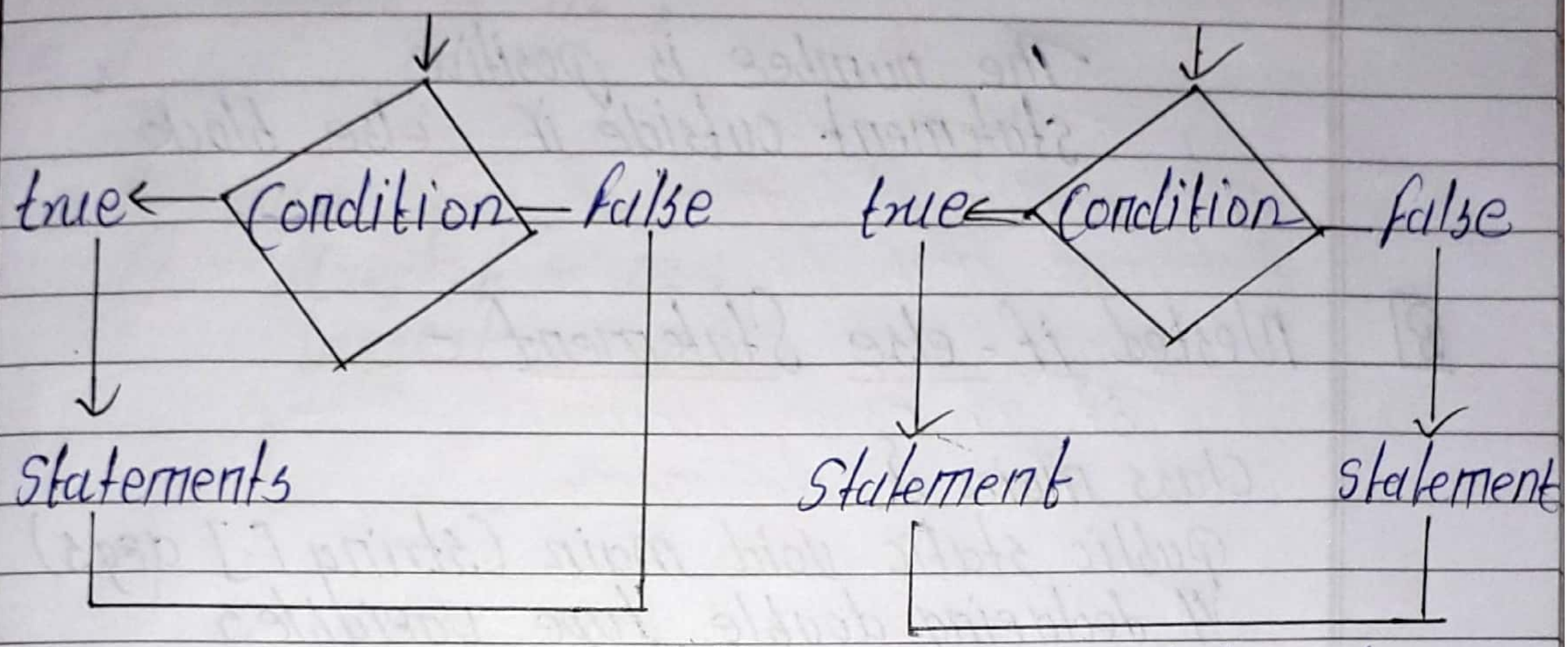
The if-else Statements.

Principal Forms :

```
if (condition) {
    statement ;        // Execute these statement if
    statement ;        // Condition is true.
}

if (condition) {       //
    statement ;        // Execute these statement if
    statement ;        // Condition is true
} else {
    statement ;        // Execute these statement if
    statement ;        // Condition is false.
}
```



Flowchart -



(a) if statement (b) if-else statement

A] Example of if-else :-

```
class Main {
 public static void main (String[] aegs) {
  int numbee = 10;
  if ( numbee > 0 )  {
      System.out.println ("The numbee is positive");
  }
   else {
      System.out.println("The numbee is not positive");
  }
  System.out.println("Statement outside if...
                                  else block");
 }
}
```

Output :-

The number is positive
statement outside if... else block

B Nested if-else Statement -

```
Class main {
 Public static void main (String [] args) {
 // declaring double type variables
 Double n1 = -1.0, n2 = 4.5 , n3 = -5.3, largest;

 if (n1 > = n2) {
  //checks if n1 is greater than or equal to n2
 if (n1 > = n3) {
    largest = n1;
  // if-else statement inside the block
  // check if n1 is greater than or equal to n3
 if
    largest = n1;
  }
    else {
      largest = n3;
    }
  } else {

  // if-else statement inside else block
  // checks if n2 is greater than or equal to n3
```

```
if (n2 >= n3)
    largest = n2;

}
else {
    ~~largest~~ = n2;
    largest = n3;
}

}

System.out.println("Largest Number :"
                    + largest);

}

}
```

<u>Output</u> - Largest Number = 4.5.

- <u>Syntax</u> -

```
if (condition) {
    // code to be executed

if (condition) {
    // code to be executed.
}
}
```

Switch Statement

The switch statement of java is another selection statement that defines different paths of execution

It is more efficient that the if statement

The expression must be of type int, short, byte char.

The selection in the switch statement is determined by the values between parenthesis after the keyword switch & expression

The break statement is used in each sequence case value statements to terminate this sequence.

The break statement is optional in the switch statement.

Scanned by CamScanner

Syntax :-

```
Switch (expression) {
    Case x :
        // Code block
    break ;
    Case y :
        // code block
    break ;
    default :
        // Code block.
}
```

Loop Statements -

In programing, sometimes we need to execute the block of code repeatedly while some condition evaluates to true. However, loop statements are used to execute the set of instructions in a repeated order.

In Java, we have three types of loops that execute similarly, However there are difference in their syntax & condition - checking time.

1. For loop.

2. while loop.

3. do-while loop.

- Let's understand the loop statements one by one

Java For loop :-

In java, for loop is similar to c & c++. It enables us to initize the loop variable, check the condition, & increment in a single line of code.

The syntax to use the loop is given below.

```
For(<initialization>, <condition>, <increment/decrement>) {

    // block of statement

}
```

Scanned by CamScanner

```
public class Calculation {
public static void main (String [] aegs) {
    // TODO Auto-generated method stub
int sum = 0;
For (int j = 1; j<=10; j++) {

sum = sum+j;
}

System.out.println ("The sum of first 10 natural
                numbers is "+ sum);

}

}
```

<u>Output</u> - The sum of first 10 natural numbers is 55.

Java While loop :-

The while loop is also used to iterate over the number of statements multiple times. if we don't know the number of iterations in advance, it is recommended to use a while loop. Unlike for loop, the initialization & increment/decrement doesn't take place inside the loop statement in while loop.

It is also known as the entry-controlled loop since the condition is checked at the start of the loop. If the condition is true, then the loop body will be executed. Otherwise the statements after the loop will be executed.

Syntax -

```
While (< condition >) {
    // loop statements
}
```

Start

condition

true

False

Statements

Terminate.

```
Public class Calcution {
public static void main (string[] aegs ) {
// TODO Auto - genrated method stub

int i = 0;

system. out . println ("Peinting the list of
                        fiest 10 even numbees\n");

While (i<=10) {

system. out. println (i);

i = i+2 ;

}
}
}
```

— O/P → Peinting the list of fiest 10 even numbers.

0
2
4
6
8
10

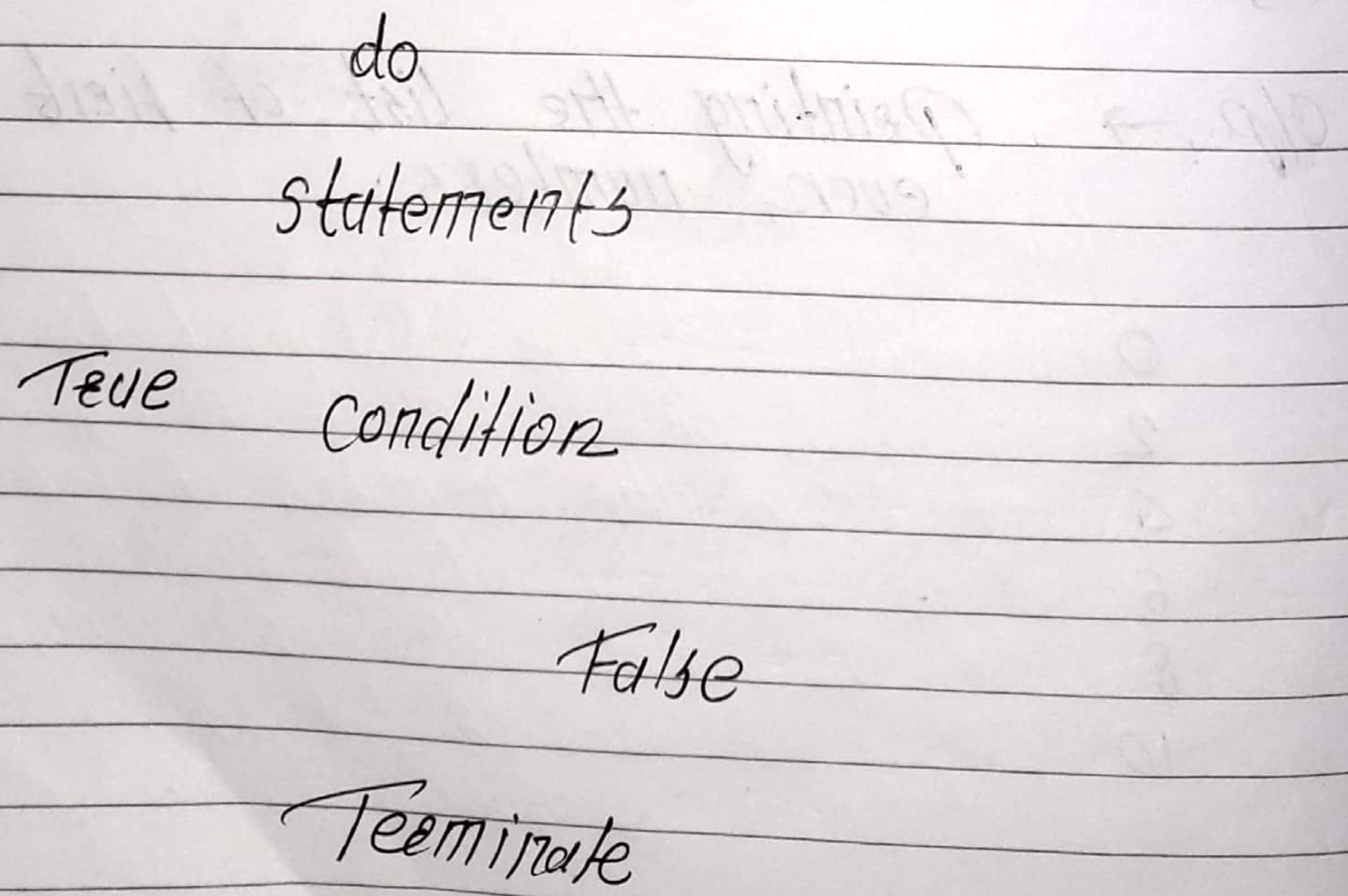
• Java do-while loop -

The do-while loop checks the condition at the end of loop after executing the loop statements. However, it is recommended to use the do-while loop if we don't know the condition in advance.

It is also known as the exit-controlled loop since the condition is not checked in advance.

Syntax -

```
do
{
// statements
} while (<Condition>);
```



```
Public class main
  {
public static void main
   {
int i = 20;
 do
  {
System.out.println (i);
 } i= i+1;
 } While (i<=20);
 }
  & nbsp;
```

output :- 20.

Jump Statement –

In java jump statements are mainly used to transfer control to another part of our program depending on the conditions. These statements are very useful from the programmers pere veiw because it allow alteration of flow of execution of the program. These statements can be used to jump directly to other statement.

① Break.

② Continue.

③ Return.

A] The Break Statement

If we want to go out of a loop then we use a break statement. If we use break statement in a loop then execution will continue with the immidiate next statement outside the loop.

After a break, all the remaining statements in the loop are skipped.

```
Package myclass 1;
class Myclass1
 {
    Public static void main (String args [])
    {
      int n =1;
      While (n<=15)
        {
          System. out. Println ("\n"+n);
          n++;
          if (n==7
            {
              Break;
            }
        }
    }
 }
```

• Output –

1
2
3
4
5
6

– Syntaxe –

```
if (condition)
{
    break;
}
```

– The Continue Statements –

The continue keywoed is used mainly with loops. when we do not want to execute some statements then we use a continue statement is confronted in the program then it will start the next iteration. It does not terminate the loop.

– Syntaxe –

```
if (condition)
{
    continue;
}
```

Scanned by CamScanner

```
package myclass1;
class Myclass1
{
    public static void main(String args[])
    {
        int[] id = {999, 888, 777, 666, 555};
        for (int x; id)
        {
            if (x == 777)

            continue;
            }
            System.out.println(x);
            System.out.println("\n");
        }
    }
}
```

— output — Run: 999
888
666
555

— Return statement —

The return statement is the last jump statement. The return statements is used to end the execution of specific method & return a value.

— Syntax —

```
if (condition)
{
    return;
}
```

```
Package myclass 1;
class Myclass 1
{
    public static void main (String [] args)
    {
        yahoo (true);
        system.out.println ("hi");
    }
    public static void yahoo ( boolean a)
    {
        system.out.println ("1");
        if (a)
        {
            return;
        }
        system.out.println ("2");
        system.out.println ("3");
    }
}
```

o/p — 1
hi

Chapter No 7 - Java Array.

@ Codes learning

Java Arrays -

Normally, an array is an collection of similar type of elements which has contiguous memory location.

Java array is an object which contains elements of similar data type. Array in java is index-based, the first element in array is stored at the 0 th index 2nd element is stored in 1st index. Java provides the features of anonymous arrays which is not available in c/c++.

0 1 2 3 4 5 6 7 8 9

Array length is 10

First index

Indices

Advantages -

◎ Code optimization -

It makes the code optimized, we can retrieve or sort the data efficiently.

◎ Random access -
We can get any data located at an index position.

Disadvantages -

• Size Limit -

We can stored only the fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection Framework is used in java which grows automatically.

Types of Arrays →
Ⓐ Single diamentional array.
Ⓑ Multi-diamentional array.

Scanned by CamScanner

Single-diamentional array -

- Syntax -

```
data Type [] arr; (or)
data Type [] arr; (or)
data Type arr [];
```

Instantiation of Array -

```
arrayRefVar = new datatype[size];
```

Let us see Example -

```
// Java program to illustrate how to declare,
   instantiate, initialize

// and traverse the Java array.

class TestArray {
public static void main (String args[]) {
 int a[] = new int [5];

 a[0] = 10;
 a[1] = 20;
 a[2] = 70;
 a[3] = 40;
 a[4] = 50;
```

Scanned by CamScanner

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

- Output -

```
10
20
70
40
50
```

- Declaration, Instantiation & Initialization of Java array -

```
// Java program to illustrate the use of declaration
// initialization of java array in a single line

class Testarray1 {
public static void main (String args[]) {
int

{33, 3, 4, 5};

for (int i = 0; i<a.length; i++)

System.out.println (a[i]);

}}
```

Scanned by CamScanner

- Output -

```
33
3
4
5
```

- Multidimentional Array -

In such case, data is stored in row & coloumn based index.

- Syntax -

```
data Type [][] arrayRefVar; (or)
data Type[][] arrayRefVar; (or)
data Type arrayRefVar [][]; (or)
data Type [] arrayRefVar[];
```

- Example :

```
class Testarray3 {
public static void main(String args[]){
  // declaring & initializing 2D array
int arr [][] = {{1,2,3}, {2,4,5}, {4,4,5}}
// Printing 2D array
For (int i=0; i<3; i++) {
for (int j=0; j<3; j++) {
System.out.print (arr [i] [j] + " ");
```

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

• Output -

```
1 2 3
2 4 5
4 4 5.
```

- Array length -

To find out how many elements an array has, use the length property:

• Example •

```
String[] cars = {"Volvo", "BMW", "Ford", Mazda"};
System.out.println(cars.length);

// output 4.
```

• JAVA METHODS •

- chapter No-8

• Java Methods -

A method is a block of code which only runs when it is called.

You can pass data, known as a parameters, into a method.

Methods are used to perform certain actions, & they are known as functions.

• Create a Method -

A method must be declared within a class. It is defined with the name of methods, followed by parenthesis ().
Java provides some pre-define method

System.out.println()

But you can also create your own methods to perform certain actions.

Scanned by CamScanner

- Example -

```
public class Main {
    static void myMethod () {

        // code to be executed

    }

}
```

- Syntaxe -

```
public static int methodName (int a, int b) {
    // body.

}
```

Here,

- Public static - Modifiee.
- Int - Return type
- Method Name - name of the method
- a, b - formal parameters
- int a, int b - list of parameters.

- Syntax -

```
Modifiee returnType name O Method
                parameter list {

        // method body

}
```

- Modifiee - It defines the access type of method
- return Type - Method may return a value.
- Name () method - This is a method name. The method signature consists method Name
- Parameter list - It is type, order & Numbe of parameter of a method.
- method body - The method body defines what the method does with statements.

@ Codes Learning

- **Access Specifier** –
  Access specifier or modifier is the access type of the method. It specifies the visibility of the method.

Access Specifier
- Public
- Private
- Protected
- Default.

- **Public** –
  This method is accessible by all classes when we use public specifier in our application.

- **Private** –
  When we use private class to access specifier, the method is accessible only in the classes in which it is defined.

- **Protected** –
  When we use protected access specifier, the method is accessible within the same package or subclass.

Scanned by CamScanner

- **Default** -
When we do not use any access specifier in the method declaration, java uses default access specifier by default.

Types of Methods.

(A). Pre-defined method (B) User-defined method

- Predefined method -

Predefined methods are the methods that is already defined in java class library is known as predefined methods. It is also known as standard library method.

We can directly use these methods just by calling them in the program. Some predefined methods are - length (), equals (), compareTo (), sqrt (), etc

Each and every predefined method is defined inside a class such as print () is defined in java.io.PrintStream class.

Demo.Java -

```
Public class Demo
{
Public static void main (String [] args)
{

// using the max () method of Math class
System.out.println ("The Max number is:" Math.
                                Max (9, 7));
}
}
```

Output -

Maximum Number is : 9.

- User-defined Method -

The method return by the user or programmer is known as user-defined method.

- To checks the number is even or odd -

```
// user defined method
public static void findEvenOdd (int num)
{
// method body
if (num % 2 = = 0)
System.out.println (num + " is even");
else
System.out.println (num + " is odd");
}
```

- Even odd.java -

```
impoet java.util.scanner;
public class Even odd
  {
public static void main (string aegs[])
  {
Scanner scan = new scanner (system.in);
System.out.print ("Enter the number: ");
int num = scan.nextInt ();
    // method. calling
find Even odd (num);
  }
public static void findEvenodd (int num)
  {
if (num % 2 == 0)
System.out.println (num + " is even");
  else
System.out.println (num + " is odd");
}
}
```

output 1 -

Enter the number : 12
12 is Even.

output 2 :

Enter the number : 99
99 is odd.

- Static Method -

A method that has static keyword is known as static method.

The main advantage of static method is that we can call it without creating an object. It can access static data members

We can also ~~use~~ create a static method by using the keyword static before the method name.

It is used to create an instance method It is invoked by using the class name.

The main best example of static method is the main () method.

- Example of static Method.

Display.Java -

```
public class Display
  {
public static void main (String [] aegs)
  {
show ();
  }
static void show()
  {
System.out.println ("It is example of static
                                  method.");

  }
}
```

- Output -

It is an example of static method.

Scanned by CamScanner

- Instance Method -

The method of a class is known as an instance method.
It is a non-static method defined in the class. Before calling or invoking the instance method, it is necessary to create an object of its class.

InstanceMethodExample.java -

```
public class InstanceMethodExample
{
public static void main (String [] args)
{

InstanceMethodExample obj = new InstanceMethodExample
                                    ();
System.out.println ("The sum is : " + obj.add(12,15
}

int s;

public int add (int a, int b)
{
```

```
s = a+b;
return s;
}
}
```

- Output –

The sum is : 25

Instance Method	∘ Accessor Method ∘ Mutator Method.

A. Accessor Method –

The method (s) that reads the instance variable (s).
We can easily identify it because the method is prefixed with word get. It is known as getters. It return the value of the private feild.

• Example -

```
public int getId ()
{
return Id;
}
```

B] Mutator Method -

• The method read the instance variable & also modify the values.

• We can easily identify it because the method is prefixed with word set.

• It is also known as setters.

• It does not return anything. It accept a parameter of the same data type depends on the feild.

Example -

```
public void setRoll (int roll)
{
this . roll = roll;
}
```

c] Abstract Method -

The method that does not has method body is known as abstract method.
In other words, without an implementation is known as abstract method.
It always declares in the abstract class. It means the class itself must be abstract if it has abstract method.

- Syntax -

```
abstract void Method_name();
```

- Example -

```
abstract class Demo
{
abstract void display ();
}
public class My class extends Demo
{
void display ()
{
System.out.println ("Abstract method ?");
}
```

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

Demo obj = new Myclass ();

obj.display ();

}

}
```

- Output –

Abstract Method...

Happy learning :)

Scanned by CamScanner

Constructors In Java

chapter no - 9.

Constructors -

In java, a constructor is a block of codes similar to the method. It is called when an instance of class is created. At the time of calling constructors, memory for the object is allocated in the memory.

It is special type of method which is used to initialize the object.

Rules for Constructor -

1]. Constructor name must be same as its class name.

2]. A constructor must have no explicit return type.

3]. A java Constructor cannot be abstract static, final & synchronized.

Scanned by CamScanner

Types of Constructors

1] Default Constructor

2] parameterized Constructor

1] Java default Constructor -

A Constructor is called "Default Constructor" when it doesn't have any parameter.

- Syntax -

<class_name> () { }

- Example -

```
class Bike1 {
Bike1() {system.out.println("Bike is created");}
public static void main (String args[])
```

```
Bike1 b = new Bike1();

}

}
```

- Output – Bike is created.

- **Java Parameterized Constructor –**

A constructor which has a specific number of parameters is called a parameterized constructor.

- Example –

```
class student4 {

int id;

String name;

student4(int i, String n) {

id = i;

name = n;
}
}
```

```
Void display () {system.out.println(id+" "+
                                    name); }

public static void main (String args[]) {

Student4 s1 = new Student4 (111, "Karan");

Student4 s2 = new Student4 (222, "Aryan");

s1.display ();

s2.display ();

  }

}
```

- Output -

```
111 karan

222 Aryan.
```

@codesfocoding

Constructor Overloading -

Constructor overloading in java is a technique of having more than one constructor with different parameter lists. They are arranged in a way that each constructor performs a different task.

- Example -

```
class students {
int id;
String name;
int age;
students(int i, String n) {
int id = i;
name = n;
}
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[]) {
    Student5 static void S1 = new student5(111, "Karan");
    Student5 S2 = new student5(222, "Aryan", 25);
    S1.display();
    S2.display();
    }
}
```

Output -

```
111 Karan 0
222 Aryan 25
```

Chapter No - 10

Java Static Keyword

@Codes_learning

• Static Keyword -

The static keyword in java is used for memory management mainly. We can apply static keyword with variables, methods, blocks & nested classes.

The static keyword belongs to the class than an instance of the class.

The static can be -

1. Variable
2. Method
3. Block
4. Nested class.

A. Java Static Variable -

The static variable can be used to refer to the common property of all objects for example - the company name of employees, college name of students etc.

The static variables gets memory only once in the class area at the time of class loading.

Scanned by CamScanner

// Java program to demonstrate use of static variable.

```
class student {
 int roll no ;  // instance variable
 string name ;

 static string college = "ITS";

 // constructor

 Student ( int r, string n) {

  ~~roll~~ rollno = r;
        name = n;
 }

                void display ()

 { system.out.println (roll no +" "+ name+" "
                + college ); }

 }

 public class TestStaticVariable1 {

 Public static void main (string args [] ) {

  student s1 = new student (111, "karan");
```

```
Student s2 = new student (222, "Aeyan");

    S1 . display ();

    S2 . display ();

  }

}
```

★ Output -

```
111 Kaean ITS
222 Aeyan ITS.
```

B]. Java Static Method -

If you apply static keywoed with any method, it known as static method.

i] A static method belongs to class eather than the object of class.

ii] static method can be invoked without the need foe creating an instance of class.

// Java program to demonstrate use of - static method.

```
class student {
    int roll no;
    string name;
        static string college = "ITS";
    static void change () {
        College = "BBDIT";
    }
    student (int r, string n) {
        roll no = r;
        name = n;
    }
                    Void display()
    { system.out.println (rollno + " " + name + " " + college); }
}
public class TestStaticMethod
```

```
public static void main (string args[]){
    student.change();
    student s1 = new student (111, "Karan");
    Student s2 = new student (222, "Aryan");
    Student s3 = new student (333, "Sonoo");
        S1.display();
        s2.display();
        s3.display();
    }
}
```

* Output –

111 Karan BBDIT.

222 Aryan BBDIT.

333 Sonoo BBDIT.

(i) It is used to initialize the static data member.

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

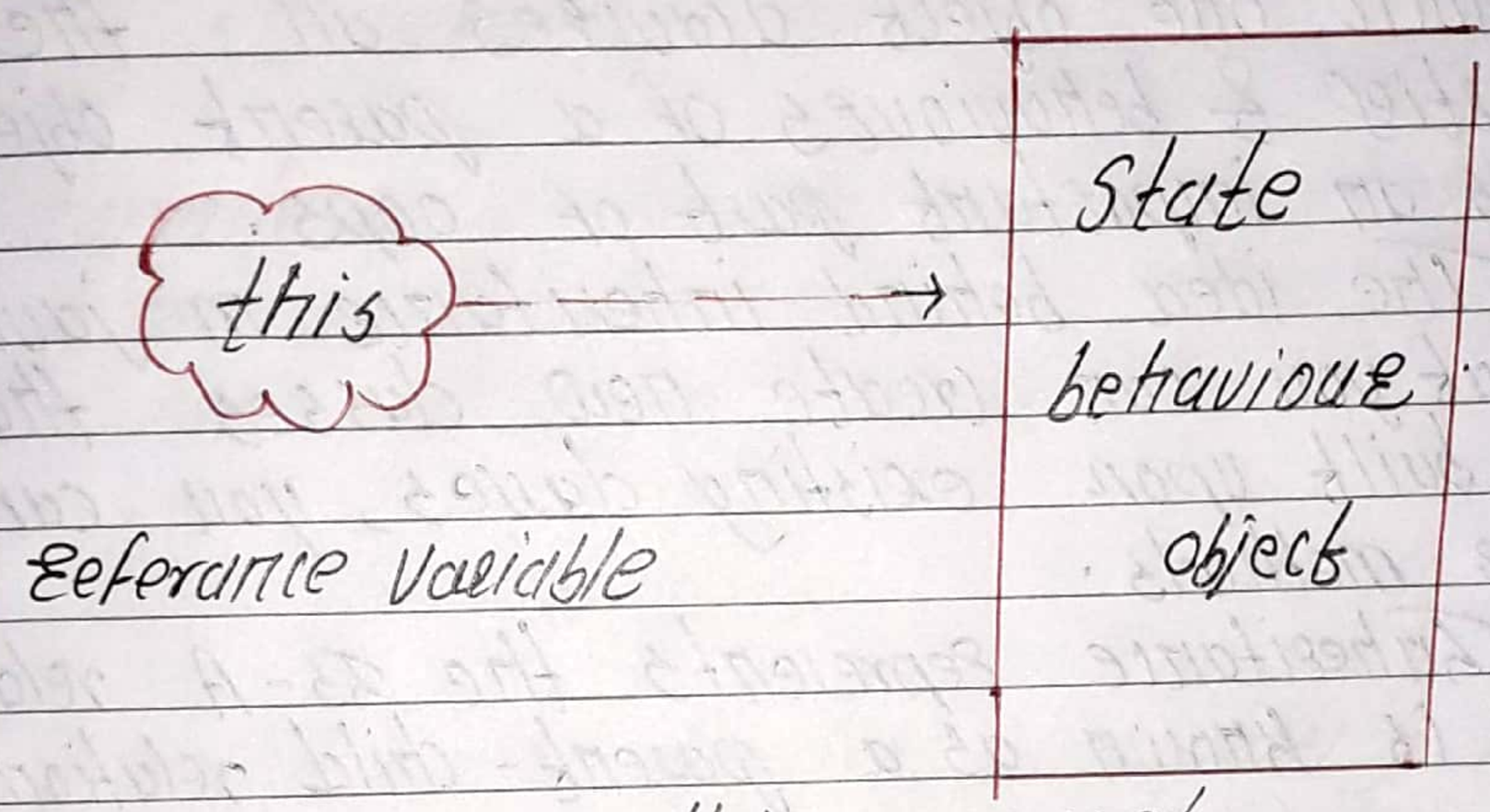
```
Class A2 {

    static {system.out.println("static block is invoked");}

    public static void main (string args[]) {

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

    }

}
```

★ — Output

```
Static block is invoked
Hello main.
```

- **this keyword in Java :-**

There can be a lot of usage of Java this keyword. In Java, this is a reference variable that refers to the current objects.



- Usage of Java this keyword -

1] this can be used to refer current class instance variable

2] this can be used to invoke current class method (implicitly)

3] this can be used to invoke current class constructor.

4] this can be passed as an argument in the method call.

5] this can be passed as an argument in the constructor call.

• Chapter No - 11 - @ Codes learning

Inheritance in Java.

• — Inheritance -

Inheritance in java is a mechanism in which one object acquires all the properties & behaviours of a parent object. It is an important part of OOPS.

The idea behind inheritance in java is that you can create new classes that are built upon existing classes, you can reuse methods.

Inheritance represents the IS-A relationship which is known as a parent - child relationship.

• — Why use inheritance in java ?

- For Method overriding
- For Code Reusability.

• — Terms Used in Inheritance -

- **class -**

A class is a group of objects which have common ~~practi~~ properties. It is a template from which object is created.

- Subclass / child class -

Subclass is a class which inherits the other class. It is also called a derived class, extended class or child class.

- Superclass / Parent class

Superclass is the class from where a subclass inherits the features. It is also called a base class or parent class.

- Reusability.

As the name specifies, reusability is a mechanism which facilitates you to reuse the feilds & methods of existing class when you create a new class. You can use the same feilds & methods already defined in the previous class.

- Syntax of Java Inheritance -

```
class subclass-name extends superclass-name
{
    // methods & feilds
}
```

Java Inheritance Example

Employee

Salary : float

↑

Programmer

bonus : int

```
class Employee {
float salary = 40000;
}
class programmer extends Employee {
int bonus = 10000;
public static void main (String args[]) {
programmer p = new programmer ();
system.out.println("programmer salary is:"+p.salary
System.out.println("Bonus of programmer is "+p.bonus
}
}
```

- Output -

Programmer salary is : 40000.0

Bonus of programmer is : 10000

- Encapsulation in Java -

Chapter - 12.

- Encapsulation -

It is the process of wrapping code & data together into a single unit;

The java Bean class is the example of a fully encapsulated class.

- Advantages -

- By providing only a setter or getter method, you can make the class read only or write only.
- It provides you the control over the data.
- It is a way to achieve data hiding in java because other class will not be able to access a data through private data members.
- Easy to test.

```
Package com.Javatpoint;
class Test {
Public static void main (string[] aegs) {
Student s = New student ();
S. setName ("Vijay");
System.out.println (s.getName());
}
}
```

- Output - Vijay.

@ codes learning

classmate
Date
Page

chapter No - 13

Java file Handling - @ Codes learning

file handling is an important part of any application.
Java has several methods for creating, reading updating & deleting files.

The file class from the java.io package allows us to work with files.
To use the file class, create an object of the class & specify the filename.

Example -

```
import java.io.file;
file myObj = new file("filename.text");
```

Scanned by CamScanner

Method	Type	Description
CanRead ()	Boolean	Tests whether the file is readable or no
CanWrite ()	Boolean	Test whether the file is writable or not
CreateNew File ()	Boolean	Creates an empty file.
Delete ()	Boolean	Delets a files.
exists ()	Boolean	Test whether the file exists.
getName ()	string	Returns the name of file.
length ()	Long	Return size of files in Byte.
list ()	string[]	Returns an array of files.
mkdir ()	Boolean	creats a dictorenay

Java Create & Write To files

- Create a file

To create a file in java, you can use the createNewFile() method. This method returns a boolean value: true if the file was successfully created & false if the file already exists.

Note that the method is enclosed in try... catch block. This is necessary because it throws an IOException if error occurs.

```
import java.io.File;
import java.io.IOException;

public class CreateFile {
  public static void main (String[] args) {
    try

      File myObj = new File("filename.txt");
      if (myObj.createNewFile()) {

        System.out.println("File created: " + myObj.
                                                getName());
      } else {

        System.out.println("File already exists.");
```

```
    }

} catch (IOException e) {
    System.out.println("An error occured.");
    e.printStackTrace();
}

}
```

Output -

File created :

Filename.txt.

- Write To a file -

In the given ex. we use the FileWriter class together with its write() method to write some text to file we created in the example.

Example -

```
import java.io.FileWriter;

import java.io.FileWriter;

Public class WriteToFile {

    Public static void main (String[] args) {
        try {

FileWriter myWriter = new FileWriter ("filename.txt");

myWriter.write("Files in Java might be tricky,
                but it is fun enough!");

myWriter.close();

System.out.println("successfully wrote to file.");

} catch (IOException e) {

System.out.println("An error occured.");

e.printStackTrace();

}

}

}
```

output- Successfully wrote to file.

- **Read a file -**

In the following example we use the scanner class to read the contents of text file we created in previous chapter.

- **Example -**

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
public class ReadFile {
    public static void main (String [] args) {
        try {
File myobj = new file ("filename.txt");
Scanner myReader = new Scanner (myobj);
while (myReader.hasNextline ()) {
    String data = myReader.Nextline ();
```

```
System.out.println(data);
}
myReader.close();
}
catch (FileNotFoundException e) {
System.out.println("An error occured.");
e.printStackTrace();
}
}
```

- Output will be -

Files in java might be tricky, but it is fun enough!

Scanned by CamScanner

Date
Page

- Java Delete a file -

To delete a file in Java, use the delete () method :

- Example -

```
import java.io.Files;
public class DeleteFile {
    public static void main (String [] args) {
        File myObj = new File ("Filename.txt");
        if (myObj.delete()) {

            System.out.println("Deleted the File :
                                " + myObj.getName());
        } else {
            System.out.println ("Failed to delete the
                                file.");
        }
    }
}
```

output - Deleted the file : filename.txt

Exception Handling -

Chapter No - 14

@ Codes learning

- What is exception handling ?

The exception handling in java is one of the powerful mechanism to handle the runtime errors, so that the normal flow of the applications can be ~~d~~ maintained.

- Dictionary meaning -

Exception is an abnormal condition.

- Exception handling is the mechanism to handle runtime errors such as, classNotFound Exception, IOException, SQLException etc.

- Advantages -

The core advantage of exception handling is to maintain the normal flow of the application.

An exception normally disrupts the normal flow of the applications.

Thats why we need to handle exceptions.

Scanned by CamScanner

Hierarchy of java Exception -

Throwable

- Exception
  - IOException
  - SQLException
  - ClassNotFound Exception
    - Runtime Exception
      - Arithmatic Exception
      - NullPointer Exception
      - NumberFormat Exception
      - IndexOut of Bounds Exception
        - ArrayIndexOutOf BoundsException
        - StringIndexOutOf BoundException
- Error
  - StackOverflowError
  - VirtualMachineError
  - OutOfMemoryEror Error.

Types of java Exception

- Checked Exception
- Unchecked Exception
- Error.

- Java Exception keywords -

Keyword	description
Try	The try keyword is used to specify a block where we should place exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.
Catch	The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone.

Finally -	The "Finally" block is used to execute the necessary code of the program, it is executed whether an exception is handle or not.
throw	The "throw" keyword is used to throw an exception
throws	The "throws" keyword is used to declare exceptions it specified that there may occurs an exception in the method. It doesn't throw an exception.

These are the some kinds of Exception Keywords.

Exception Handing Example java -

```
public class JavaExceptionExample {
    public static void main (String args[]) {
    try {
        int data = 100/0;
                } catch (ArithmaticException e)
    { system.out.println (e); }

        System.out.println ("rest of code...");

    }

}
```

Output -

Exception in thread main java.lang.ArithmaticException: / by zero rest of code.

JAVA PROJECT

FOR

BEGINNERS ...

@ Codes learning.

Scanned by CamScanner

Project -1

Breakdown Ball Game -

Developing a Java GUI application to implement the Breakdown Ball Game.

Prequisite (s)

Java

Skills to be learned -

Java Swing

Java AWT

JFrame

JPanel

Scanned by CamScanner

Getting started

- Setting up the development environment.

First you need to set up the development environment for the Java CLI project. For this you need to installed java 8 or higher.

Additionally you need a code-editor or IDE.

Requirements -

1. Install Java (JDK 1.8)
2. Install any IDE - intellij or Eclipse.
3. Do the environment setup for java on your local machine.
4. Create a java project using command line app as the template.
5. Provide the project Name & base package.

- Refeerances -

- Installing Java
- Installing Eclipse
- Installing Intellij IDEA.
- Installing Visual studio Code.

- Tip -

you can use online IDE as well. For that you don't need to installed any IDE's for that instance.

- Online IDE.

- Expected outcome -

You should be able to set up the initial development environment required to develop the Game.

- Game terminal -

Design the Game Terminal

Game terminal is the window where the game is to played. It consists of the following aspects.

- Window size.
- Window visibility
- Game title
- Default close operation.

Requirements -

1. Create a main class inside the base package.
2. Import java swing dependancy
3. Create a Jframe object

Scanned by CamScanner

(4) Design the window terminal by writing your code.

Congrats !! You have sucessfully designed the Game terminal.

Referances -

- Java swing
- Java Jframe
- Interactive Console application

- Being it On -

- You can use other java frameworks as well apart from Jframe to implement a graphical user interface.

Scanned by CamScanner

- Gamplay -

Game design sits under the broader feild of game development & refers to use of creativity & design to develop a game for entertainment or educational purpose.

Gameplay design is responsible for the central part of game experience.

- Requirement

- JPanel
- Actionlistner
- Keylistner.

@ Codes learning.

Design the Props -

- Design the Props (Bricks) using 2D Array -

Breakout begins with four to eight rows of bricks with each two rows a different colour. The colour order from the bottom up is yellow, green, orange, red.
Using a single ball, the player must knock down as many briks as possible by using walls.

Requirements -

1. Create a Map Generator class inside the base package
2. Crea Import Java AWT dependency
3. Design the bricks.
4. Implement the code for calculating score in Gameplay class.

Scanned by CamScanner

classmate
Date
Page

Congratulation !!

- You have successfully designed the entire game.

References -

- Java Graphics 2D.

Bring It on -

Use of Graphics2D will help to - draw the bricks over the terminal

@ Codes learning

- Play the Game -

Run the Game in terminal -

Wrap up the code of game & start it off right away using terminal directly.

Requirements -

1. Build the project
2. Import any dependencies
3. Run the main () class
4. play the Game in terminal.

Congratulations !

you have successfully started your Game.

Reference - Java Graphics 2D.

Code for Breakdown Ball Game -

```
Package bredkoutBall ;
impoet java.awt.Color ;
impoet java.awt.Font ;
impoet java.awt.Graphics ;
impoet java.awt.Graphics2D ;
impoet java.awt.Rectangle ;
impoet java.awt.event.actionevent ;
impoet java.awt.event.actionlistner ;
impoet java.awt.event.Keyevent ;
impoet java.awt.event.Keylistner ;
impoet java.swing.Timer ;
impoet javax.swing.Jpanel ;
impoet
```

```
public class Gameplay extends JPanel.
    @ Overeide.
    Public void KeyPressed (KeyEvent e) {
        if (e.getKeyCode () == KeyEvent.VK_Right)
            {
        if (playerX >= 600)  {
            playerX = 600;
        }
        else {
            moveRight ();
        }
    if (e.getkeyCode () == KeyEvent.VK_LEFT )
            {
    if (playerX < 10)  {
            playerX = 10 ;
        }
}
```

```
if (e.getKeyCode() == KeyEvent.VK_ENTER)
{
        if (!play)
        {

            play =      true;
            ballposX =  120;
            ballposY =  350;
            ballXdir =  -1;
            ballYdir =  -2;
            playerX =   310;
            score =     0;
            totalbricks = 36;

            map = new MapGenerator(4,9);

            repaint();

        }

    }

}
```

@CodesLearning

```
public void moveRight () {
    play = true ;
    playerX += 15 ;

  }

}
```

@codes Dedening

Scanned by CamScanner

Contact App Android

PROJECT - 2

Learning development by building stuff is the best way to go about it. Here we'll be building our own

Contact/phone Book application. This application will help us to manage our contacts, having features like, resister as a user & store data for that user account The data will contains name & phone, number of entities.

Prequisite (s) -

Java, solite (basic DBMS queries)

Skills to be learned -

Java
Custom Recycler views in Android
solite.

Scanned by CamScanner

Getting The Concept Ready -

This task is all about the concepts that will be needed in building our application

Requirements -

1. Understanding how activities are rendered in the application

2. Understanding the concept of Activity life cycle.

3. learning about Intents in Android, which will be your best friend in Android application development.

Bring It On -

- Try creating the basic layout of your application, may be try to draw out the design of your app.

- XML is always there to make your designs easier & easy to visualize too.

Setting up the main project -

It's time to work on the actual project. We'll start off by creating the basic structure of project.

Requirements -

- Add all necessary dependencies for your project in the (build.gradle) file. Gradle is a build system commonly used in all java applications to automate building testing & deployment.
- (build.gradle) contains (scripts) to automate several tasks like simply copying some file from one directory to other.
- It contains (compiledsdkversion) (buildtypes) & dependencies.

① appcompat - maven dependency for using frameworks.

② Recyclerview - dependency for using recycler view in app.

③ (cardview) - for using (cardview) class with viewholder class.

dependencies {

implementation 'com.android.support:appcompat-v7:28.0.0'

implementation 'com.android.support:recyclerview-v7:28.0.0'

implementation 'com.android.support:cardview-v7:28.0.0'

implementation 'com.android.support.constraint:2.0.4'

implementation 'com.android.support.design:28.0.0'

implementation 'de.hdodenhof:circleimageview:3.0.0'

- Each activity can be designed using xml & the component (xml) files can be found in the layout folder of the res directory. In the activity.xml file we can design our activity screen using xml & drag methods.

- Always remember to map the elements in the .xml file to your class activity files

- The drawable folder under res directory will contains all your images.

Referances -

- Dependency in build gradle
- Creating New Activities
- Layouts & Views in Android
- Intents & Uses.

Registering Users & Authenticating

Login:-

The basic idea of our application is that multiple user can resister on the application & store details of various contacts & access the same on login. In milestone, we'll be accepting user data on resistration which needs to be fed to the database & then while login the user will authenticated when the user enters valid input in the placeholders & clicks on login button.

Requirements -

- Creating placeholders for the name, username, mobile number, password for the user input in the Registration activity.
- Create a Userdata class which will contains the way in which data will be stored in our database
- SQLiteOpenHelper is used to create an object to create tables. Here also defined schema of database by executing a create table query in the onCreate () method in DatabaseHelper class.

References -

- SQLite Helper
- SQLite Basic Commands
- Accessing Database
- ContentValues.

JAVA CODE

```
import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import com.amulyakhare.textdrawable
import com.amulyakhare.textdrawable.util.
                                ColorGenerator;
import Java.util.ArrayList;

Class ContactRVAdapter extends RecyclerView.
                        Adapter< ContactRVAdapter> {
```

```
private context context;
private ArrayList<ContactModal> ContactModalArrayList;

public ContactRVAdapter (context, context, ArrayList<ContactModal>
                                ContactModalArrayList) {

    this. Context = Context;
    this contactsModalArrayList = ContactModalArrayList;
}

public ContactRVAdapter. ViewHolder onCreateViewHolder. {
    return new ContactRVAdapter. ViewHolder. inflate;
public void filterlist {
        ContactModalArrayList = filterlist;
        notifydatasetchanged ();
}
public void onBindViewHolder. {
    ContactModal modal = ContactsModalArrayList.get;
```

```
holder.ContactTV.setText ;
ColorGeneratorE generatorE = ColorGenratorE.MATERIAL;

int Color = generator.getRandomColor();

TextDrawable drawable2 = TextDrawable.builder()

        .width(100)
        .height(100)
        .endconfig()

    .buildRound(modal.getUserName().substring(0,1));

holder.itemview.setOnclicklisteme.{

public void onclick(view v) {

        Intent i = new intent ;
        i.putExtra("name", modal.getusername());
        i.putExtra("contact", modal.getContactNum());

        Context.startActivity(i);
        }
    });

}
```

```
Public int getItemCount () {
        Return ContactModelArrayList.size ();
}

    public class ViewHolder extends RecyleView. {

    private ImageView contactIV ;
    private TextView contactTV ;

Public ViewHolder {
        super (itemView);

    ContactIV = itemView.findViewById Id ;
    ContactTV = itemView.findView By Id ;

    }

}

}
```

@ codes learning.

Scanned by CamScanner