PAGE No. 1
DATE 31/01/22

INTRODUCTION AND HISTORY OF JAVA

1. Java is a high-level, general-purpose Programming language that is class-based and object oriented used to develop an application.

2. Java was originally developed by James Gosling at Sun microsystem and released in 1995. James Gosling and his team members started the Project in early 90s.

3. James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language Project in June 1991.

4. The language was initally called Oak after an oak tree stood outside Gosling's office

5. Why had they chosen java name for Java language ? The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "Silk", "Jolt", "DNA", etc. They wanted Something that reflected the essence of the technology: revolutionary, dynamic, lively. cool, unique. and easy to Spell and fun to Say.

Teacher's Signature:.....

PAGE NO. 2
DATE 31/01/22

6. Why Oak? Oak is a Symbol of strength and chosen as a national tree of many countries like the U.S.A, France, Germany, Romania, etc.

7. In 1995, Oak was renamed as "Java"

8. Java is an island of Indonesia where the first Coffee was Produced (called java Coffee.) it is a kind of espresso bean. Java name was chosen by James Gosling while having a Coffee near his office

9. Later the Project went by the name Green and was finally renamed Java, from Java Coffee, the Coffee from Indonesia. Gosling designed Java with a C/C++ style syntax that System and application Programmers would find familiar.

10. Sun Microsystems released the first Public implementation as Java 1.0 in 1996.

11. JDK 1.0 released in (January 23, 1996). After the first release of Java, there have been many additional feature added to the language. Now Java is being used in Windows applications.

Teacher's Signature.....

Web applications, Mobile application, Cards, etc. Each new version adds the new features in Java.

1. JDK Alpha and Beta (1995)
2. JDK 1.0 (23rd Jan 1996)
3. JDK 1.1 (19th Feb 1997)
4. ~~JDK~~ J2SE 1.2 (8th Dec 1998)
5. J2SE 1.3 (8th May 2000)
6. J2SE 1.4 (6th Feb 2002)
7. J2SE 5.0 (30 sep 2004)
8. Java SE 6 (11th Dec 2006)
9. Java SE 7 (28th July 2011)
10. Java SE 8 (18th March 2014)
11. Java SE 9 (21st Sep 2017)
12. Java SE 10 (20th Mar 2018)
13. Java SE 11 (September 2018)
14. Java SE 12 (March 2019)
15. Java SE 13 (September 2019)
16. Java SE 14 (March 2020)
17. Java SE 15 (September 2020)
18. Java SE 16 (March 2021)
19. Java SE 17 (September 2021)
20. Java SE 18 (to be released by March 2022)

Java SE 8 se Java decieds even no. raha to March aur odd raha to September mai updated version released karenge.

Teacher's Signature:.....

PAGE NO. 34
DATE 31/01/22

Features and Advantages of Java

1. Java is Platform independent
→ Means it can run on any Platform like Windows, Linux, Mac.

2. Portable
→ as it can run on any Platform, So its Portable.

3. Java is Secure language.
→ With the Help of 4 access modifier i.e Public, Private, Protected and Default, Developers can hide the code. Also if we want our code should not override then we use final keyword. This adds to the Security of Java.

4. Java is object Oriented Programming language.
→ Means Java is based on the concepts of object, class, Methods, Inheritance, Abstraction, etc. which improves the quality of Software and also Productivit

5. Java is robust in nature.
→ Means Smallest of Smallest exception is a handled which reduces the

Teacher's Signature

Chances of Software Crash in future

6. → It's Simple as Compared to C & C++ Pointers Concept is not Present in Java. Also memory memory management is automatically Handled in Java.

7. Java is free and open Source.

8. It Promises "WORA" which means Write Once Run Anywhere

Disadvantages –

1. Java does not Support multiple inheritanc

2. Verbose and Complex Codes Java Codes are Verbose, meaning that there are many words in it and there are many long and Complex Sentence that are difficult to read and understand

3. Java requires a Significant amount of Space as Compared to other languages like C, C++.

Teacher's Signature.....

Applications or Uses of Java

1. Java is used in Desktop applications.
2. Java is used in Mobile applications
3. Java is used in web applications
4. Java is used in Embedded Systems.
5. Robotics, Gaming.
6. More than 3 billion run on Java

Java Platforms / Editions

1. J2SE (Java to Standard Edition)
   - It is used in Desktop or Windows based application.
   - It includes core topics like OOPs, String, Regex, Exception, Inner classes, Multithreading I/O stream, Networking AWT, Swing, Reflection, Collection, etc.
2. J2EE (Java to Enterprise Edition)
   - It is use to develop web and enterprise application.
3. J2ME (Java to Micro Edition)
   - It is micro platform that is dedicated to mobile applications.

Teacher's Signature.....

JAVA Installation AND Download

PAGE NO. 7
DATE 03/02/22

Download JAVA

1. Search JDK download on Google
2. Select 1st website www.Oracle.com (Kit 17th)
3. Select your OS (Windows / Linux / mac OS)
4. Select * 64 installer link (152 MB file)
5. Download will start

Download ECLIPSE

1. Search Eclipse download on Google
2. Select 1st website www.eclipse.org
3. Select * 86-64 download option (113 MB file)
4. Download will start.

JAVA Installation

1. from Download folder Copy Jdk+ Eclipse File and paste in in any Drive (except C-Drive) in a folder, name that folder as Application.
2. Select Jdk.17 File → right click on mouse → Click RUN As Administer → Popu Screen opens (in Some Cases no Popup open) Press YES → NEXT → NEXT → CLOSE.
3. open C Drive → Program Files → Java → Jdk 17 → Bin

Teacher's Signature.....

Copy the Path after Copy minimize the screen you will return to desktop

4. Right Click on This PC option → Properties → Advanced System Setting → Enviroment Variable

Popup Screen of Enviroment Variable will open as

Double Click on Path in System Variables [in second block]

Select New button [in right side corner] now one more popup Screen open [at the bottom paste and OK]

5. Now again go to C Drive → Program Files → Java → Jdk 17

Copy the Path

6. Go to desktop & Right Click on This PC option → Properties → Advance System setting → Environment Variable.

Popup window will shown Select New button In System Variables

Popup window will open In this Write a Variable name (~~Ex~~: JAVA_HOME)

Teacher's Signature.....

And in Second box right Click & Select Paste option your Copied Path will Paste Here
AND Press OK button

Now Time to Check JAVA got installed or not in the PC, just follow the Process

Go to Desktop Select Search Option on left Hand bottom Corner and Write there Cmd

After that Command Prompt will open

Black Screen will open
Type Java -Version (give Space after typing java) and Press Enter.
(J is small)

After that there will show Version of java installed in your PC

```
C:\Users\Hp> java -Version
java Version "17" 2021-09-14 LTS
Java (TM) SE Runtime Enviroment (build
                                17+35-LTS-2724)
Java Hotspot (TM) 64-Bit Server VM
    (build 17+35-LTS-2724, mixed mode, sharing
C:\Users\Hp>
```

This Was the Installation Process of JAVA

Teacher's Signature.....

PAGE NO. 10
DATE 04/02/22

Installation of Eclipse

1. Before start installing Eclips make sure your Internet Connection is ON.
2. Open Application folder where we saved our Eclipse file
3. Select Eclipse right Click on mouse and the Press RUN AS ADMINISTRATOR
4. Pop up Window will open as shown in Pic. Below

   Select this option (Top first option)
   Click on INSTALL
   Click Accept now button

5. Installation loading will start this Process may take 7 to 8 min time. THen new window open as shown in Pic. below

   Press Launch button
   Click on Launch only

And this is Home Screen of ECLIPSE.
Your Eclipse is Installed Successfully.

Teacher's Signature.....

11
05/02/22

1. PROJECT

→ Project is the first thing which we create in our eclipse IDE.

→ Project is a collection of multiple Packages, multiple classes, source folder, JRE system library etc. We can create as many Packages and classes we want in Project

How to create a Project -

Click on File → New → Java Project → Enter Project name in window → Create a new module -info.file (Select → don't Create)

2. JRE System Library

→ It is the Collection of jar files, Predefined classes and Supporting files

→ It Contains all files with .jar extension.

3. Src Folder

→ Src folder is a Collection of Packages and classes.

→ We can Create multiple Packages and classes inside Src folder.

4. Package

→ Package is a Collection of multiple Classes or group of classes.

→ Multiple Packages can be Created in our Project

Teacher's Signature:.....

→ if Package Contains no class then Symbol is Colourless
→ if Package Contains then its Colourful.

Some important Points while Creating Package :-

1. Package name can be anything.
2. Package name always start with lowercase.
3. If there are 2 words then Second word will be capital Ex. basic_Program -class.
4. Package name does not Contain Space in between.
5. In Special Characters, we can use "_" & "$" Symbol only.
6. Package name cannot start with integer.

How to Create Package :-

Right Click on Src folder → New → Package → Write Package Name → Click Finish.

5. Class

Class is a Collection of member function, Variables, datatypes, Printing Statem -nt Scanning Statement, object, Conditio

Teacher's Signature.....

Statement, loop Statement, Constructor etc.

How to Create a Class :-
Right click on Package → New → Class → Write Class Name → Finish.

SYNTAX of class :-

Public class Class Name
{

} // class body

Rules Related to class :-

1. Your Complete Code will be written inside the Class
2. Class Name should not have Space.
3. First letter of class Cannot be integer value.
4. First letter should always be Capital of every word.
5. Best Practice is to not use keyword as Class Name.
6. To Change the name of the Class.

→ Right click on class name
→ Click Refactor → Rename → Write New Name → Finish

Teacher's Signature.....

PAGE No. 14
DATE 05/02/22

first Program

```
Package myFirstProgram ;
Public Class MyIntroduction {
 Public static void main (string [] args)
 {
   System.out.println ("Name = Gaurav Raut");
   System.out.println ("Education = Bsc (CS) 2021
  System.out.println ("Mobile = 8806747044");
  System.out.println ("Address = Wardha, MH");
  System.out.println ("Gender = Male ");
  System.out.println ("Hobbies = Badminton");

 }

}
```

Output :-

```
Name = Gaurav Raut
Education = BSc (CS) 2021
Mobile = 8806747044
Address = Wardha , MH
Gender = Male
Hobbies = Badminton
```

Teacher's Signature.....

About Main Method

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

Public = Access Specifier
Static = Keyword
Void = return type
main = method name

few important Points of main method :-

- Main method is known as the entry Point to start the execution of our Program.
- It is mandatory to write the main method.
- Without main method we cannot run our Program.
- When we don't write the main method we will not get any error. But when we run it we will get the error.

Various ways to write the main method :-

1. static Public Void main (String [] args)
   → Order of modifier Can be Changed.
2. Public static Void main (String.... args)
3. Public Static Void main (String args [])

Why do we set environmental variables?

* If we don't set enviromental variable How our Program will run.

- Firstly what we do is, we download and install JDK.

- If we write our Program on notepad then file inside the bin folder only. (This is our "majboori" as we have no option because our Path is not set.) After saving in bin folder, now to run the Program also we have to enter the Path till our bin folder in Cmd.
This means we need to follow certain Steps every time which is lengthy or time Consuming.
So the easy way for this is to set the environment variable. After that we can run our Program from anywhere, this save our time and effort.

NOTE :- Open notepad and Cmd by right click run as administrator

Teacher's Signature.....

If we set our enviroment Variable Path then how our Program will run.

As our jdk is installed and we Have set the envionment variables as well.
Write our Program in notepad.
Now Save the program anywhere as per your Choice.
Then Compile and run the Program.
Here no need to right click and select run as administrator for notepad and cmd.

Run Program on Notepad

Write a Simple Program in notepad.

```
Public class Hello1
{
  Public static void main (string [] args)
{
  System.out.Println ("My Name Is ~~It~~ Gaurav Raut"
  }
}
```

Save the Notepad file by ClassName.java i.e name should be same as ClassName
Ex. Hello1.java

Teacher's Signature.....

And also Save this inside G: Drive or any Drive of your choice.

Now open Command ~~Propt~~ Prompt and follow the below Steps to Compile and run:-

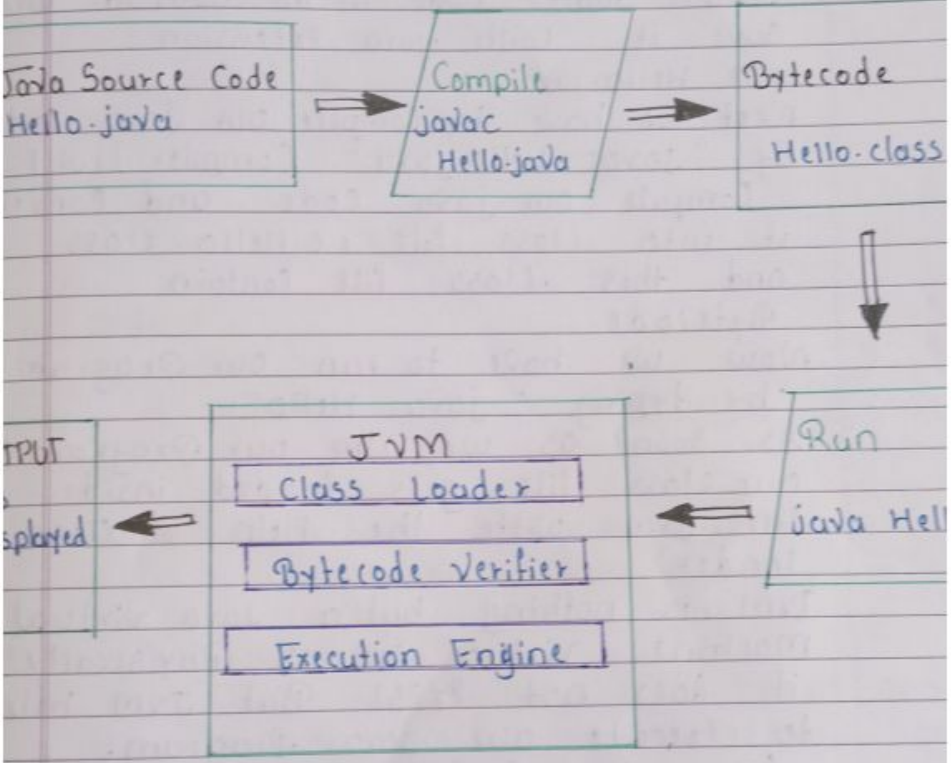
utput

```
Microsoft Windows [Version 10.0.19043.1526]
(c) Microsoft Corporation. All rights reserved.
C:\Users\HP>D:
D:\>javac Hello1.java
D:\>java Hello1
My Name is Gaurav Raut
```

As we Compile the Program then Hello1.class file is created in G: Drive.

Teacher's Signature:

Execution Flow of a Java Program



Lets Write our first Program in ECLIPSE

```
Public class Introduction {
   Public static Void main (String [] args )
  {
     System.out.Println ("Name :");
     System.out.Println ("Gaurav");

   }
}
```

Output :-

Name :
Gaurav

Difference between System.out.Println() and System.out.Println ()

1. System.out.Println()
   -> This will Print the Statement an move the Cursor to the new li

Ex.
```
System.out.Println ("Hello");
System.out.Println ("Friends");
```

output -> Hello
Friends

Teacher's Signature .....

System.out.println()

→ This will print the statement but will not move cursor to the new line.

Ex. System.out.print ("Hello");
System.out.print ("Friends");

output → HelloFriends

SYMBOLS

//

→ This is used to comment out a particular statement.

()

→ This is called as a method signature.

+

→ It is used to end a particula sentence.

;

→ It is used to end a particular sentence

Teacher's Signature.....

\n

→ This means to move to a new line at this specific point.

Program 1

```
Public class Symbols
  {
    Public static void main (String [] args )
 {
  System.out.println ["My Name is" + "RAJ" ];
  System.out.println (" Im from" + "Nagpur" );
   }
}
```

Output :-

My Name is RAJ
Im from Nagpur

Program 2

```
Public class Symbol
{
   Public static void main (String [] args
   {
      System.out.println ( "my name is Raj"
\n" + " from Nagpur " + "\n" + " completed
                B.E " );
   }
}
```

Teacher's Signature.....

Output -

My name is Raj
From Nagpur
Completed B.E

Program 3

```
Public class Symbols
{
  Public Static Void main (String [] args)
{

System.out.Println ("My name is Raj "+"
  \n"+" From Nagpur "+" Completed B.E");

  System.out.Println ("-------");
  System.out.Println("My name is Rahul
  \n From Satara \n Completed M.E" );
    }
}
```

Output :-

My name is Raj
From Nagpur
Completed B.E

My name is Rahul
From Satara
Completed M.E

Teachar's Signature.....

Object

1. Object is an instance of class.
   → instance means "Example"

2. object is an entity of a class.
   → entity means to have existence.

3. object is used to call non-static method.

Syntax

- ClassName objReferenceVar = new Constru
- Ex.
  Test obj = new Test ();

ClassName
→ It will be the name of the class at time of object creation.

objReferenceVar
→ It is used to provide the reference or name of the object.
→ It is always written in minimum words.

Teacher's Signature:

New

→ new is keyword used to create a new object.

Constructor ()

→ It is same as class name
→ It is used to load non-static method or variable into object

Methods (Member Function)

* Methods are declare inside a class body and outside main method.
* Methods name always start with lower case and inner words in Uppercase.
* We can declare multiple methods inside a single class.
* Methods are created to write a group of codes inside it.
* Methods are used to perform code reusability.

Teacher's Signature.....

There are 2 types of Methods

1. Business method i.e Main Methods
2. Regular Method
   -> Static Method
   -> Non-Static Method

* Static Method

1. A static method has a static keyword
2. Static method can be called without creating an object
3. Static method can be called using (className.methodName();)
4. Static method will be created inside class body and outside the main method
5. Method name can be anything as per the requirement.

Syntax of static Method

```
Public static void methodName()
{


}
```

Program

```
Public class Test {
    Public static void myMethod ()
  {
   System.out.Println ("Im inside the
                              method");
    }
   Public static void main (String [] args)
    {
   Public static
      Test. mymethod ();
    }
}
```

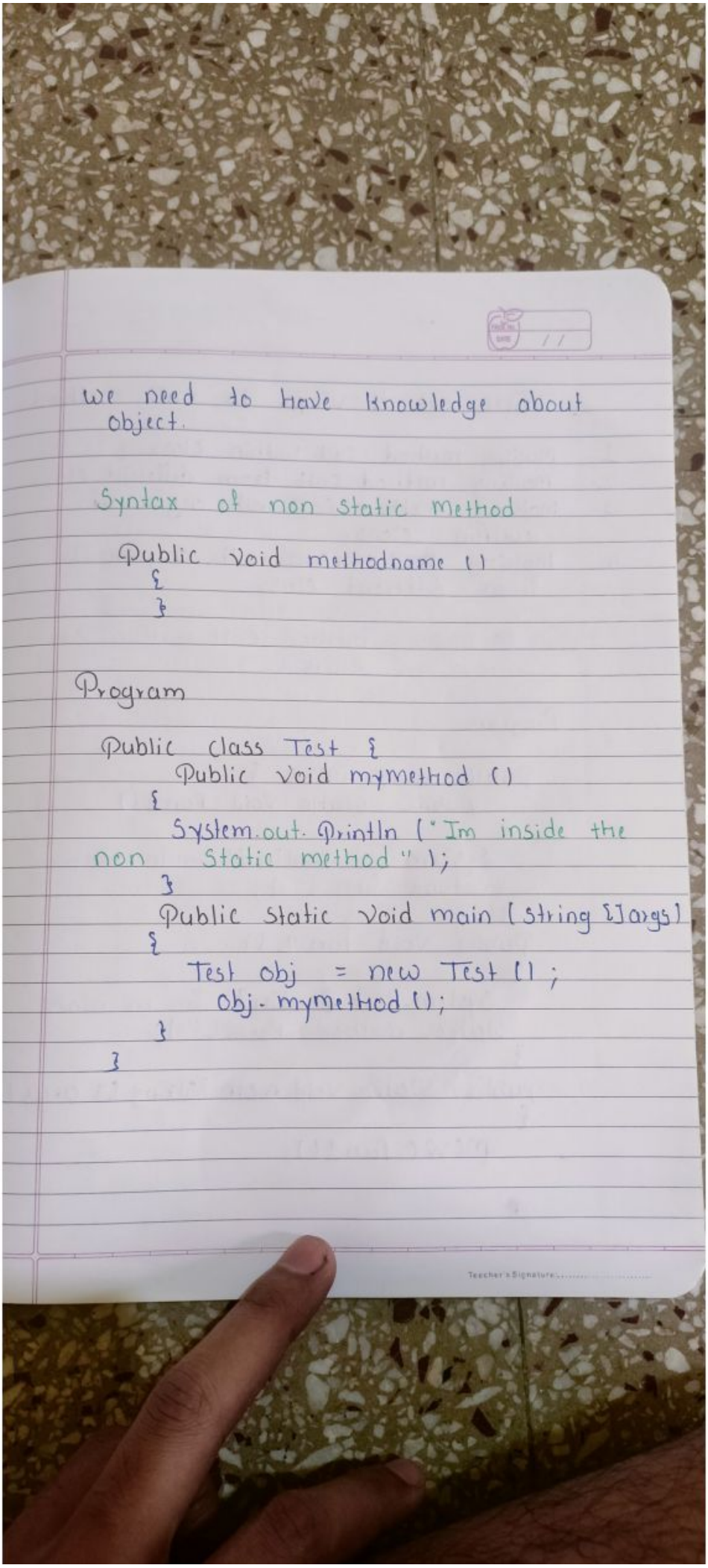
* Non-static Method

1. This method does not contain static keyword.

. We need to create an object of a class for calling non-static method.

It is preferable to create non-static method because it is more secure.
As static method can be easily accessible in main method.
But for non-static method

Teacher's Signature.....



we need to have knowledge about object.

Syntax of non static method

```
Public void methodname ()
  {
  }
```

Program

```
Public class Test {
     Public void mymethod ()
   {
     System.out.Println ("Im inside the
non    Static method" );
    }
    Public static void main (String [] args)
   {
      Test obj  = new Test ();
        obj.mymethod ();
    }
}
```

There are 4 ways to call method.

1. Making method call within class.
2. Making method call from different class.
3. Making method call with arguments within class.
4. Making method call with arguments from different class.

* Making method call within same class

Program

```
Public class MCWC {
    Public static void Fun1()
  {
    System.out.Println("Im in static
    method fun 1");
  }
  Public void fun2()
  {
    System.out.Println("Im in Non
    static method fun 2");
  }
  Public Static Void main(String[] args)
  {
    MCWC.fun1();
```

*

Teacher's Signature.....

```
MCWC obj = new MCWC ();
 obj. fun 2 ();
 }
}
```

Output –

```
Im in static method Fun 1
In in Non static method Fun 2
```

* Making method call from different class

Program

```
public static void cat ()
{

Public class MCDC1 {
 Public static void cat ()
{
  System. out. Println (" Dog eats chicken ");
 }
 Public void drink ()
 {
    System .out. Println (" cat drinks Milk ");
  }
}
```

Teacher's Signature.....

```
Public class MCDC2 {

  Public static void main (String [] args)
   {
      mcdc1 cat ();
      mcdc1 obj = new mcdc1();
       obj . drink ();
   }
}
```

Output -

Dog eats chicken
Cat drinks Milk

* Making method call with arguments within same class

Program

```
Public class MCAWC {
  Public void mobile ()
  {
    System.out.println ("I Have
                  mobile" );
  }
  Public void Computer ( int a)
```

Teacher's Signature.....

```
    {
      System.out.Println("I Have a Computer");
    }
  Public void lapton (String a)
  {
    System.out.Println("I Have a laptop");
  }
  Public static void main(String[] args) {

    MCAWC obj = new MCAWC();
     obj.mobile();
     Obj.Compute(3);
     Obj.laptop("HP");
  }
}
```

Output –

```
I Have mobile
I Have a Computer
I Have a laptop
```

* Making method call with arguments from different class

Program

```
Public class MCADCJ {
  Public void fruits()
```

Teacher's Signature.....

```
{
  System.out.println("I like Mango");
}

Public void Vegetables (String a)
  {
    System.out.Println ("I like Potato");
  }
}

Public class MCADC2 {
  Public static void main (string [] args)
  {
    MCADC1 b = new MCADC1 ();
    b.fruits ();
    b.Vegetables ("Hi");
  }
}
```

Output :-

```
I   like  Mango
I   like  Potato
```

Teacher's Signature:.....

Variables

- In java Programming language Variable is a Piece of memory used to store information
- One Variable can store one information at a time.
- Variable is assigned with a data type.
- Syntax

```
datatype variableName = value ;
```

- There are 3 types of Variables :-

1. Local Variable
2. Instance / Global Variables
3. Static Variable

* Local Variable

1. Local Variable is declared inside the method body / constructor body.
2. Scope of that Variable remains only within the method / constructor body.
3. Local Variable cannot be defined with static keyword.
4. We cannot declare more than one local Variable of same name within method.

5. It does not have any default value in java We need to provide the value, otherwise it gives error.

* Instance / Global Variable

1. Instance variable is declared inside the class body, but outside the method body.
2. It is also known as Global Variable.
3. This variable cannot be accessed in static method.
4. But can be accessed in non static method within a class.
5. Instance variable has a default value. For int its "0", String its "null", Boolean its "false"

* Static Variable [Class Variable]

1. A variable declared with static key-word is called static variable.
2. It is also known as class variable.
3. Declared inside class body but outside method body.

Teacher's Signature.....

4. ~~But can be accessed in non static method within a class.~~

5. Static variable can be accessed in both static as well as non static method.

5. ~~Inst~~ Static variable has a default value. For int its "0", String its "null", Boolean its "false"

Example :-

Program 1

```
Public class Var Example {
  int a = 20;          // Instance Variable
Static int b = 30;     // Stactic Variable
Public void myMethod ()
  {
    int d = 20;        // Local Variable
  }
}
```

① Local Variable

```
Package Variable ;
  Public class Variable_local {


Public static void main (String [] args) {
 Variable_local b = new Variable_local
    b.local variable ();
    local 1();
}
 Public void localvariable () {
      int i = 45;
    System.out.Println (i);
  }
    Public static void local1()
  {
     int i = 50;
    System.out.println (i);
  }
```

Output:- 45
50

Teacher's Signature.....

Q. Write a Program of ~~local~~ global Variable & With multiple output Printing

```
Package feb batch
  Public class GlobalVariables {

        float f = 89.56f;
  Public Static Void main (String[] args)
{       GlobalVariables x = new GlobalVariable();
          x.test();
          x.test();
          x.test();
          x.test();

  }
  Public Void test () {
        System.out.println (f);
    }
  Public Static Void test ()
{
        System.out.println (f);
    }
  }
```

Output:

89.56
89.56
89.56
89.56

Static Variable :-

```
package Variable ;
Public class StaticVariable
{
   Static int m = 100;
   Static int n = 10;
 Public Static Void main (String [Jargs)
 {
    Static Variable S = new StaticVariable();
      S. addition ();
      S. division ();
 }
    Public Void addition () {
        int o = m+n;
      System.out.println (m);
      System. out. println (n);
   }
    Public static Void division () {
        int p = m/n;
      System.out.println (m);
      System. out. println (p);
    }
}
```

Output:- 100
110
10
10

Here, you can not declare variable as static inside method. In otherwords we can say that, Local variable can not be declared static.

Teachar's Signature

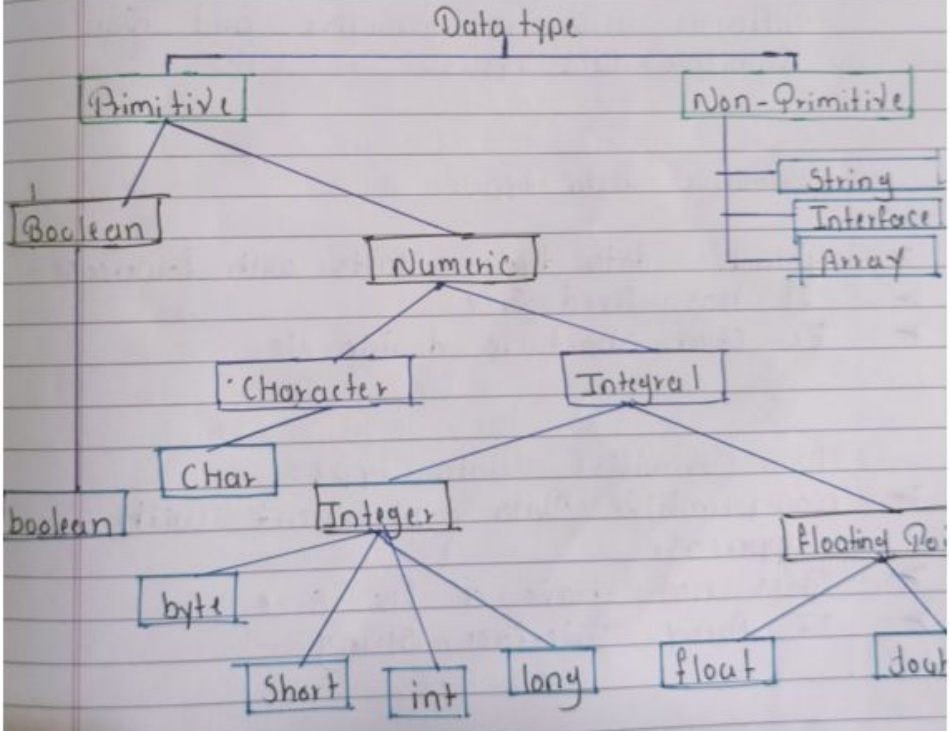
Data Types

As the name suggest data type means type of data.

Data type are use to represent type of data or information which we are going to use in our Program

In java Programing language it mandatory to declare data type before declaration of Variables.

Ex. String name = "Jethalal"



Size:

1] Boolean = 1 bit
2] Char = 2 byte
3] Short = 2 byte
4] int = 4 byte
5] float = 4 byte
6] long = 8 byte
7] double = 8 byte
8] 1 byte = 8 bit

Difference between Primitive and Non Primitive data types.

Primitive Data type:

- Primitve data type starts with lowercase.
- It has fixed size
- Ex. Char, short, in, double, etc.

Non Primitive Data type:

- Non-primitive Data type start with uppercase
- Does not have a fix size.
- Ex. Array, Interface, String

Primitive Data type:

1. Boolean data type:

-> It is used to stored only possible values i.e "true" or "false"
-> Boolean data type specifies 1 bit information
-> Ex. boolean b = false;

2. Byte data type:

-> It is an example of Primitive data type
-> It size is 8 bit
-> The range of this data type is '-128 to -12

3. Short data type:

-> It size is 2 byte
-> A short data type is smaller than 'int'
-> The range between '-32768 to 32767

4. int data type

-> It is commonly used to data type for integers.
-> It size is 4 byte.
-> It range is -2147483648 to 2147483647'
-> Ex int a = 100;

5. long data type

→ its size is 8 byte.
→ it range is between '-9,223,372,036,854,775,
to 9,223,372,036,854,775
→ It is used when range of value is more than 'int'.
→ Ex long i = 334435861 L ;

6. Float data type

→ Its size is 4 byte.
→ Its not used to precise value, it is used for fraction / decimal values.
→ Ex. float f = 223.45 f ;

7. double data type

→ It Size is 8 byte
→ It is use for decimal / fraction values.
→ Ex. double h = 20.4

8. Chara data type.

→ It Size is 2 byte
→ It is used to store a Single cha
→ character must surrounded by sin
→ Ex Char = 'e'

Teacher's Signature:

Non-Primitive Data type:

1. String

-> String is a non-primitive Data type
-> It start with uppercase
-> It does not have any fixed memory
-> If you want to store combination of multiple characters, digit, fraction, numbers, special character then that type of info. we have to stored in string data type with double inverted commas "".

How to decide which data type we have to use?

-> Data type is decide on the basis of infor-ation which we are going to be used in the Program.
Suppose, if information contains digit we co use int, short, etc.

Q. Write a Program in Java by using an datatype in Printing.

```
Package Ddatatype
    Public class VariableDataType {

    Public Static Void main (String [] args)
    {
        // Data type Variable = Value

        boolean H = false;
        byte e = -128;
        Short f = 32167;
        int i = 21346789;
        long m = 12345678910111L;
        float g = 35.463;
        double U = 779.009;
        char K = X;

    System.out.Println (K);
    System.out.println (H);
    System.out.println (e);
    System.out.println (f);
    System.out.println (i);
    System.out.println (m);
    System.out.println (g);
    System.out.Println (U);

    }

}
```

Teacher's Signature:

output :- x
False
-128
32167
21346789
12345678911011
35.463
779.009

Teacher's Signature

Constructor

- Constructor block similar to method.
- Constructor name is same as class name.
- Constructor does not have any return type.

Ways to Call a Constructor

- By creating object of class
  Constructor executes automatically when we create a object.
- By using 'new' keyword with Constructor -Name with method Signature followed by semicolon. i.e. new ConstructorName();

Why we need to Use Constructor ?

- Constructor is used to initialize the data members i.e variable of the class.
- Constructor is used to load a non-static method or variable into object.
- Also, whenever we want a piece of code to run as soon as object is created we write it inside Constructor.

Teacher's Signature:

Program

```
Public class ClassConstructor {
    // 1. Variable Declaration
    // 2. Variable initialization
    // 3. Usage

    String name ;        // Declaration
    int    age ;

    ClassConstructor ()
{
    name = "Gaurav" ;    //
    age  = 25 ;

System.out.println ("Name is "+name+" and
                    age is "+age) ;

    }
    Public Static Void main (String [] args ) {
    new ClassConstructor ();
    }
}
```

Output :-

name is Gaurav and age is 25

There are 2 types of constructor

Default Constructor
User defined Constructor

- zero Argument / Non Argumen / Non Parametarized Constructor
- Parameterized / Argument Constructor

1] Default Constructor :-
~~User~~ t → According to java each class must have a constructor, if no constructor is present then at time of compilation compiler will create a default constructor for the class and this default constructor will have a blank body

2] User defined Constructor
→ If Programmer is declaring the constructor in class then it is known as User defined Constructor
User defined Constructor is classified into 2 types.

1] Zero Argument Constructor
As the name says, A Constructor with no arguments is called as zero argument Constructor.

Teacher's Signature.....

2] Parameterized Constructor

A constructor is called Parameterized Constructor when it accepts a specific number of parameters

Program:-

```
Public class ConstructorTypes {

ConstructorTypes ()
{
System.out.Println ("zero Argument
                    Constructor");

}
ConstructorTypes (int a)
{
System.out.println ("1 Argument Constructor
                    type int");

}
ConstructorTypes (string s)
{
System.out.println("1 Argument Constructor
                    type String");

}

Public static void main (String [] args) {

   new ConstructorTypes ();
   new ConstructorTypes (5);
   new ConstructorTypes ("India");

}
}
```

Output
zero Argument Constructor
1 Argument Constructor type int
1 Argument Constructor type String

NOTE :-
Multiple Constructor can be declared inside the class but the argument passed should be different

This keyword

* This is a keyword refers to Current class instance variable in a method or Constructor.
* This keyword is used to solve the naming conflict.
This keyword is used to invoke Current class method.
  - If we don't add this keyword in the Program, Compiler will automatically add this keyword while invoking the method.
* This keyword is used to invoke Current class Constructor.

Operators

* Operators in Java :- Operator is a Symbol which is use to Perform operations.

There operators are used to manipulate variables.
There are many types of operator are given below.

1] Unary Operator
2] Arithmatic Operator
3] Shift operator
4] Relation operator
5] Bitwise operator
6] logical operator
7] Ternary operator
8] Assignment operator

① Unary operator :- Postfix - expr++, expr--
Prefix - ++expr, --expr

② Arithmatic operator :- multiplication * / %.
Addition + -

③ Shift operator :- Left Shift <<
right Shif >>

④ Bitwise operator :- bitwise AND &
bitwise inclusive OR |
bitwise exclusive ^

⑤ Logical operator :- Logical AND &&
Logical OR ||

⑥ Relation operator :- Comparision < > <= => instanc
Equality == != (equal to, not equal to)

⑦ Assignment operator :- Assignment = += -= *= /= %= ^= |= <<= >>=

⑧ Ternary Operator :- Ternary ? :

Unary operator :- In the Java Unary operator require only one operand.
- Unary operator are used to perform various operations.

1] Incrementing / decrementing a value by one
2] negative an expression
3] Inverting the value of a boolean.

Example (Postfix :- exp++ , exp--)
[We can write Prog. insi non static]

// Postfix → Package basicProg;
Public class UnaryOperator {

Public static void main (String [] args
{
int i = 10;
System.out.println (i); //10

Teacher's Signature

```
        System.out.Println (i++);  // 10
        System.out.println (i);    // 11
    }
}
```

Output :- 10
10
11

// Postfix (exp--) → int 10;

```
System.out.Println (i);    // 10
System.out.Println (i--);  // 10 को print
System.out.Println (i);    // 9
```

Output :- 10
10
9

Example -

```
Package basicProg;
Public class UnaryOperator {

    Public static void main (String[] ar
        int i = 10;
// Postfix e++  System.out.Println (i);    // 10
        System.out.println (i++);   // 10
        System.out.println (j++);   // 11
        System.out.println (j++);   // 12
// Postfix (exp--)  int j = 15;
```

Expression :-

```
Public class Operator {
Public Static void main(String[]
                              args){
System.out.println(10*10/5+3-1*4/2);
```

output :- 21

{ 10*10/5+3-1*4/2 = 10*2+3-1*2
= 20+3-2
= 21

Logical AND & Bitwise AND

* Logical AND (&&) →

i) Doesn't check second condition if first condition is false.
ii) It check Sencond condition if first condition is true

Use - Logical AND

0	0	0
0	1	0
1	0	0
1	1	1

* Bitwise AND (&) →
- Bitwise & operator always checks both conditions whether first condition is true or false.

Example:-
```
Package basicProg;
Public class Operator {
    Public static void main (String[] arg
    {
        // Logical AND (&&)
        int a = 50;
        int b = 10;
// true && false = F System.out.println (b<a && b>a);
// false && true = F System.out.println (a<b && b<a);
```
Logical

```
        // - Bitwise AND (&)

// false & true = F System.out.Println (a<b & b>a);
// false & true = F System.out.println (b>a & a>b);
// true & false = F System.out.Println (a>b & b>a);
// true & true = T System.out.println (b<a & a>b);

    }
}
```

Output :-
```
False
False
false
false
false
True
```

Teacher's Signature

* Write a Program with Logical && using boolean value & integer value?

```
Package basicProg;
Public Class Program
{
    Public static void main (String [] args)
    {
        boolean num1 = true;
        boolean num2 = False;

        int a = 10
        int b = 20
        int c = 0

System.out.println (" num1 && num2 );
System.out.println( c>b && a>b);
System.out.println( b>c && a<b);
System.out.println ( a<c && b>a);

}

}
```

Output :- False
True
False

Example:-

```
Public class Operator {
Public static void main (String [] args)
{
    int a = 10;
    int b = 5;
    int c = 20;

System.out.println (a<b && a++<c);
 System.out.println (a);
        // ... Bitwise AND (&)
System.out.println (a++<c & b<a);
 System.out.println (a);

 }
}
```

Output:- False
10
11

5] Logical OR (||) and Bitwise OR (|)

Use (OR) →

0	0	0
0	1	1
1	0	1
1	1	1

Teacher's Signature

* Logical OR (||)
  i) doesn't check second condition, if first is true.
  ii) It checks second condition only if first is false.

* Bitwise OR (|)
  - Operator always checks both conditions whether first condition is true or false.

Example:-
```
Package basicProg;
Public Class Operator {
  Public Static Void main (String [] args)
  {
      int a = 10;
      int b = 5;
      int c = 20;
// T   System.out.println ( c>b || b>a);
// T   System.out.println ( a<b || b<a);
          // ... Bitwise OR (|)

// T   System.out.println ( a>b | a<c);
// T   System.out.println ( a>b | b>c);

  }
}
```

Output:-
```
true
true
true
true
```

Control Statements

⇒ As the name suggests control statement means which controls the flow of the program.

⇒ A java program executes from top to bottom but if we want to control the order of execution of our program based on the logic applied, we use control statement.

Various types of Control Statement

* Control Statements

= Conditional Statements

1. If Statement
2. If else
3. If else - if
4. Nested if
5. Nested if - else
6. Switch

= Looping Statements

1. for loop
2. While loop
3. do While loop

= Jumping Statements

1. break
2. Continue

If Statement

If Statement is used to test the condition It will execute if blocks only when condition is true.

Syntax

```
if (condition)   // will execute when
{                   condition is true
    // code
}
```

Example :-

```
Public class IfStatementUse
{
    Public static void main(String [] args)
    {
        int i = 10;
        if ( i >= 9 )        -> Condition
```

first checks condition is true or correct

```
        {
            System.out.println("I is Greater than 4");
        }
    }
}
```

Output :-

I is Greater than 4

Example 2 :-

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

    int num = 5
    if (num > 0)
        {
            System.out.println("No. greater than 0");
        }
    System.out.println("End of the Code");
    }
}
```

Output :-

No. greater than 0
End of the Code

If Else Statement

In If Else statement, if the specified condition in the if statement is true it will execute if block otherwise else block will be executed.

~~if~~ if statement ~~false~~ condition false ~~else~~ else block execute ~~if~~ ~~true~~ true ~~else~~ if ~~false~~

Syntax

```
if (condition) // when condition is
{                    true it will execute
   // code
}
else                 // when (if block condition
{                       is false it will execute
   // code
}
```

```
Public class IfElseConditionUse {

Public static void main (String [] args)
{
   if int i = 17;
      if ( i <= 9 )
   {
      System.out.Println ("I is not Gre
                           than 9");
   }
```

```
        else
            {
                System.out.println("I is greater
                                    than 9");
            }
    }
}
```

Output:-

I is ~~Gr~~ greater than 9

Example 2:-

```
Public class IfElse {

Public static void main (String [] args) {

        int num = 10;
        if (num >0)
      {
       System.out. Println ("Number is
                    Greater than zero");
       }
      else
       {
       System.out. Println ("Number is
                    less than zero");
       }

  }
}
```

Output :- Number is greater than zero

If Else - If Statement

The if else-if statement contains if statement followed by multiple else-if statements

In if else-if statement, suppose if condition is true, then if statement will be executed and the remaining code will be skipped.

But suppose no condition are true then else block will be executed

जर कोणती if मधे condition false असेल तर else block execute होईल. जर कोणताही जर एक condition If ची True असेल तर बाकी skip होतील.

Syntax :-

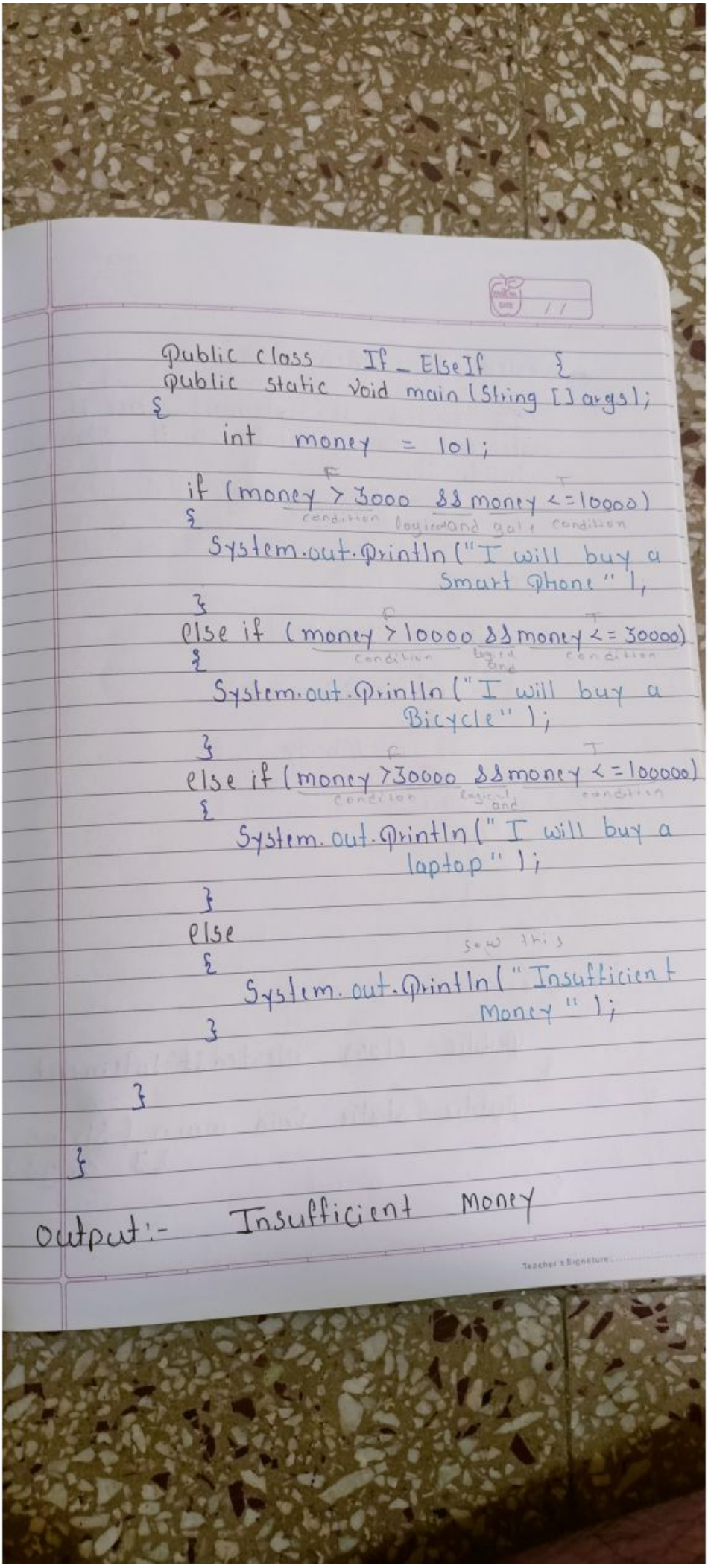
```
if (condition 1)
{
    // code
}
else if (condition 2)
{
    // code
}
else
{
    // code
}
```

Example :-

```
Public class IfElseIfStatement
 {
 Public static void main (String [] args)
{
    int a = 5;
    int b = 9;

 if (a > b)
  {
    System.out.Println(" condition first
                          is true");
  }
 if Else ( a<0)
   {
    System.out.Println(" else if Condition
                          is true");
   }
 Else
     {
     System.out.Println(" Both Condition
                  is false So Else block
                          will execute");
     }
}

}
```

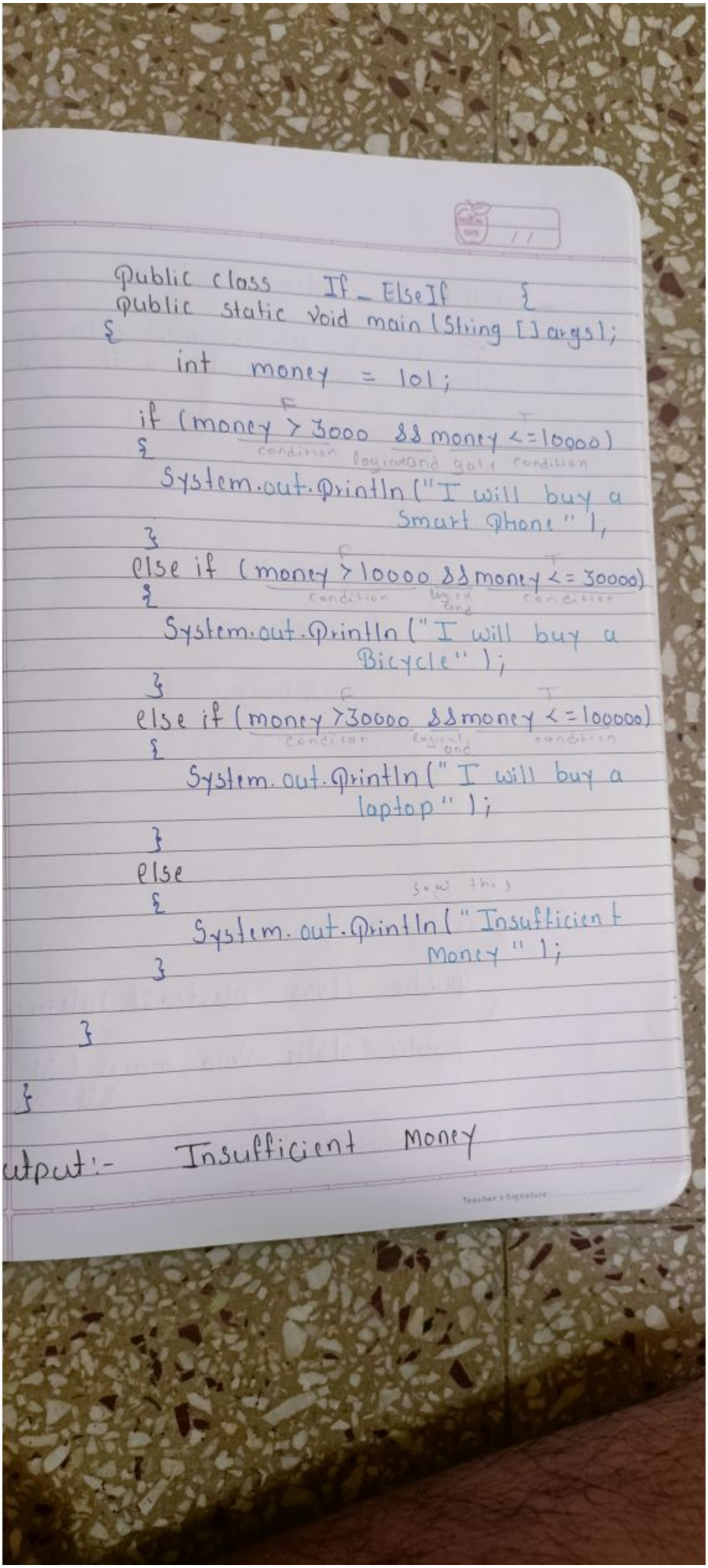
Output :- Both Condition is false So Else block will execute.

Teacher's Signature:.....



```
Public class If_ElseIf {
Public static void main(String [] args);
{
    int money = 101;

  if (money > 3000 && money <=10000)
  {
    System.out.println("I will buy a
                        Smart Phone");
  }
  else if (money > 10000 && money <= 30000)
  {
    System.out.println("I will buy a
                        Bicycle");
  }
  else if (money > 30000 && money <= 100000)
  {
     System.out.println("I will buy a
                         laptop");
  }
  else
  {
     System.out.println("Insufficient
                         Money");
  }

}
```

Output:- Insufficient Money



```
Public class If_ElseIf {
Public static void main(String []args);
{
    int money = 101;

if (money > 3000 && money <=10000)
{
  System.out.println("I will buy a
                    Smart Phone");
}
else if (money > 10000 && money <= 30000)
{
  System.out.println("I will buy a
                    Bicycle");
}
else if (money > 30000 && money <= 100000)
{
   System.out.println("I will buy a
                    laptop");
}
else
{
   System.out.println("Insufficient
                    Money");
}

}

}
```

utput:- Insufficient Money

Nested If Statement

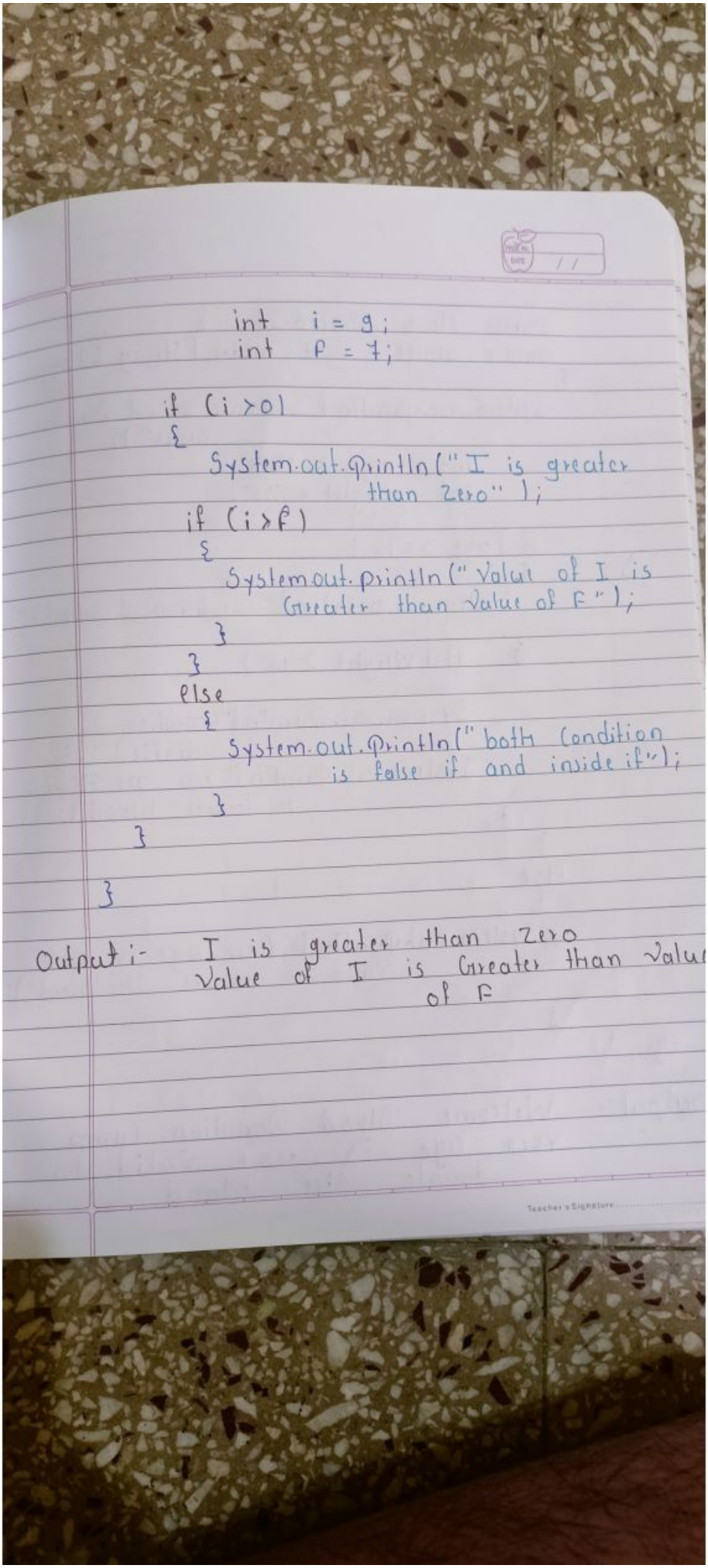
In nested if-statement, the if statement can contain a if statement inside it. (nested if म्हणजे if च्या अंदर if statement असणे) if आणि त्या मधल्या दोन्ही if barobar असतील तेव्हा तो inside if execute नाहि तर else

Syntax

```
if ( condition 1 )
{
    if( condition 2 )
    {
        // body
    }
}
else
{
}
```

Example :-

```
Public class NestedIfStatement
{
Public static Void main ( String [] args )
{
```



```
int  i = 9;
int  f = 7;

if (i > 0)
{
   System.out.println("I is greater
                than zero");
  if (i > f)
   {
     System.out.println("Value of I is
        Greater than value of F");
     }
  }
 else
   {
     System.out.println("both condition
          is false if and inside if");
     }
 }

}
```

Output :- I is greater than zero
Value of I is Greater than value of F

Example 2:-

```
Public Class Nested IF {
Public static void main (String [] args)
{
 System.out.Println ("Welcome Blood Donation
                      Camp");
      int  age = 14;
      int  weight = 35;

  if (age >= 18)
  {
    System.out.println(" Condition 1 satisfied

     if (Weight >45)
      {
        System.out.Println("Condition 2
                            satisfied");
        System.out.Println("You are Eligible
                            to donate blood");
      }
  }
 else
  {
   System.out.Println("Your age is not
                       valid to donate the blood"

  }
} }
```

Output:- Welcome Blood Donation Camp
your age is not valid to donate the blood.

Nested If-else Statement

In Nested if Statement, the if Statement Contain a if-else Statement inside it.

If च्या आतमध्ये If आणि Else statement
↳ and main If ला Else देतो
जर If ची condition barobar असेल तर तो आत येतो. नाहीतर Else print करतो

Syntax:-

```
if ((condition 1)
{
      if (condition 2)
      {
            // code
      }
      else
      {

      }
}
else
{

}
```

Example:-

```
Public class NestedIfElseStatement
Public static void main (String [] args
```

```
int a = 3;
int b = 9;
int c = 6;

if (c<b)
{
  System.out.Println (" we are if ");

   if (a<b)
   {
    System.out.Println(" we are inside
                    if statement");
   }
   else
   {
     System.outPrintln(" both statement
                          are false");
   }

}
else
 {
   System.out.Println(" if condition is
                        wrong");
 }

}
```

Output :- We are if
We are inside if statement

Example 2 :-

```
Public class Nested If Else
Public static Void main (String [] args)
 {
  System.out.println ("Even, odd or Negative");

    int   a = -7;
    if (a>0)
  {
   System.out.println ("Yes number is Positive");

    if ( a %2 = = 0)
      {
         System.out.println("Yes number is
                              even number");

    else
      {
         System.out.println ("Number is odd");

      }
  }
  else
    {
      System.out.println (" It's a Negative
                           number");

    }
 }

}
```

Output :- It's a Negative number

Switch Case Statement

In Java Switch Case statement Contains multiple block of code called as cases and a single case is executed based on the variable which being switched

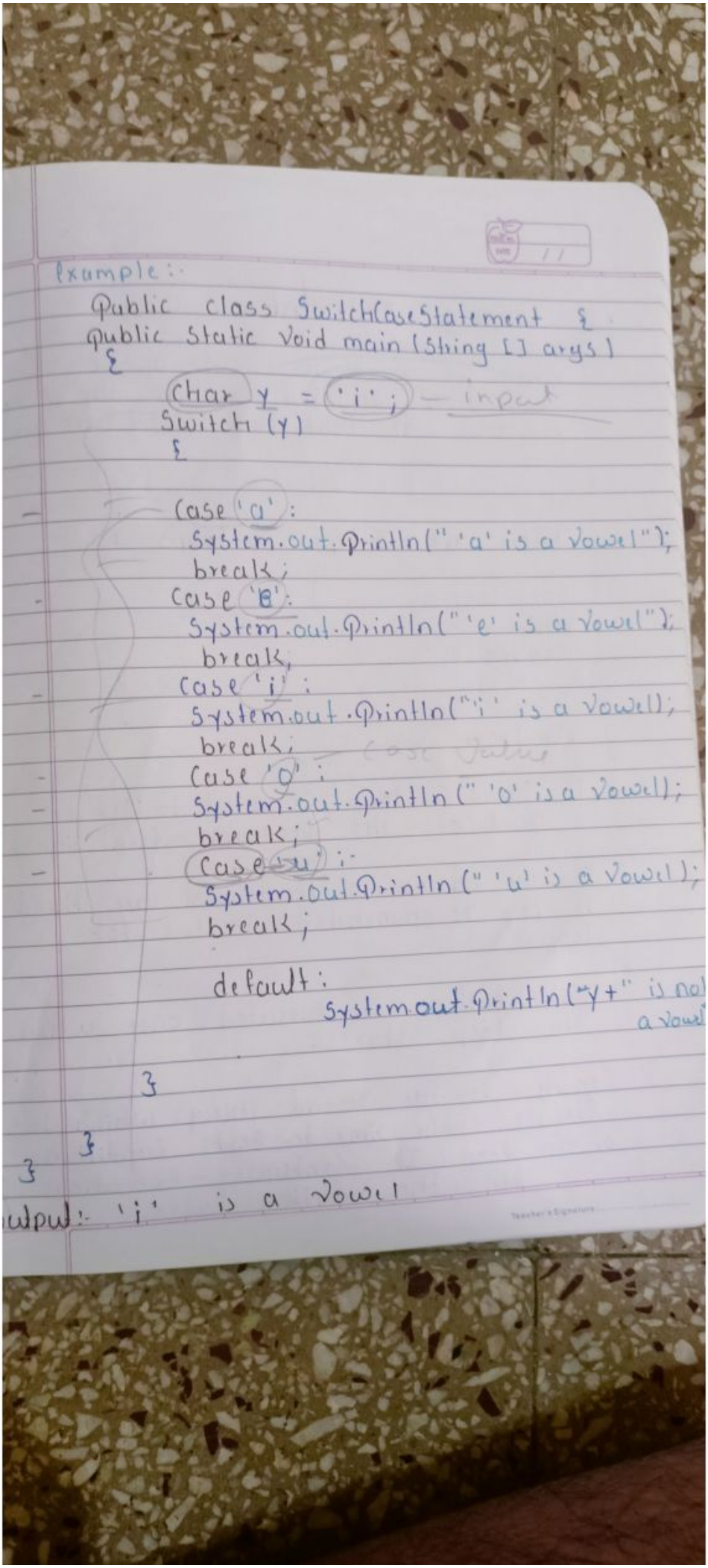
In Switch statement we are going to use only one input

Switch statement works with int, string, char etc.

The case value must be unique, it should not be duplicate.

Each case statement can have a break statement to terminate the code once condition is satisfied. And it is optional, if not used next case will be executed

While using Switch statement, we must see that the case expression will be of the same type as variable.



Example:-

```
Public class SwitchCaseStatement {
public Static Void main (String [] args)
 {
      char y = 'i'; - input
     Switch (y)
      {

     case 'a':
       System.out.println(" 'a' is a Vowel");
       break;
     case 'e':
       System.out.println(" 'e' is a Vowel");
       break;
     case 'i':
       System.out.println("'i' is a Vowel");
       break;
     case 'o':
       System.out.println (" 'o' is a Vowel);
       break;
     case 'u':
       System.out.println (" 'u' is a Vowel);
       break;

       default:
            System.out.println("y+" is not
                                a Vowel
     }
  }
}
```

utput:- 'i' is a Vowel

Looping Statement

When we want to execute a set of instruction repeatedly when the condition is true at that time loops are used

There are 3 types of loops

For loop
While loop
Do-While

For loop

In java for loop is used to iterate a part of the program several times.

If the number of iterations are fixed it is recommended to use for loop.

Initialization is executed once when the loop starts.

Condition is the second thing which is execute each time to test condition of the loop. It continues execution till the condition false.

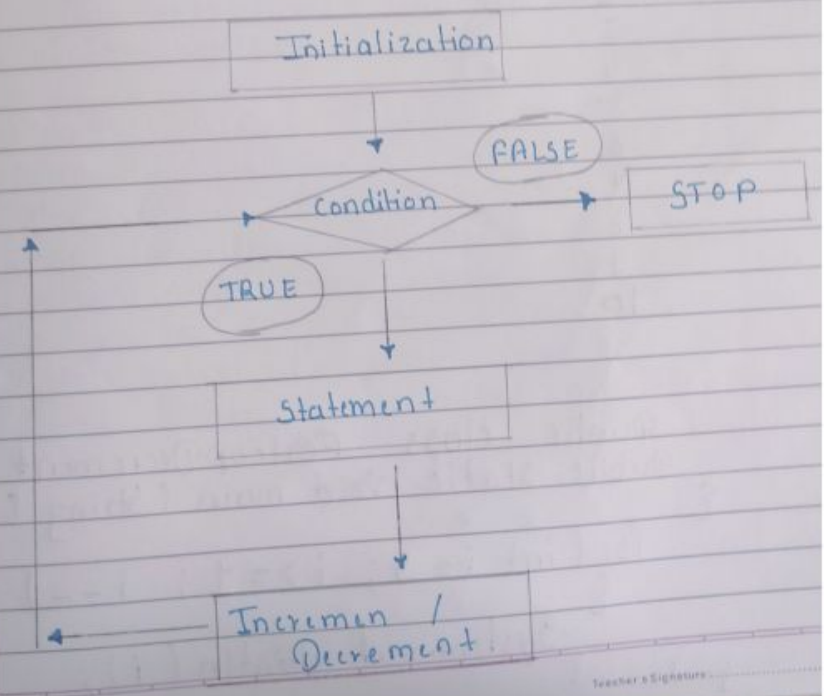
Teacher's Signature:

for loop enables us to initalize the loop variable, check the condition and increment / decrement in a single of code.

Syntax :-

```
for (initialization; condition; increment / decrement)
{
    // code
}
```

Flowchart :-



Example :- Using loop Print 1 to 10 Serial no.

```
Public class ForLoop {
Public static void main (String [] args)
 {
    for(int i = 1; i <= 10; i++)
    {
      System.out.println(i);
    }
  }
}
```

initialization condition increment

Output :
1
2
3
4
5
6
7
8
9
10

```
Public class ForLoopDecrement
Public static void main (String [] arg
{
   for (int i = 5; i >= 1; i--)
     {
       System.out.println (i);
} } }
```

Teacher's Signature.....

output :- 5
4
3
2
1

Nested for loop :-

- In for loop, we are iterating loop with only one variable 'i' but Incase, I have two variables 'i' & 'j' then we are going to uses nester for loop

- Basically nested for loop means, for loop inside another for loop

Here 1st check outer loop condition. Condition is true then goes into the inner for loop. then inner for loop execute until condition because false and then again goes into the outer for loop this process continuous upto outer for loop condition become false then terminate the for loop.

Syntax

```
for (initialization; condition; incr/dec)
  {
    for (initialization; condition; incr/dec)
    {
        statements;
    }
  }
```

Example :-

```
Public class NestedForLoop {
Public static void main (String [] args)
{
  for(int i = 1; i<=5 ; i++)
    {
      for(int j=1; j<=i ; j++)
    { System.out.println("*");
      }
      System.out.println(" ");
    }
}
}
```

Output :-

```
*
* * *
* * *
* * * *
* * * * *
```

While loop :-

While loop is also execute set of instruction repeatedly but condition to be true

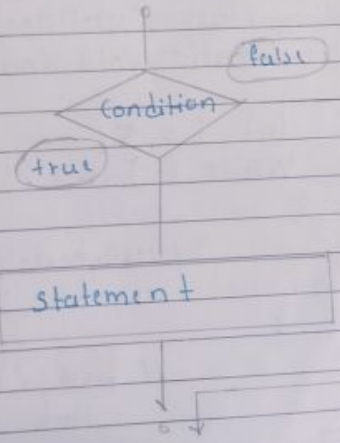
Here no. of iteration is not fixed

Here, firstly checks condition & while loop blocks executed.

Syntax :-

```
while (condition)
{
    statement ();
}
```

Flowchart :-



Example. Print 1 to 10 using While loop

```
Public class Whileloop
Public static void main (String [] args)
{
    int i = 1;
    While (i <= 5)
    {
        System.out.Println (i);
        i++;
    }
}
}
```

Output :- 1
2
3
4
5

(2) i--

```
Public class Whileloop
Public static void main (String [] args)
{
    int i = 5;
    While (1 <= i)
    {
        System.out.Println (i);
        i--;
    }
}
}
```

// First Check Condition then execute while Block

Output :- 5
4
3
2
1

Infinite While loop :

When no. of iteration is not fixed then the particular loop go into the infinite loop

→
```
While (true)  <- condition
{
    Syst Statement (*);   <- infinite loop
}
```

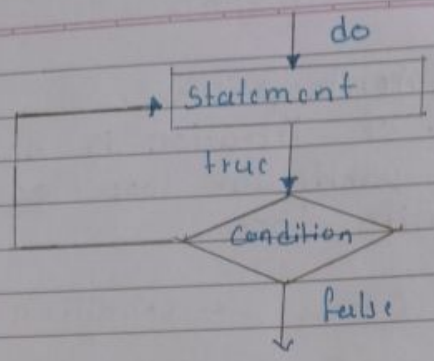
output :-
```
infinite loop
-11-
-11-
-11-
Terminate
```

Do - While loop

- Here no. of Iterations is not fixed but you must have to execute the loop at least once.
- or we can say, the java do-while loop is executed atleast once because condition is checked after loop body means firstly execute do with body & then check condition.

Syntax :-
```
do
{
    Statemen ();
}
while (condition)
```

Teacher's Signature



Example:

```
Public class DoWhileloop
{
  Public static Void main(String[]args)
{
     int i = 1;
    do
      {
       System.out.Println(i);
      }      i++;
      }
     While(i<=5);
  }
}
```

// Here, execute first do block & then check condition.

Output:-

1
2
3
4
5

Logical Programs (Imp)

StarDesignPatternRightAngled.

```
Public class StarRightAngled
{
 Public Static Void main (String [] Args)
 {
     for (int i=1; i<=5; i++) // 1<=5
      {                       // 2<=5 // 3<=5 // 4<=5
       for(int j=1; j<=i; j++) // 5<=5
       {                       // 1<=1
        System.out.Print ("*");
       }
       System.out.println(" ");
      }
 }
}
```

// 2<=1 SecondRound // 1<=2
// 2<=2 // 3<=2 trid Round // 1<=3
// 2<=3 // 3<=3 // 4<=3 fourthround
// 1<=4 // 2<=4 // 3<=4 // 4<=5 // 5<=4
// 1<=5 // 2<=5 // 3<=5 // 4<=5 // 5<=5
6<=5

Output:-

```
*
* *
* * *
* * * *
* * * * *
```

Logical Program (Imp)

StarDesignPattern Opposite Right Angle

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

  for (int i=5; i>=1; i--)
{ // 5>=1  // 4>=1
```

(ii) 2nd time (i) rotation

```
    for (int j=1; j<=i; j++)
{ // 1<=5 / 2<=5 /3<=5 /4<=5 /5<=5
  // 1<=4

  System.out.print("*");

  }

  System.out.println(" ");

  }
}
```

Output :-

```
* * * * *
* * * *
* * *
* *
*
```

Logical Program (Imp)

Check for Prime No.
check 5 is a Prime No. or Not

```
Public class CheckforPrime No. {
Public static void main (String[] args)
{
    int a = 5;
    int temp = 0;

    for (int i = 2; i<=a-1; i++)
    {   //2<=4  //3<=4  //4<=4  //5<=4
                                  X

        if (a%i == 0) //5%2  2)5   5%3
        {                 4
            temp = 1;  not (1)  3)5 X
                              3
                              2
                      4)5
                        4 X
                        1
        }

    }

    if (temp == 0)
    {
        System.out.Println(" a+" is a Prime no.")
    }
    else
     {
        System.out.Println( a+" is not a Prime
     }
  }
}
```

output:- 5 is a Prime No.

Logical Program (Imp)

Print Prime Number

```
Public class ForPrimeno
{
Public static void main (String[] arg
{
  int temp = 0;

  For (int i=2; i<=50; i++)
  {
   For (int j=2; j<i; j++)

   {
       if (i%j == 0)
     {

       temp = 1;

      }

  }
if (temp == 0)
{
  System.out.Println (i+" is a Prim
                          no." );

}
```

Teacher's Signature.....

```
else
 {
        temp = 0;
  }
 }
}
```

Output :-

```
2  is  a  Prime  no.
3  is  a  Prime  no.
5  is  a  Prime  no.
7  is  a  Prime  no.
11 is  a  Prime  no.
13 is  a  Prime  no.
17 is  a  Prime  no.
19 is  a  Prime  no.
23 is  a  Prime  no.
29 is  a  Prime  no.
31 is  a  Prime  no.
37 is  a  Prime  no.
41 is  a  Prime  no.
43 is  a  Prime  no.
47 is  a  Prime  no.
```

Logical Program (Imp)

Fibonacci Series

```
Public class FibonacciSeries
{
public static void main (String [] args)
{
    System.out.println (" Fibonacci Series");
      int a = 0;
      int b = 1;
      int c = 0;

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

   While ( c < 40)
{          // 0+1=1  // 1+1=2  // 1+2=
   c = a + b ;

   a = b ; // b=1
   b = c ;

System.out.print ( " "+c+" ");
   }
}
}
```

Output:- Fibonacci Series

0 1 1 2 3 5 8 13 21

Logical Program (Imp)

Swapping two variable without using third variable

```
Public class SwappingWithoutThird
{
Public static void main (String [] args)
{
    int a = 10;
    int b = 20;

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

    a = a+b ;     // 10+20 = 30
    b = a-b ;     // 30-20 = 10
    a = a-b ;     // 30-10 = 20

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

}
}
```

Output :-

```
10 Swapping 20
20 ---- 10
```

Teacher's Signature

Logical Program (Imp)

Swapping two Variable ~~using~~ Using third Variable (Helping Variable)

```
Public class SwappingWithThird
{
Public Static Void main (String [] args)
{
        int a = 10;
        int b = 20;
        int c; // (as a Helping Variable)

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

        c = b;      // c = 20
        b = a;      // b = 10
        a = c;      // a = 20

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

}
}
```

output :- 10 Swapping 20
20 ---- 10

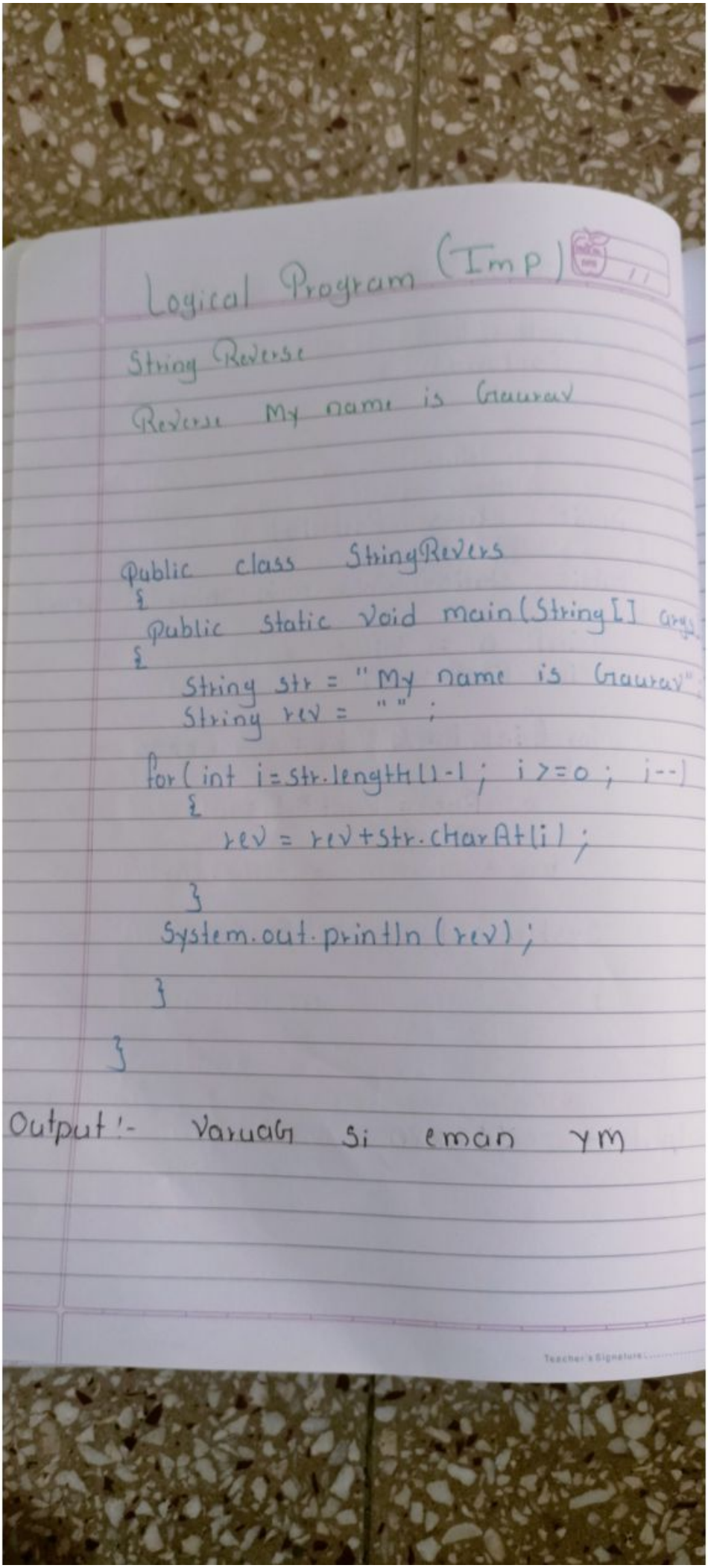
Logical Program (Imp)

Factorial

Find Factorial 9.

```
Public class Factorial
{
Public static void main(String[] args)
{
    int a = 9;
    int fact = 1;

    for (int i=1; i<=a; i++)
    {
        fac = fact * i;
    }

    System.out.println(Fact);

    }

}
```

Output :- 362880



Logical Program (Imp)

String Reverse

Reverse My name is Gaurav

```
Public class StringRevers
{
 Public static void main(String[] args)
{
    String str = "My name is Gaurav";
    String rev = "";

 for(int i=str.length()-1; i>=0; i--)
   {
      rev = rev+str.charAt(i);

   }
  System.out.println(rev);
  }

}
```

Output :- Varuag si eman yM

OPP's (object Oriented Programming System)

- Opp's is a methodology. or we can say it that is a method used for designing a program with class and object. Opp is also called Core java.

- Opp Concept in java is to improve code readability and reusability.

- Main Principle / Pillars of Opp's are:

-> Inheritance

-> Polymorphism

-> Abstraction

-> Encapsulation

-> Inteface

-> Casting / Conversion

Inheritance:-

- It is a mechanism in which one class acquires the all the property of another class with the Help of extends keywords.

- Also in other words we can say that Inheritance means acquiring super class Properties in sub class with the Help of extends keyword is called inheritance

- Super class and Sub class

1. In Java, the super class is also known as the Parent class, it is the class from which a child class (or a sub class) inherits its properties

2. In Java the Subclass inherits Properties from super class. Also a SubClass is a class that extends another clas

Types of inheritance are :-

1] Single-level inheritance
2] Multi-level inheritance
3] Hierarchical inheritance
#
4] Multiple inheritance ∴ Not Support Java.

- Single - Level Inheritance

1. Single-level Inheritance Happens between two classes.

2. In this Subclass acquires all the Property of Superclass using Extends Keyword.

- Multi - Level Inheritance

1. Multi-level Inheritance Happance between three or more classes.

2. In Multi-level Inheritance one Subclass acquires all Property of another Superclass and this Phenomenon continues this is known as multi-level inheritance

- Hierarchical Inheritance

1. In this one Superclass can serve its Properties to multiple Subclass

2. When two or more Classes inherits Properties from a Single class it is known as Hierachical Inheritance.

Teacher's Signature

3. Also multiple Single level inheritance is called Hierarchical inheritance.

Polymorphism :-

- Polymorphism is derived from 2 Greek words, Poly and morphs.
- Here Poly means many and morphs means forms.
- There are 2 types of polymorphism in java

1. Compile time polymorphism
2. Run time polymorphism.

Compile time Polymorphism :-

- In this method declaration is going t be get binded with method defini at Compile time based on arguments.
- As binding take place at compi time so it is called as early bindin

- It is also called as Static binding
- Best example of compile time Polymorphism is Method Overloading.

* Method Overloading

1. Method overloading happens in the same class.
2. In this we can declare multiple methods having the same method name but different arguments.
3. Different arguments means (type different, number different, and sequence different).
4. We can use both static as well as non static method.

Program:-

```
Public class MethodOverloadin
{
  Public void multiply (int a, in
  {
    System.out.println( a*b);
  }
```

```
Public void multiply(int d, int e, int c)
{
    System.out.println(d*e*c);
}
Public void multiply(double d, int e,
                     int c)
{   System.out.println(d*e*c);
}

Public static void main(String[] args)
{
    MethodOverloading m = new
                          MethodOverloading();
    m.multiply(2,4);
    m.multiply(2,3,4);
    m.multiply(4.0,5,6);
}
}
```

Run time Polymorphism

- In this method declaration gets binded with method defination during run time based on object creation is called run time Polymorphism.

- As a binding takes place at run time so it is known as late binding

- Also known as Dynamic binding
- Best example of run time Polymorphism is method overriding.

Method Overriding

1. Method overriding happens between 2 classes.
2. In this method name and arguments both are same.
3. We have to use extends keyword means inheritance must be performed
4. We cannot use static method.
5. We need to create object of that class of which method we want to call.

Program:-

```
Public class Panipuri {

    Public void eat ()
{
System.out.println("I want to eat Panipuri
}
}
```

```
Public class Icecream extends Panipuri
{
    Public void eat()
    {
      System.out.println("I Want to eat
Ice cream");
    }

    Public Static void main (String [] args)
{
    Ice cream i = new Icecream();
      i.eat();

    Panipuri p = new Panipuri();
     P.eat();

  }
}
```

Output:-

I Want to eat Ice Cream
I Want to eat Panipuri

Conversion / Casting

- Converting one type of information into another type of information is called Conversion or Casting.

There are two types of Conversion / Casting

1] Primitive Casting

2] Non - Primitive Casting

Primitive Casting

- Converting one data type of information into another data type information is known as Primitive Casting.
- Primitive Casting Happens between 2 data types.
- There are three types of Primitive Casting.

1. Implicit Casting
2. Explicit casting
3. Boolean Casting

Implicit Casting

Converting lower data type of information into higher data type is called implicit casting

Also known as Automatic casting or Widening Conversion.

There is no loss of data

Program :-

```
Public class Implicitcasting {

Public static void main (String [] args)
{
    byte b = 10;
    short h = b;

System.out.println (h);

    short s = 100;
    int a = s;
    System.out.println (a);

    int n = 1010;
    long l = n;
System.out.println (l);
}
}
```

Explicit Casting

- Converting of higher data type information into lower data type information is called Explicit Casting.
- Also known as Forceful Casting or Narrowing Conversion.
- In this Casting there may be loss of data.

Program

```
Public class Explicitcasting {

  Public static void main (String [] args

    int i = 100000;
    Shot b = (short) i ;
    System.out.println (b);

    Float a = 57.99F;
    Short s = (short) a ;
   System.out.println (f);

    }
}
```

Output :- -31072
57
4.5454544E8

Boolean Casting

- Java does not support Boolean Casting. It is considered as a incompatible type of Casting. because we cannot convert true into false or vice versa

Non Primitive Casting

- Non Primitive Casting means Converting one type of class into another type of class is known as non-primitive Casting
- It is classified into 2 types.

1. Upcasting
2. Downcasting

Upcasting

- Assigning subclass Property into Superclass is known as Upcasting.
- Before Performing upcasting inheritance operation take placle. Where Properties of Superclass are inherited into subc

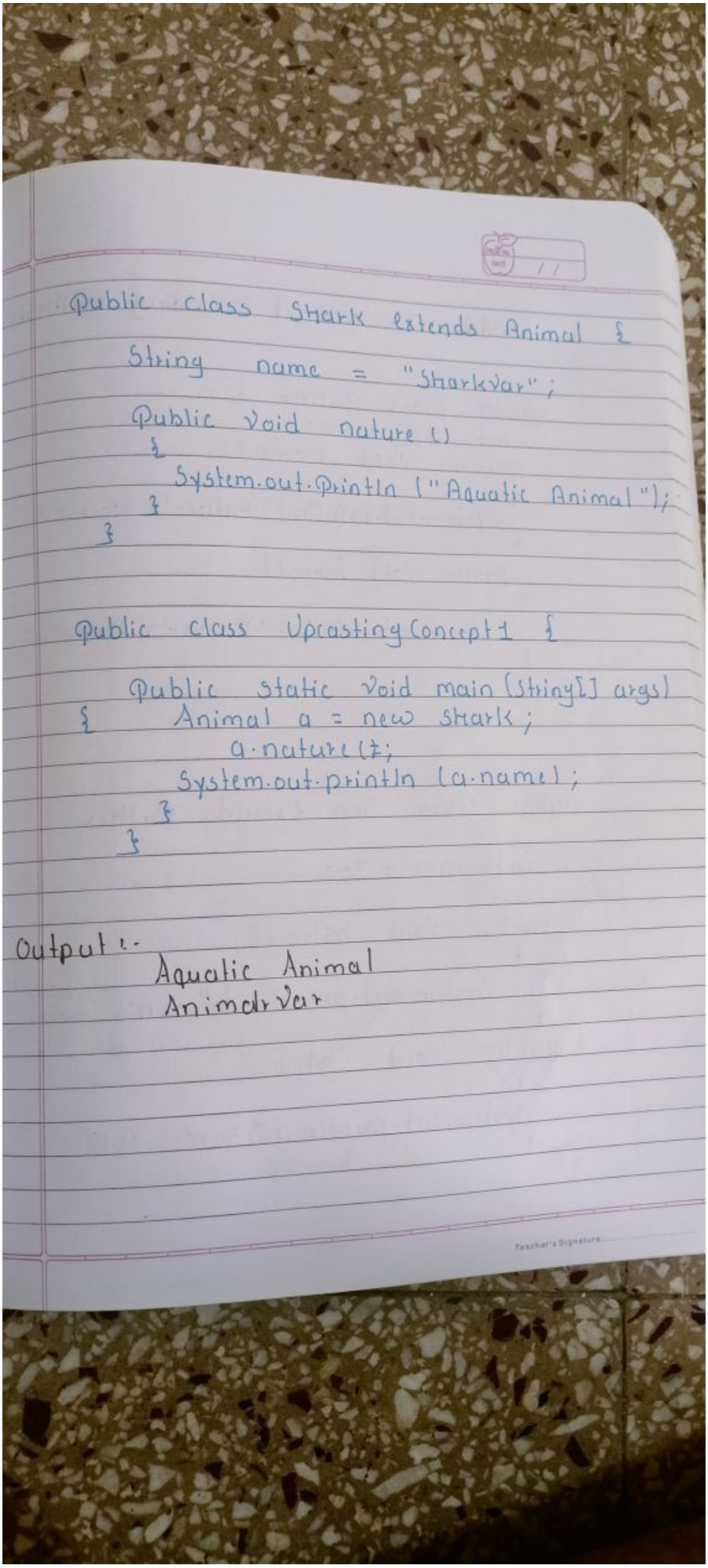
- Programmers can declare new Properties in Subclass.
- At the time of upcasting the Properties which are inherited from Superclass are only eligible for upcasting. New Properties are not eligible
- SYNTAX

```
Superclass refVar = new Subclass();
```

Program

Condition 1 - With Overriding method

```
Public class Animal {
String name = "AnimalVar";
Public Void nature ()
{
System.out.Println ("Animal");
}
}
```



```
Public class Shark extends Animal {
    String name = "SharkVar";
    Public void nature ()
    {
      System.out.Println ("Aquatic Animal");
    }
}

Public class Upcasting Concept1 {

    Public static void main (String[] args)
{     Animal a = new Shark;
        a.nature ();
      System.out.println (a.name);
    }
}
```

Output :-

Aquatic Animal
AnimalVar

Condition 2 - Without Overriding methods

```
Public class Father {
  int age = 60;
  Public void Home ()
  {
   System.out.println (" Father's House ");
  }
   Public void farm ()
  {
   System.out.println (" Father's Farm ");
  }
}
```

```
Public class Son extends Father {

   int age = 30;

   Public void bike ()
     {
       System.out.println (" Son's ~~job~~ bike ");
   }
   Public void job ()
     {
      System.out.println (" Son's job ");
     }
}
```

```
Public class UpcastingConcept2 {
  Public static void main(String[] args)
{
    Father f = new Son();
     P.Home();
     f.Farm();
    System.out.println(F.age);
  }
}
```

Output:

```
Father's House
Father's Farm
60
```

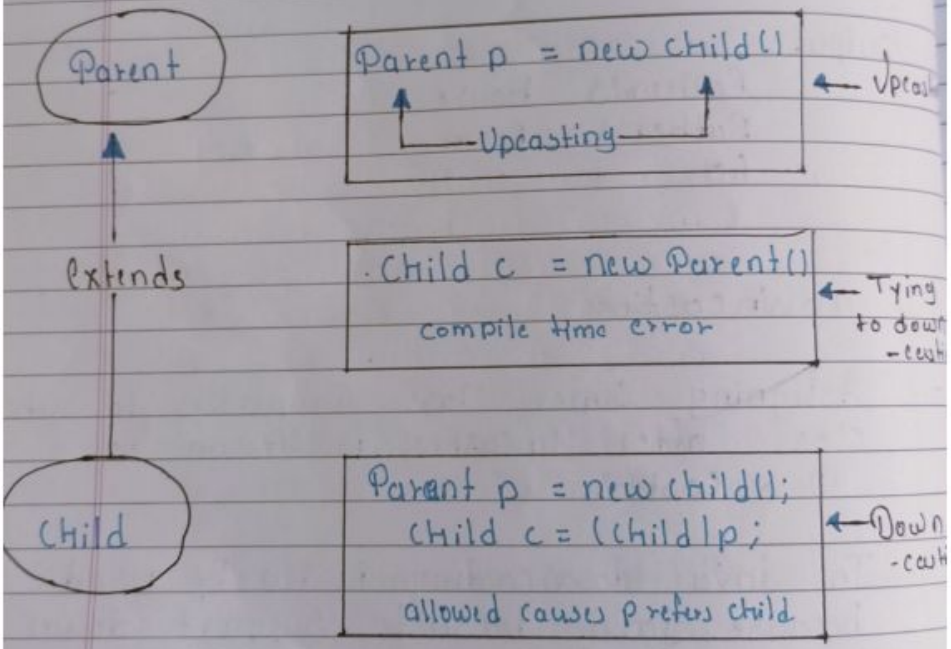
Downcasting

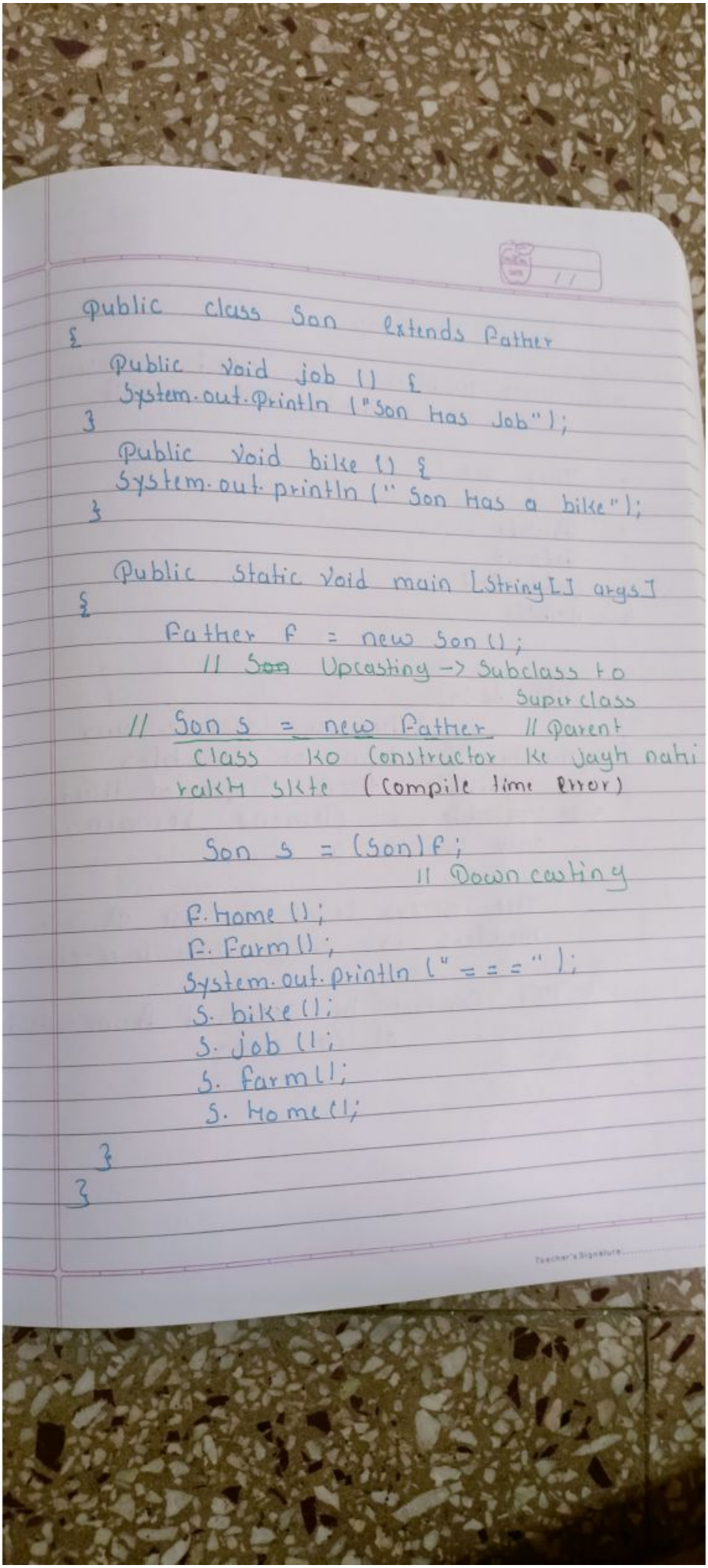
- Assigning super class Properties to Sub class after upcasting is known as Downcasting
- In java downcasting is rarely used because java doesn't support down-casting directly.
- Before Performing downcasting Compu upcasting should be Performed.

Teacher's Signature

- We can Perform downcasting in explicit way it forcefully
- So there may be loss of data or information.

Simply Upcasting and Downcasting





```
Public class Son extends Father
{
   Public void job () {
   System.out.Println ("Son Has Job");
}
   Public void bike () {
   System.out.println (" Son Has a bike");
}

   Public static void main [String[] args]
{
      Father f = new Son ();
        // Son Upcasting -> Subclass to
                            Super class
   // Son s = new Father  // Parent
      class ko Constructor ke jayh nahi
      rakhh sikte (Compile time Error)

        Son s = (Son)f;
                    // Down casting
      f.Home ();
      f. Farm ();
      System.out.println ("====");
      s. bike ();
      s. job ();
      s. farm ();
      s. Home ();
  }
}
```

Access Modifiers

- Access modifiers is used represent Scope of methods and Variables

- They are

1. Private
2. default
3. protected
4. Public

Private :-

- When we declare any member function or variables as Private then Scope of that ~~the variable~~ element remains only within a class

The access level of the Private modifiers is only with in a class

It Cannot be accessed from outside of the class

Default :-

When we declared any member of a class as default then scope of that elements remains only within Package.

There is no keywords to represent Default access modifiers.

The access level of default modifiers is only with in Package

If we do not specify any access Specifier, it will be default.

Protected :-

When we declared any member of a class as Protected the scope of that elements remains within the Package. Also it can be accessed outside Package only by inheritance.

- The access level of a Protected modifiers is within the Package and outside the Package through inheritance.

- Public :-

When we declare any member of class as public then scope of member remain everywhere throughout the Project.

The access level of public modifier every where. It can be accessed from within the class, outside of class, within Package and outside the Pack

Abstraction :-

Abstraction is one of the Pillar [Principle] of the object-Oriented Programming language.

Abstraction is a Process of Hiding the implemented code and Providing only functionality to the end User is known as Abstraction.

Example :- 1. ATM machine where we can withdraw and deposit the money but we don't know the internal Process on how it's done.

2. Msg in this we can send or recieve msg but we don't ~~no~~ know what is internal Process on how its done.

Abstract class

A class which is declared with abstract keyword is known as abstract class.

An a abstract class means where Programmer can declare both Complete a

Teacher's Signature

as well as incomplete method. In short it can have abstract and non-abstract methods.

An a incomplete methods means where method declaration is present but method definition is not present

Incomplete method should be declared with abstract keyword and end method Signature and Semi colon. followed by

We cannot create object of abstract class, but we can provide the reference variable of that abstract class.

Teacher's Signature.....

Concrete class

As we cannot create object of abstract, so we have approach to create an object of concrete class by using extends keyword.

Concrete class contains all the definition of incomplete methods present in abstract class. like if 10 incomplete methods are present in abstract class then subclass should provide definition of all the 10 abstract methods.

Concrete class have complete method only.

Teacher's Signature

Note :-

= If method inside a class is abstract then compulsory class should also be abstract

- And if class is abstract then its not compulsory to have abstract method. because abstract class have complete as well as incomplete methods in it.

Interface :-

- Abstraction can be achieved in 2 ways

1. Abstract class
2. Interface

Interface is one of the oops Principle . it is 100 % abstract in nature

That means method declare inside are by default - Public and abstract

Interface name always start with uppercase letter

Variable declared inside interface are by default - Public static final.

There is no constructor concept is present.

We cannot create object of interface as we cannot create object of interface so we have an approach to create the object of implementation class. Also we can provide the reference variable of interface while creating the object.

Implementation class will implements interface. That is, we need to perform inheritance between class and interface

Implementation class must contains all the implementation for all the abstract method in interface.

At the time of implementation methods should also be declared as Public.

Teacher's Signature.....

One interface extends other interface
(C with extends keyword)

(Imp! Interv) We can declare static method inside interface but its definition should be inside interface. That is we can declare Complete static methods only. Because after Java Version 8 interface does not allow incomplete static method

(Imp! Interv) Multiple inheritance can be achieved using interface.

Syntax :-

```
Public interface InterfaceName
{
}
```

Encapsulation :-

The Process of wrapping the data and corresponding methods into a single unit is called Encapsulation.

Example are Medicine capsule, School Bag.

If any component follows data Hiding and abstraction then it is called Encapsulation.

Encapsulation = Data Hiding + Abstraction

Steps to achieve encapsulation.

We Have to declare a variable / class as Private. (By declaring Variables as Private we Have achieved data Hiding means outside Person cannot access this Variable.)

We Have to use Public getters and Setters method to modify and view the Variable Value.

Few Points about encapsulation.

User would never know of ~~work~~ what is going on in the background. they would be only aware of update the field by Set method and read a field by get method

Set and get words used in the methods names are only for naming Purpose so that Programm should understand which is set and get method.

Advantages are :-

1. Encapsulated class is easy to test so it is better for doing unit testing
2. It Provides Security.

Difference between Abstract Class and Interface

Abstract class	Interface
Abstract class can have abstract and non abstract method (com-lete and incomplete Meth.)	Interface have only abstract method
Abstract keyword is use to declare abstract class	Interface keyword is use to declare In
abstract class can extend using "extends" keyword	Interface can imp-ents with "implem Keyword
We cannot create object of abstract class so we have approach to create object of concreter class this concrete class extends abstract class	we cannot create ob at Interface class s we have approach t create object of im-ment class. this implementation clas implements interfa
Abstract class achieves Partial abstraction i.e (0-100)	Interface achieves fully abstractio
Variables in Abstract can final, nonfinal, static, non static.	Variable in Interfa only static an final

Teacher's Signat

Access modifiers for method can be any except Private.	Access modifiers declared here for methods are by default Public and abstract.
Abstract class can have constructur	Interface cannot have Constructor.
Multiple Inheritan cannot be achieved by abstract class.	multiple inheritance can be achieved using inteface.

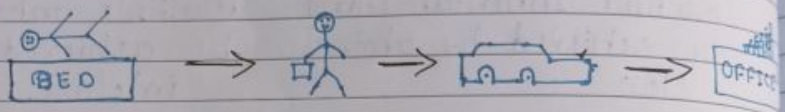
Difference between class and Inteface.

class	Interface
Class Have Complete methods	Interface Have abstract methods
Access modifiers for methods can be any Public, Private Protected.	Access modifier for method in only Public
A class Can Suppo -rt Constructor	Intetace does not support Constructor.
A class is declaring using class keyword	Interface is dellaring using intetace keyw
We Cannot achieve multiple inheritance in class	We Can achieve multiple inheritan -ce in Interface.

Exception Handling

Exception Handling is very important if ~~our~~ we want our Program to run Properly.

Real Time Example :-



This will be termed as a normal Flow because these is your daily routine and this executes everyday.

But if bike/car get Puncture or Brake fails. this can be said as a unwanted event

Unwanted Event = Exception.

Exception :- Any Unwanted event which distrubs the normal flow of the Program is called Exception.

And Handling these Exception called as Exception Handling.

Teacher's Signature.....

* How We Can Handle the Exception?
-> By Creating the alternat Way. [Way]

Consider the Privious example
If our bike / car tyre Puncture, this is an a exception we can Handle these exception -> we can do is Parks the bike in Somewhere and take any other mode [auto or cab] commute and reach to the office.

Program :-

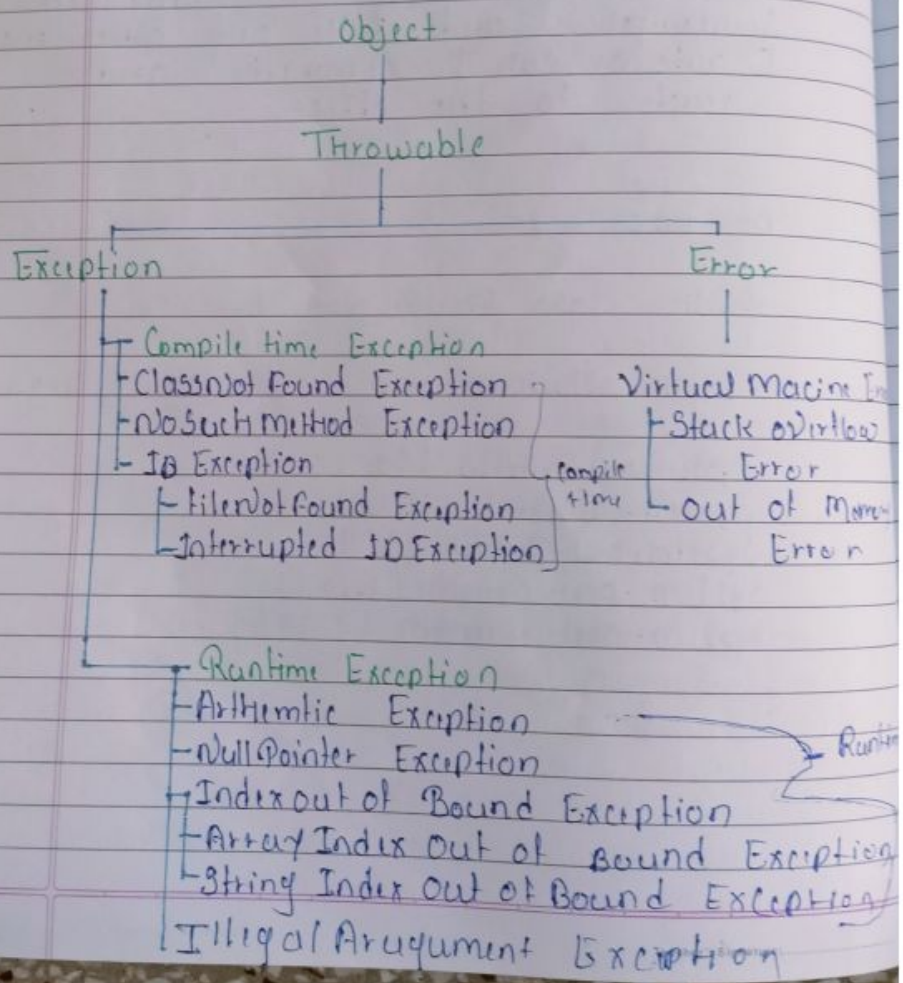
```
Public class FirstProgram {

  Public Static Void main (String [] args)
{
   System.out.println ("1");
   System.out.println ("2");
   System.out.println ("3");
   System.out.println (2/0);
   System.out.println ("5");

  }
}
```

Teacher's Signature.....

Hierarchy :-

1. object class is a Parent class of all the classes in Java.

2. Throwable is the Parent class of the Exception and Error.



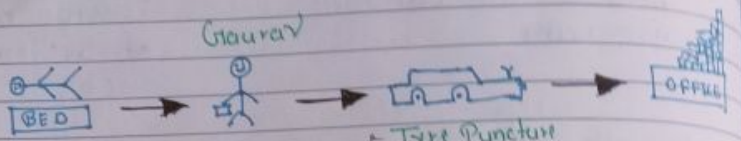
Exception	Error
Exception occur because of our Program.	Error occur because of lack of System resour -ce
Exception can be Handle by Programers.	Error cannot be Handle by Programmer.
Exception are two Types	Error are only of 1 Type
1. Compile Time Exception Or Checke Exception 2. Run Time Exception Or Un checked Exception	1. Runtime Exception Or Unchecked Exception

Note :-

All the Exception occur at run time only.

Not at a Compile time

What is the difference between Checked & Unchecked Exception ?



Mother / Compiler

Run time
- Tyre Puncture
- Brake Fail
- Rain
  - ↳ TyrePuncture Exception
  - BrakeFail Exception
  - Rain Occur Exception

Wallet WalletNotFound Exception
TiffinBox TiffinBoxNotFound Exception
CompanyID CompanyIdNotFound Exception

Lets Consider Here a character as mother. Now mother will ask us you Have Have taken your wallet, Company ID, Tiffin Box.

Usually mother asks this questions so that in future (or office) we should not face any issue i.e. In Runtime we should not face any issue at our office.

Here mother will act as a compiler and it does not mean

that Exception Has occured at Compile time we are just checking it.

So mother is just checking so that His Son should not face any Exception at runtime.

[If we forget our ID or tiffin we should not Have any Exception at Home (mean Compile we have Exception or Problems at office (mean In Runtime)

["Agar aap apna ID Bhul jate Ho to ghar pe koi Exception nhi ayenga. Exception aygu wo Hai runtime me."]

So, Exception which are checked by Compiler is called Compile time Exception. [Checked Exception]

And

Exception which are not checked by Compiler is called as Unchecked Exception [Run time Exception]

	Checked Exception	Unchecked Exception
1	Checked exception are the exception that are checked at compile time.	Unchecked Except are not checked at Compile time
	The Program gives a Compilation error if a checked Except -ion occur	The Program Comp fine because compil is not able to find the Checked the Unch cked exception.
-	In checked Exception occur we need to Handle it by using try Catch block	Unchecked Exception occur mostly due to Programing mistake.

So, Exception Depending on real time Example for checked and Unchecked

Checked Exception:- WalletNotFound Exception, CompanyIDNotFound Except Tiffinbox notfound exception

Unchecked Exception:- Tyrefailexception
-> Brakefail exception
-> Trafficblock exception.

Teacher's Signature:.....

* Next Question app ke dimag me Hoga ki yahi particular exception checked exception ke under kyu aata hai?

= Checked exception are only those exception which is checked by compiler.

Unchecked exception are those exception which are not checked by compiler.

* SYNTAX :-

```
try
{
    =
    =          // code
    =
}
catch (Exception e)
{
    =
    =          // code
    =
}
```

Imp Points for try catch block.

1. Try block should always have catch block

2. With 1 try block we can write (or have) a multiple catch block (Bu

Should be in a Sequence (First child after Parent)

3 The Program can Have try catch and finally block as well

4 Nested try catch block also valid
Ex:-

```
try
{
    try
    {

    }
    catch (Exception e)
    {

    }
}
catch (Exception e)
{

}
```

* Methods to Print Exception Information in Java.

① e.printStackTrace();

This is a best way as it displays all the info.

③ Sysout (e);
System.out.println(e);

This ~~Displa~~ Display Exception name and discription, But Not StackTrace.

④ Sysout (e.getMessage());
System.out.println(e.getMessage());

This display only discription.

* finally block

finally block always executed whether Exception is Handle or not.

Syntax (Most Perferable Way)

```
try
{
}
Catch (Exception e)
{
}
finally
{
}
```

Flow of Program

① If Exception Occur

try - Catch - finally

② If Exception does not occur

try - finally

ARRAYS

- Array is use to store a group of information.
- Array should be first declared with its capacity
- Array index start from zero.
- In array we can represent multip values with a single var

Limitations of Array :-

1. Size of array is fixed, it ca

Teacher's Signature.....

not be increased or decreased once declared.

2. Readymade or a Predefine method support is not available. We have to write the code for it.

3. It is Homogenous in nature means can store only one type of data.

Program 1

```
Public class Array1
{
  Public static void main (String [] args)
  {
    int a [] = new int [5];
     a [0] = 1;
     a [1] = 2;
     a [2] = 3;
     a [3] = 4;
     a [4] = 5;

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

output :-

1
2
3
4
5

```
Public Class Array2 {
 Public static void main (String [] args)
 {
   int a[] = new int [3]
      a[0] = 1;
      a[1] = 2;
      a[2] = 3;
      a[3] = 4;   // trying to initialize
                     3rd index value
    for(int i=0; i<a.length; i++)
    {
      System.out.println( a[i] );
    }
 }
}
```

Output :-

Exception in thread "main" java.lang.
ArrayIndexOutofBoundsException: Ind
3 out of bounds for length 3 at
array.Array1.main (Array1.java:12)

Collection :-

The Collection in Java Provides architecture to store and manipulate the data efficiently In Java. Collection can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

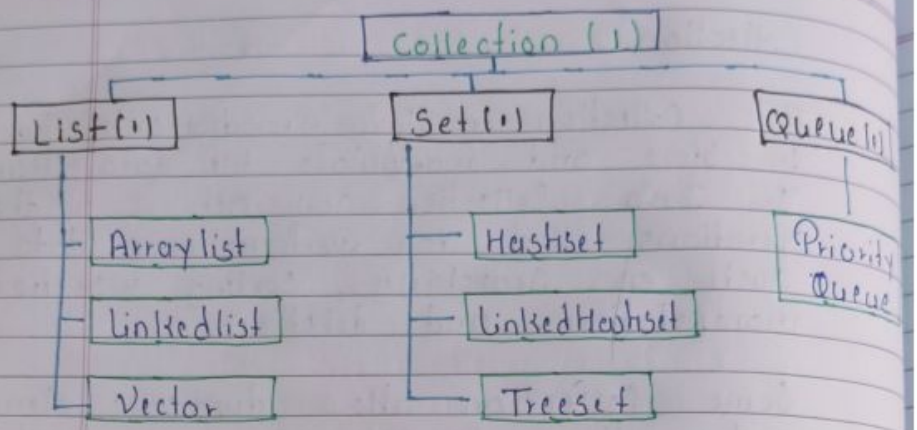
Some Collection allow duplicate element and others do not. Some are ordered and others unordered.

Limitations in array are overcome here.

1. In Collection size is not fixed, it can be increased or decreased as per requirement.

2. Heterogeneous and Homogeneous in nature.

3. Readymade or Predefine methods suppor is available which make Programm task super easy.



Collection Hierarchy

* List

1. Duplicate are allowed in list
2. Allows any number of null value.
3. Here ordered of insertation is maintained.

* Set

1. Duplicates are not allowed.

2. Allows only one null value.
3. Order of insertation is random.

* Arraylist

1. Duplicates are allowed.
2. Allows any number of null value
3. Order of insertation is maintained
4. Default Capacity of arraylist is 10.
5. Arraylist is resizable.
6. Icremental capacity = (current capaci * 3/2)
7. Manipulation with Arraylist is slow be it internally use an array. If any element is removed or a in the array all the other are shifted in the memory. So a worst choice for manipulat
8. Best choice for retrieval opera

Program I

```
import java.util.Arraylist;
Public class Arraylist {

Public Static Void main (String [] args) {

    Arraylist al = new Arraylist ();

     al.add (100);
     al. add ("abc");
     al. add ('D');
     al. add (100);
     al. add (null)
     al. add (null)

  System. out. println (al);

  System. out. println (al.size ());

  System. out. println ( al.get (2));

   al. remove (4)

   System. out. println (al.isEmpty ());

   al. clear ();

   System. out. Println (al);

  }
}
```

LinkedHashSet

Duplicates are not allowed.

Allow only ~~Nul~~ one null value.

Order of insertation is maintained

LinkedHashSet is a child of HashSet i.e LinkedHashSet extend HashSet.

Default Capacity of LinkedHashSet is 16 and load factor is 0.75.

LinkedHashSet is not Synchronized and not thread safe. if multiple thread try to modify at same time then the final op outputcome is not determined

LinkedHashSet is introduse in version 1.4

It is best choice when we have to remove duplicate element and order of insertation is mandatory.

TreeSet

Duplicates are not allowed.

Null value are not allowed.

Order of Insertation will be according to default natural sorting order.

Only Homogenous elements are allowed

~~Th~~ Treeset Has no default Capacity.

Treeset is not Synchronized and not thread Safe, If multiple thread [try to modify] at same time then the final outcome is not deterministic.

Introduce in Version 1.2 v.

Difference between HashMap and HashTable.

HashMap	HashTable
HashMap is non-Synchronized and not thread safe	HashTable is Synchronized and thread safe
HashMap allows one null key and multiple null values	HashTable doesn't allow any null key or null value
It inherits a AbstractMap class.	It inherits a Dictionary class
It not a legacy class	It is a legacy class
It introduce in version 1.2v.	In introduced in 1.0v

Program 1

```
import java.util.Arraylist;
Public class Arraylist {

Public static void main (String[] args) {

    Arraylist al = new Arraylist();

      al.add (100);
      al.add ("abc");
      al.add ('D');
      al.add (100);
      al.add (null)
      al.add (null)

System.out.println(al);

System.out.println (al.size());

System.out.println (al.get(2));

 al.remove(4)

 System.out.println (al.isEmpty());

 al.clear();

 System.out.Println(al);

 }
}
```

Teacher's Signature

OUTPUT

```
[100, abc, D, 100, null, null]
6
0
[100, XYZ, D, 100, null, null]
[100, XYZ, D, 100, null]
true
false
[]
```

Program 2

```
import java.util.LinkedList;
Public class LinkedList1 {

Public static Void main (String[] args)
{
    LinkedList 11 = new LinkedList

        11.add (500);
        11.add ("abcd");
        11.add ('w');
        11.add (1030);
        11.add (null);

System.out.Println.(11);

System.out.Println (11.size());
```

```
System.out.println (ll.get (2));

ll. Set (1, "XYZ");
System.out.println (ll);

ll.remove (4);
System.out.println (ll);

System.out.println (ll.contains ("XYZ"));

System.out.println (ll.isEmpty ());

ll.clear();

System.out.println (ll);

}
}
```

Output:-

```
[500, abcd, W, 1030, null, null]
6
W
[500, XYZ, W, 1030, null, null]
[500, XYZ, W, 1030, null]
true
false
[]
```

Program 3

```
Public class Hashset {
    Public static void main (String [] args)
{
        Hashset H = new Hashset ();
        H.add ("Gaurav");
        H.add (100);
        H.add ('D');
        H.add (90.3);
        H.add (100);
        H.add (null);
        H.add (null);

    System.out.println (H);
    System.out.println (H.size ());
    System.out.println (H.contains (100));
     H.remove (null);
    System.out.println (H);
    System.out.println (H.isEmpty ());

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

Output :-

[null, Gaurav, 100, D, 90.3]

5

true

[Gaurav, 100, D, 90.3]

false

[]

Teacher's Signature.....

Program 4.

```
Public class LinkedHashSetEx {
Public static void main (String [] args)
    LinkedHashSet l = new LinkedHashSet();
    l.add ("Peter");
    l.add (170);
    l.add ('Y');
    l.add (90.8);
    l.add (null);
    l.add (null);
    l.add (170);

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

Output

[Peter, 170, Y, 90.8, null]

1. What is Synchronization ?

→ It is a Process by which we control the accessibility of multiple threads to a Particular Shared resource.

2. Problem Without Synchronization?

→ Final outcome is not deterministic.

→ Thread Interference.

3. Advantages of Synchronization ?

→ Final outcome is deterministic

→ No Thread Interference.

4. Disadvantages of Synchronization:

→ Increase the wating time period of thread.

→ Create Performance issue.