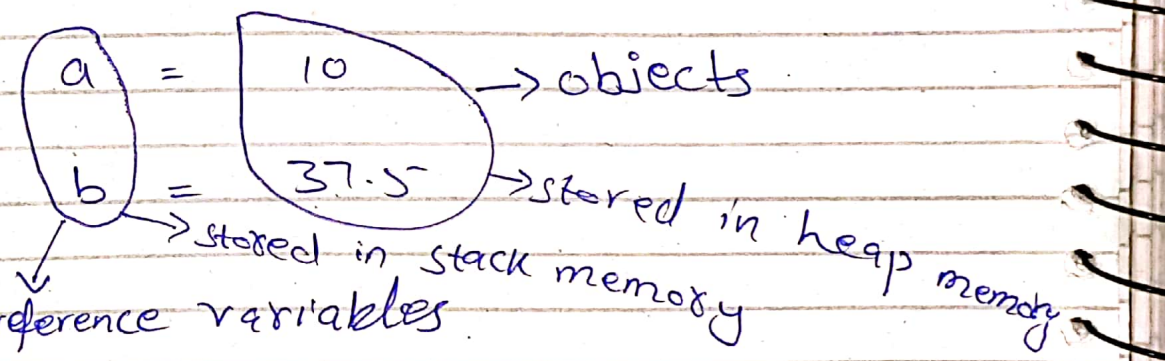
Two type of memories in programmings are:

i) Stack

ii) Heap ← stored in

Stored in → a = 10

→ Stack memory pointing towards heap memory.



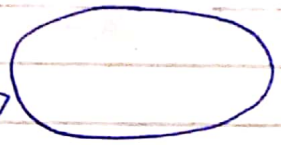
⇒ if original variable changed, then change will be visible to all reference variables.

⇒ Object with no reference variable, then garbage collector will take this object / remove this object.

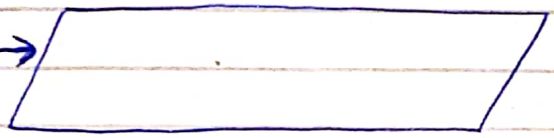
oblec

Day 02 :-

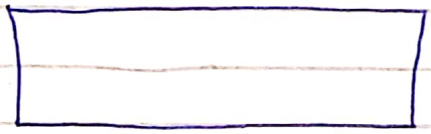
Start / Stop →



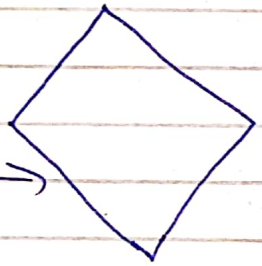
Input / Output →

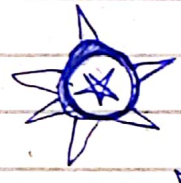


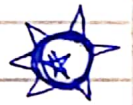
Processing →



Condition →







07 Lec

Day 03

JDK = JRE + Development Tools
(Java Development Kit)

JRE = JVM + Library classes
(Java Rutime Environment)

Java Virtual Machine
(JVM)

JIT
Just-in-time

ob lec

Day - 02

JDK =

It's basically a package, a set of files that we can download from the internet.

→ It provides an enviroment to develop and run the Java program.

→ If we want to create an application, we need JDK for that. It contains all those libraries and all those things.

→ If we want to develop Java program, JDK is required. we can download from internet oracle or what ever.

JRE is the box basically and JVM is the actual content inside the box.

→ compiler is only in JDK.

→ JRE is nothing but JVM + extra files.

① Difference

JDK is a package of tools for developing Java-based software, whereas the JRE is a package of tools for running Java code.

08) First Java program - Input/Output, Debugging and Data types

-> Every file that ends with Java is a class itself.
-> Class name start with Capital letter. (It's convention).
-> Class is just name group of properties and functions.

Changing location of Byte Code:

* Javac -d . Demo.Java
  ↳ current Directory
* Javac -d .. Demo.Java
  ↳ will store byte code in previous directory.

* Where javac
  ↳ this will tell location of javac where it is located like

output
/usr/bin/javac

* ls /usr/bin | grep javac
  ↳ will find javac in current Directory
* javac
  ↳ is only doing double click on this file
* Javac = /usr/bin/javac
  ↳ this is very tedious so we don't want to do.

08 lecture

Day 02

How to add environment variables for windows? Search on google if there is any error occur while running commands.

* Cat .zprofile
  ↳ will show path in linux.

* echo $path → printing environment variable paths
  ⇒ Environment variables are just a list of folder addresses in which your computer will check whether the commands that you have written in terminal is available or not.

⇒ Public class can only be the class, that is the name of the class.
→ Other then public class, can be the different name from file name.

Shortcut Command: (i) PSVM
(ii) SOUT (for output)
↳ in IntelliJ Idea for creating main() function Template.

package com.kunal
In simple terms, it is a folder in which your java file lives.

08 Lecture

Day 02

How to add environment variables for windows? Search on google if there is any error occur while running commands.

* Cat .zprofile
  ↳ will show path in linux.

* echo $path → printing environment variable Paths

⇒ Environment variables are just a list of folder addresses in which your computer will check whether the commands (in terminal) that you have written is available or not.

⇒ Public class can only be the class, that is the name of the class.
→ Other then public class, can be the different name from file name.

Shortcut Command: (i) PSVM ↳ in IntelliJ Idea for creating main() function Template.
(ii) SOUT (for output)

package com.kunal
In simple terms, it is a folder in which your Java file lives.

08 lecture

Day 02

Why we create package?

It might need to have some privacy.

→ All the things are happening in Java in classes.

→ Java people have provided some basic function to us that created Java. These functions makes our life easier.

→ Everything in lang package we can access in our files.

Output in Java

* System.out.println("Hello World");
  - ↳ is a class that provides all these (println) nice functions.
  - → System contains out variable which is of type print stream
  - → println is basically built-in function that is written by developers that create Java

Input in Java

```
import java.util.Scanner

Scanner input = new Scanner(System.in)
```

↳ Here in the object we have to pass from where we are taking the input i.e may be i'm reading a file etc.

of lec 2000 | 340 | 343 | 1500 Day 02

2000+500 + 680 = 3180 usman

System.out.println(" ");
→ On standard output stream, print something.

i) Scanner input = new Scanner(System.in);
→ which is initializing this particular object.

constructor

correspond to keyboard input

→ Every class in Java extends the object class.
↳ confused

→ pointing to the object of Scanner class.

ii) System.out.println(input.next);
↳ will take first word of sentence.

* input.nextLine();
↳ will take whole sentence of characters

* input.nextInt();
↳ will take only Integer.

Primitive Data types

→ Primitive is basically any data type that you cannot break even further.

i) int rollNo = 64; // four bites
ii) Char letter = 'x';
iii) float marks = 98.67f; // 4 bytes

Lecture 08 Day 02

→ // 8 bytes

iv) double largeDecimalNumbers = 4892032.4567;
→ // 8 bytes

v) long largeInteger = 34567802381L;

vi) boolean check = false;

Question!

Why f and L at end of floating and long value?

→ Bydefault, floating point values of type "double", that's why we add f at the end of float "type" value.

→ Bydefault, Integer value of int type, so we specify L with "long" type.

Carefully Note:

Although above data types are primitive types, they contains classes for these as well. these classes called Wrapper classes.

⇒ These type of data have extra methods that can be performed on this data.

i.e Integer rno = 64;
rno.byteValue();

{ in competative programming, we need these methods, so we use this.

08

Lecture 08

Day 02

Debugging:

Try by yourself on intellij Idea.

literals:

In primitive data type, literals basically are the syntactically representation of like boolean, character, int, long, double etc.

```
int a = 10;
    ↓       ↓
identifiers  literal
```

```
int a = 234_000_000;
System.out.println(a);
```

↓

Underscore will be ignored and comma will be placed.

* Type Conversion Vs Type Casting

→ Search by yourself.

Type Casting & Conversion

→ Destination type should be greater than Source type.

→ Java also performs automatic type conversion when storing a literal integer in an integer constant into variable of type like bite, short, long etc. (It takes ASCII value of that)

Lecture 08 Day 03

Type Casting:

Compressing the bigger number in a smaller type explicitly is called type_casting.

```
int num = (int)(67.56f);
System.out.println(num);
// output : 67
```

-> Automatic type promotion in expressions

```
int a = 257;
byte b = (byte)(a); //257 %.256
                    = 1
```

```
byte a = 40;
byte b = 50;
byte c = 100;
int d = a*b/c;
```

↳ dealing as a integers for this expression although these type is byte

-> byte evaluation automatic converted into int. So we have to use explicitly casting

```
byte b =50;
b = (byte)(b2);
```

Lecture 08

Day 03

Type promotion Rules:

```
byte   b = 42;
char   c = 'a';
short  s = 1024;
int    i = 50000;
float  f = 5.67f;
double d = 0.1234;
double result = (f * b) + (i / c) - (d - s);

System.out.println((f * b) + " " + (i / c)
                   + " " + (d - s));
// 238.14 515 -1023.8766

System.out.println(result);
// 1777.0166146484376
```

Note: Resulting type from expression always promoted to bigger one type.

float + int + double = double

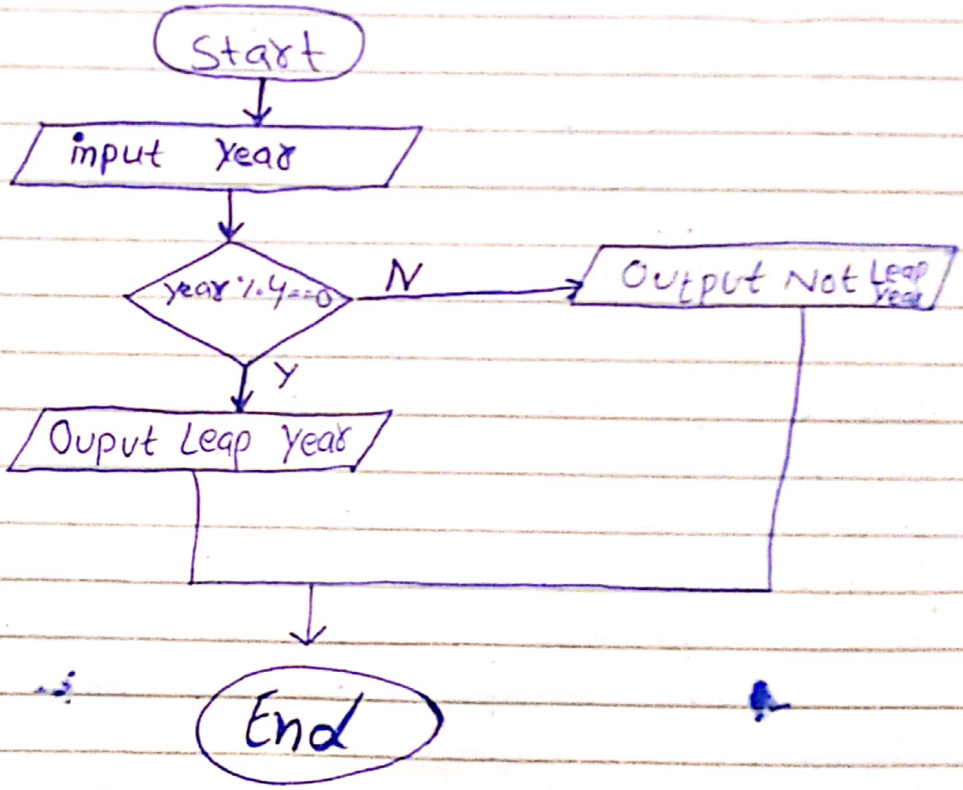
Day 04

Assignment 01

Create flowchart and pseudcode for the following:

1. Input a year and find whether it is a leap year or not.

flowChart



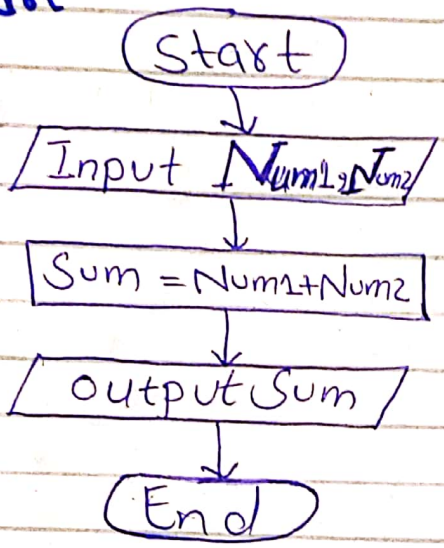
Pseude Code

```
Start
Input Year
if Year %4 ==0
    Output leap year
else:
    Output Not leap Year
exit
```

Day 04

2. Take two numbers and print the sum of both.

FlowChart

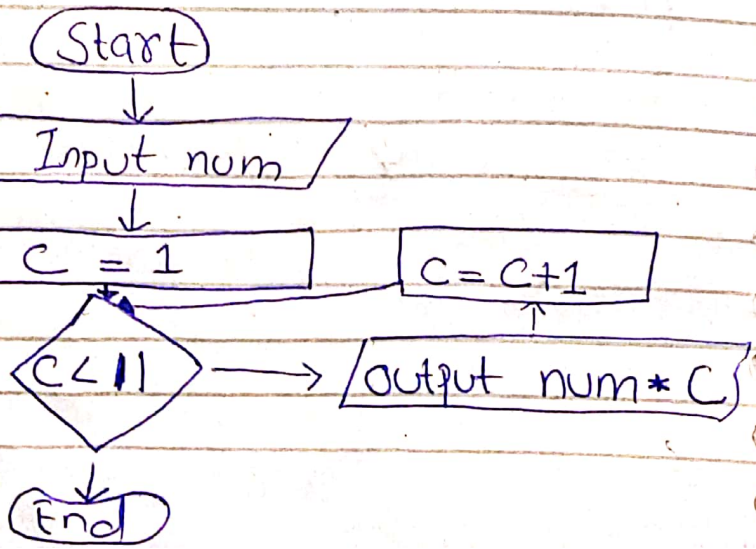


Pseudo Code

Start
Input num1 and Input num2
Sum = num1 + num2
Output Sum
exit

3) Take a number as input and print the multiplication table for it.

Flow Chart



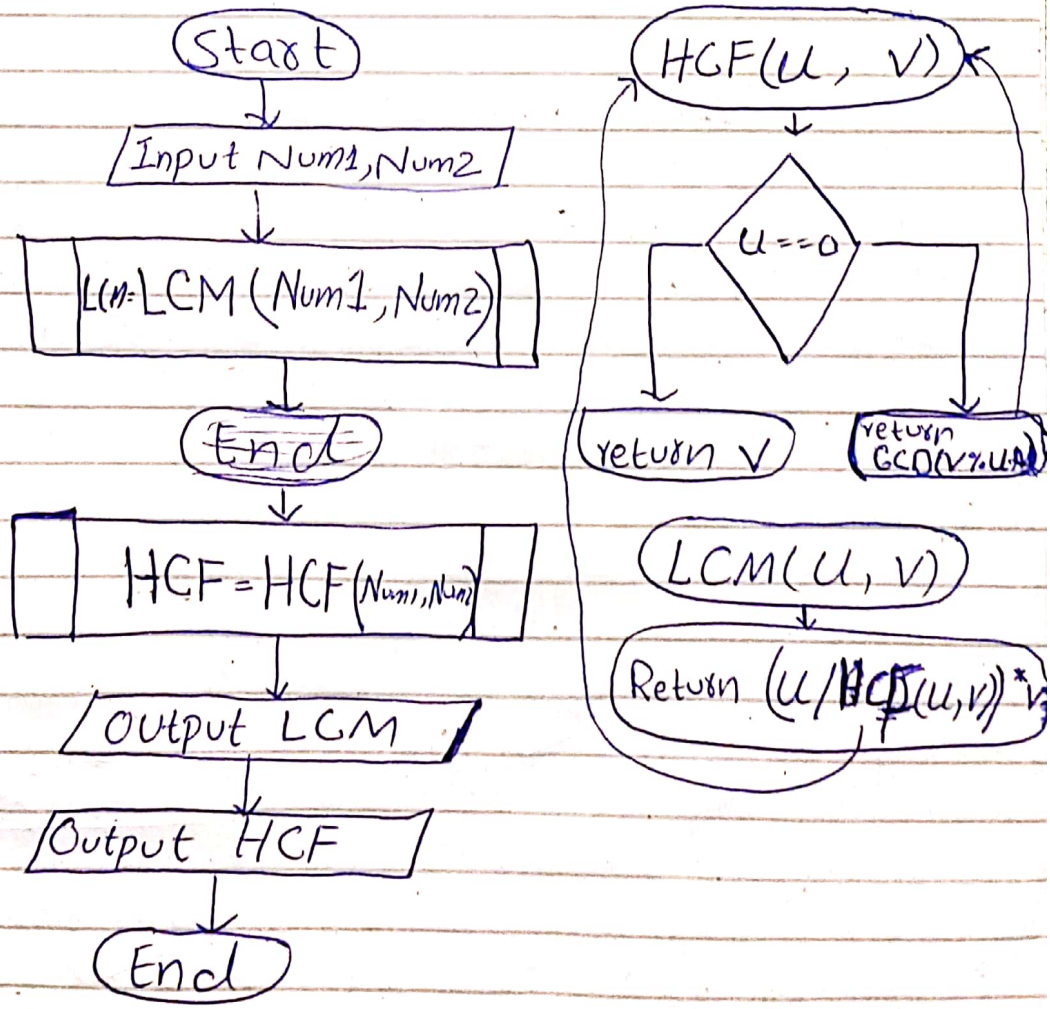
12 | 80 Days
72
8

Pseudo Code

Start
Input number
c=1
While c <= 11
    Output number * c
    c = c+1
end While
Exit.

4) Take two numbers as inputs and find their HCF and LCM.

flowChart



Day 05

Pseudo Code

Start
Input num1, num2
Output GCD(num1, num2)
Output LCM(num1, num2)
End

GCD(num1, num2)
if num1 == 0
    return num2
return GCD(num2 % num1, num1);

LCM(num1, num2)
return (num1 / GCD(num1, num2)) * num2;

Conditionals and Loops

```
if(ch >= 'a' && ch <= 'z'){
    System.out.println("LowerCase");
}else{
    System.out.println("UpperCase");
}
```

Note Carefully

Another approach for finding count of specific characters from String

i) n % 10
ii) n / 10

> Both points are very important

Note:

Bit Masking and Shift operators are important in Java.

79532 ⇒ 23597

ans = 7 * 10 + 9 = 79
= 79 * 10 + 5 = 795
⋮

ans = 7953 * 10 + 2 = 79532

10 Lec
11 Lec

Day 07
Day 08

Functions / Methods in Java

Coding Rule:

DRY : Don't repeat yourself

"Static" example:
Whole population of World is same for every person living in any country.

→ Cursor on not defined function call, the alt + enter, creates function automatically.

```
main() {
    name = "Kunal";
}
```

→ object
This is in heap memory.

Note:
No pass by reference in Java, Only pass by value.

```
psvm () {
    String name = "Ahmad";
    change(name);
}
static void change(nam) {
    nam = "Sultan";
}
```

→ not changing, only creating a new object.

* primitives: int, short, char, byte
  ↓
  Just passing value

Shaddowing

a) We can't use object dependent things in object independent things

مٹادے اپنی ہستی کو اگر کچھ مرتبہ چاہیے
تو دانہ خاک میں مل کر گل و گلزار بنتا ہے۔

Variable length argument
↳ always added at the end

Function Overloading:

function with same name but with different arguments.

i) Servence of arguments
ii) types of arguments
iii) number of arguments

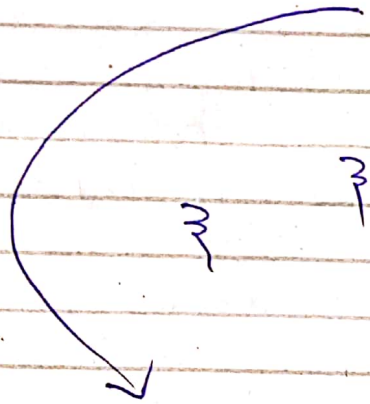
SomeNumber / 10
SomeNumber % 10
} Keep in mind these two instruction for logic.

eg:- public class Shadowing {
    static int x = 90;
    psvm () {
        System.out.println(x);

        x=50;
        System.out.println(x);
    }
}

90

50



Here high-level scope is shadowed by low-level scope.

Variable Argument

is used to take a variable number of arguments.

Syntax:

```
static void fun (int ...a){
        // method only
}

fun(1,2,3);
```