YSPM's

Yashoda Technical Campus, Satara

FACULTY OF ENGINEERING

Department of Computer Science and Engineering

Laboratory Manual

-: Course Name :-

COMPETITIVE PROGRAMMING LAB

-: Course Code :-

BTCOL606

-: Class :-

T.Y. B.Tech.

(Semester VI)

Yashoda Shikshan Prasarak Mandal's

Yashoda Technical Campus,

Satara

Faculty of Engineering

Department of Computer Science and Engineering

Content

Vision and Mission of the Institute

Vision of the Institute:

YTC, Satara looks forward to become a globally renowned institute of Centre of excellence in technology and management education for rural community for technical and professional knowledge.

Mission of the Institute:

M1: To achieve the quality and an academic excellence in the frontier engineering areas and management relevant primarily to the nation.

M2: To train and produce the highly skilled and globally competent professionals through quality technical education and to prepare them with industry ready engineers for immediate employment and entrepreneurship.

M3: To inculcate and develop the research culture to be attributed to quality outputs in terms of research practices and products.

M4: To develop the professionals having high values of ethics, lifelong learning, teamwork, leadership and social responsibility.

M5: To enhance and empower the rural community by improving the productivity of the agricultural sector.

Vision and Mission of the Department

Vision of the Department

To lead in technical, quality education, innovation, research for development of sustainable & inclusive technology for the society.

Mission of the Department

M1: To create ambience of academic excellence through state of art infrastructure

M2: To create student-centric pedagogy that will lead to employability.

M3: To create a software engineering professional with knowledge of multidisciplinary fields, can provide innovative products & service to society.

M4: To train and motivate the students for lifelong learning, employability and entrepreneurship

Program Educational Objectives (PEOs)

The PEO's are to facilitate graduating students to be

PEO1: Equipped to analyze and solve industry problems using a strong foundation in engineering sciences and computer science and engineering.

PEO2: Skilled in solving real-world problems in a multidisciplinary environment using modern tools and techniques.

PEO3: Developed into an ethical IT professional who contributes to societal and environmental progress.

Program Outcomes (POs)

Engineering graduates will be able to

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Demonstrate understanding of contemporary knowledge of engineering to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning**: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

The PSO's are to facilitate graduating students -

PSO1: To be able to give solution in networking, OOP, web development, cloud, IOT on real life application using open-source software.

PSO2: To be able to acquaint with modern trends in industry/research giving novel solution to existing social problems.

University Syllabus

At least twenty-five problems solving on competitive programming platforms such as,

https://uva.onlinejudge.org

http://hackerrank.com/

http://codechef.com

Course Outcomes

Competitive Programming Lab (BTCOL606)	
Course Outcomes (COs): After completing the course student	
BTCOL606_1	Learn to implement various Programming Challenges by Choosing different Programming Languages
BTCOL606_2	Implement various searching and sorting Algorithms, Arithmetic and Algebra technique to solve real-world problems

CO-PO Mapping:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
BTCOL606_1	3	3	2									
BTCOL606_2	2	3	3									
Average	**2.5**	**3**	**2.5**									

CO-PSO Mapping:

Cos	PSO1	PSO2
BTCOL606_3	2	1
BTCOL606_4	3	2
Average	**2.5**	**1.5**

List of Experiments

Exp. No.	Title of Experiment
1	Program to print Hello, World. Hello, Java.
2	Program to read data from stdin and then print them to stdout.
3	Program by using *if-else* conditional statements to automate decision-making processes.
4	Program to print formatted output.
5	Program to do some simple math using loops.
6	Program to hold integer values and must determine which primitive data types are capable of properly storing that input.
7	Program to read n lines of input until you reach *EOF,* then number and print all n. lines of content.
8	Program to find the Parallelogram and Constraints are -100 $\leq$B $\leq$ 100, -100 $\leq$ H$\leq$ 100.
9	Program to convert integer n into string.
10	Program for input containing the space separated month, day and year, respectively, in MM DD YYYY format.
11	Program for a single double-precision number denoting payment.
12	Program for a String where the first line contains a string A, the second line contains another string B. The strings are comprised of only lowercase English letters.
13	Print a substring consisting of all characters in the inclusive range from start to end-1.
14	Program to Print Lexicographical order.
15	Program to print Yes if it is a palindrome, print No otherwise.
16	Program to Check Two strings, a and b, are called anagrams if they contain all the same characters in the same frequencies.
17	Print the number of tokens, followed by each token on a new line.
18	Program for pattern syntax checker.
19	Program to write a regular expression to find the valid IPs for a provided string containing any combination of ASCII.
20	Program to remove instances of words that are repeated more than once but retain the first occurrence of any case-insensitive repeated word.
21	Program to check the valid username regular expression.
22	Program to print the content enclosed within valid tags.
23	Each number must be printed in the exact same format as it was read from stdin, meaning that .1 is printed as .1, and 0.1 is printed as 0.1.
24	Program to print if the given number is a prime number, then print prime; otherwise, print not prime.
25	Program to print sequential elements of the given array

Practical No.: 1

WELCOME TO JAVA!

Aim: Program to print Hello, World. Hello, Java.

Tools used: www.hackerrank.com

Theory:

In Java, we usually use the println() method to print the statement. It belongs to the PrintStream class. The class also provides the other methods for the same purpose. In this practical we will learn how to print in java. The method we should use depends on what we want to print and what type of output we want. There are some methods to print the statements

- print() Method
- println() Method

print() Method:

The print() method is used to print text on the console. It is an overloaded method of the PrintStream class. It accepts a string as a parameter. After printing the statement, the cursor remains on the same line. It also works if we do not parse any parameter.

Syntax: system.out.print();

println() Method:

It is an upgraded version of the print() method. It also used to display text on the console. It is an overloaded method of the PrintStream class. It accepts string as a parameter. After printing the statement, it throws the cursor at the starting of the next line. It is the main() difference between the println() and the print() method.

Syntax: system.out.println();

Here System.out.pritnln()/print() is used to print an argument that is passed to it. The statement can be broken into 3 parts which can be understood separately as:

1. System: It is a final class defined in the java.lang.package.
2. out: This is an instance of PrintStream type, which is a public and static member field of the System class.
3. println()/print(): As all instance of PrintStream class have a public method

println()/print(), hence we can invoke the same on out as well.

Program:

```
public class Solution {
    public static void main(String[] args)
    {
        System.out.println("Hello, World.");
        System.out.println("Hello, Java.");
    }
}
```

Output:

```
Hello, World.
Hello, Java.
```

Conclusion:

We printed two lines of output:

1. Print Hello, World. On the first line
2. Print Hello, Java. On the first line

Practical No.: 2

JAVA STDIN AND STDOUT

Aim: Program to read data from stdin and then print them to stdout.

Tools used: www.hackerrank.com

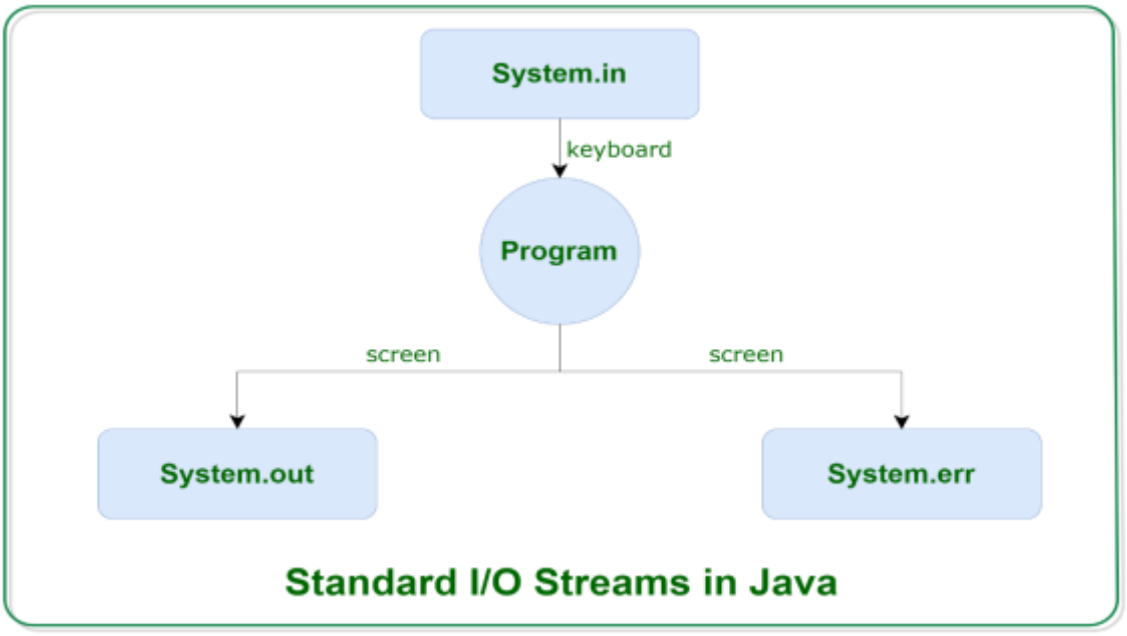
Theory:

Java brings various Streams with its I/O package that helps the user to perform all the input-output operations. These streams support all the types of objects, data-types, characters, file etc to fully execute the I/O operations.

3 Standard streams that java has to provide which are also most common in use

1. System.in: This is the standard input stream that is used to read characters from the keyboard or any other standard input device.

2. System.out: This is the standard output stream that is used to produce the result of a program on an output device like the computer screen.



The list of print functions that we use:

- print(): This method in java is used to display a text on the console. This text is passed as the parameter to this method in the form of string. This method prints the text on the console and the cursor remains at the end of the text at the console.

  Syntax: System.out.print(*parameters*);

- println(): This method in java is also used to display a text on the console.

It prints the text on the console and the cursor moves to the start of the next line at the console. The next printing takes place from the next line.
Syntax: System.out.println(*parameters*);

3. System.err: This is the standard error stream that is used to output all the error data that a program might throw, on a computer screen or any standard output device.

Java User Input:

The scanner class is used to get user input, and it is found in the java.util package. To use the scanner class, create an object of the class and use any of the available methods found in the scanner class documentation.

Input Types:

Method	Description
nextBoolean()	Reads a boolean value from the user
nextByte()	Reads a byte value from the user
nextDouble()	Reads a double value from the user
nextFloat()	Reads a float value from the user
nextInt()	Reads a int value from the user
nextLine()	Reads a String value from the user
nextLong()	Reads a long value from the user
nextShort()	Reads a short value from the user

Program:

```
import java.util.*; public

class Solution {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int a = scan.nextInt();
        System.out.println(a);
```

```
int b = scan.nextInt();
System.out.println(
b);
```

```
        int c = scan.nextInt();
        System.out.println( c);
    }
}
```

Output:

```
42
100
125
```

Conclusion:

1. We read 3 integers from stdin and then printed them to stdout. Each integer printed on a new line.
2. There are three lines of output:
    a. On the first line, print String: followed by the unaltered String read from stdin.
    b. On the second line, print Double: followed by the unaltered double read from stdin.
    c. On the third line, print Int: followed by the unaltered integer read from stdin.

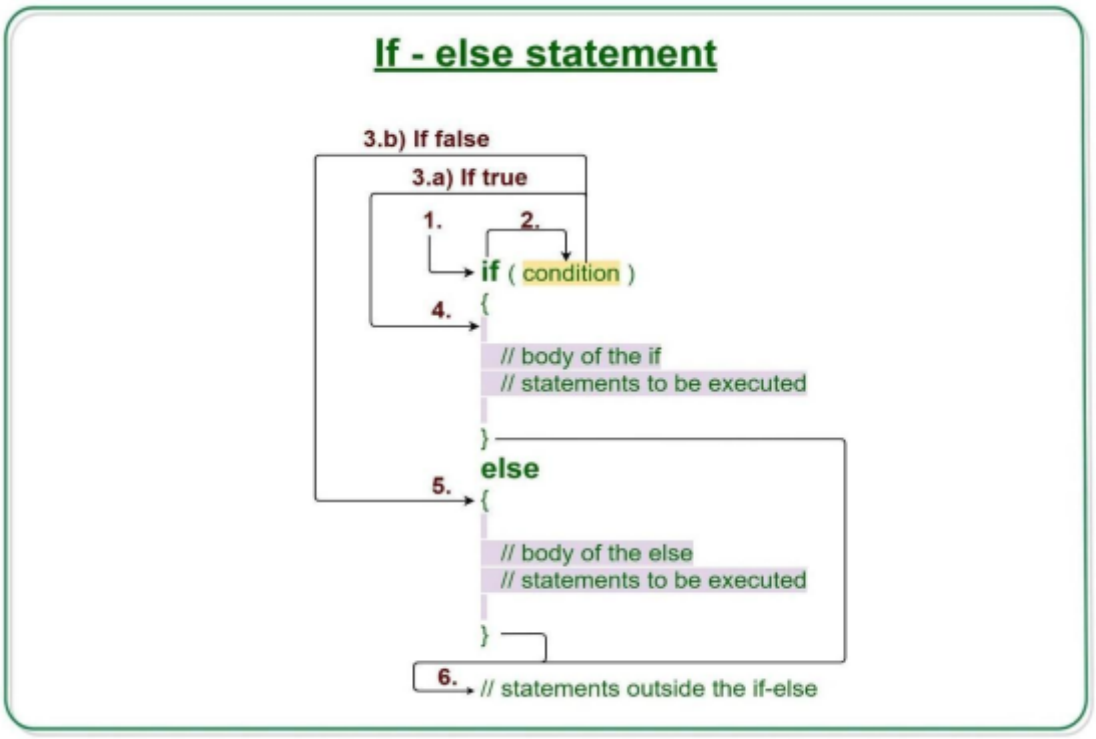
Practical No.:
3 JAVA
IF-ELSE

Aim: Program by using *if-else* conditional statements to automate decision-making processes.

Tools used: www.hackerrank.com

Theory:

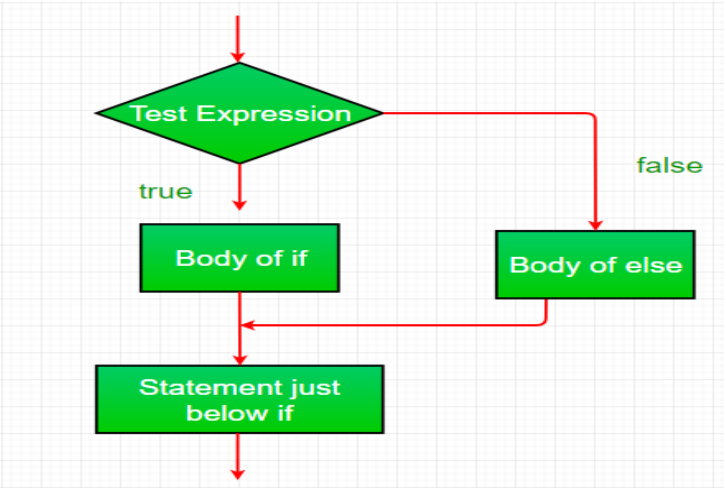
Decision Making in Java helps to write decision-driven statements and execute a particular set of code based on certain conditions. The if statement alone tells us that if a condition is true it will execute a block of statements and if the conditoion is false it won't. but what if we want to do statements else if the condition is false.



Syntax:

```
If(condition)
{
    //Executes this block if condition is true
}
else
{
    //Executes this block if condition is false
}
```

Flow Chart:



Working of if-else Statements

1. Control falls into the if block
2. The flow goes to the condition
3. Condition is tested
    a. If condition is true, go to the step 4.
    b. If condition is false, got to the step 5.
4. The if-block or the body inside the if is executed.
5. The else block or the body inside the else is executed.
6. Flow exits the if-else block.

Program:

```
import java.io.*; import
java.math.*; import
java.security.*; import
java.text.*; import
java.util.*;
import java.util.concurrent.*; import
java.util.regex.*;

public class Solution
{
 private static final Scanner scanner = new Scanner(System.i
n);
 public static void main(String[] args)

{
```

```
        int N = scanner.nextInt(); if(N%2
        == 0 && N>=2 && N<=5)
        {
            System.out.println("Not Weird");

        }
        else if(N%2 == 0 && N>20)
        {
            System.out.println("Not Weird");
        }
        else if(N%2 == 0 && N>=6 && N<=20)
        {
            System.out.println("Weird");
        }
        else if(N%2 != 0)
        {
            System.out.println("Weird");
        }
        }
}
```

Output:

Sample Input 0

3

Sample Output 0

Weird

Sample Input 1

24

Sample Output 1

Not Weird

Conclusion:

Printed weird when the number is odd and not weird when the number is even.

Practical No.: 4

JAVA OUTPUT FORMATTING

Aim: Program to print formatted output.

Tools used: www.hackerrank.com

Theory:

In Competitive Programming, it is essential to print the output in a given specified format. There are different ways in which we can format output in java. Some of them are given below:

- Using System.out.printf()
- Using Decimalformat class
- Using SimpleDateFormat class(for formatting Dates)

Here in this experiment Formatting output done by using System.out.println(), the purpose of this experiment is to understand the formatting of output using print method.

For example:

Input format: Every line of input will contain a *String* followed by an *integer*. Each *String* will have a maximum of 10 alphabetic characters, and each *integer* will be in the inclusive range from 0 to 999.

Output Format: In each line of output there should be two columns:

The first column contains the String and is left justified using exactly 15 characters. The second column contains the integer, expressed in exactly 3 digits; if the original input has less than three digits, you must pad your output's leading digits with zeroes.

Program:

```
import java.util.Scanner; public
class Solution {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in); int
        i = sc.nextInt();
        double d = sc.nextDouble();
        sc.nextLine();
        String s = sc.nextLine(); sc.close();
        System.out.println("String: " + s);
        System.out.println("Double: " + d);
        System.out.println("Int: " + i);
```

```
        }
}
```

Output:

Sample Input

42

3.1415

Welcome to HackerRank's Java tutorials!

Sample Output

String: Welcome to HackerRank's Java
tutorials! Double: 3.1415
Int: 42

Conclusion:

Each *String* is left-justified with trailing whitespace through the first 15 characters. The leading digit of the *integer* is the 16th character, and each *integer* that was less than 3 digits now has leading zeroes.

Practical No.: 5

JAVA LOOPS

Aim: Program to do some simple math using loops.

Tools used: www.hackerrank.com

Theory:

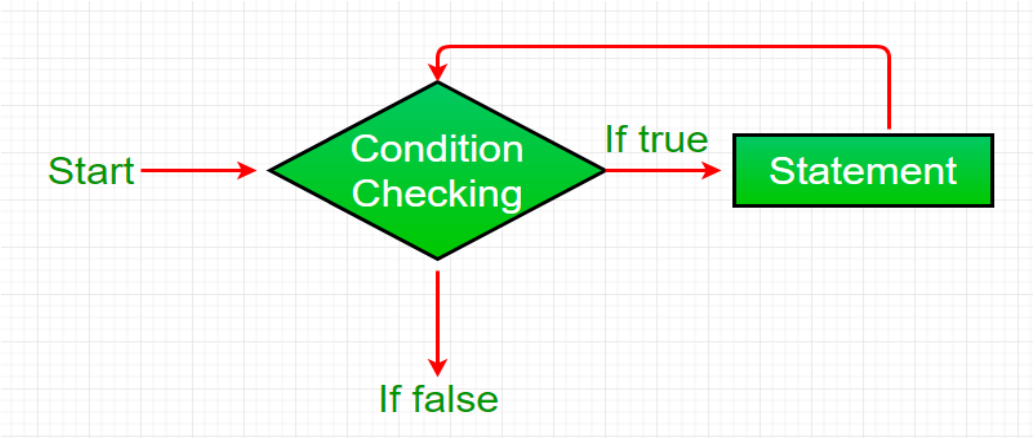
Looping in programming languages is a feature which facilitates the execution of a set of instructions/functions repeatedly while some condition evaluates to true. Java provides three ways for executing the loops. While all the ways provide similar basic functionality, they differ in their syntax and condition checking time.

While loop: A while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The while loop can be thought of as a repeating if statement.

Syntax:

```
while(Boolean condition)
{
        Loop statements…
}
```

Flow Chart:



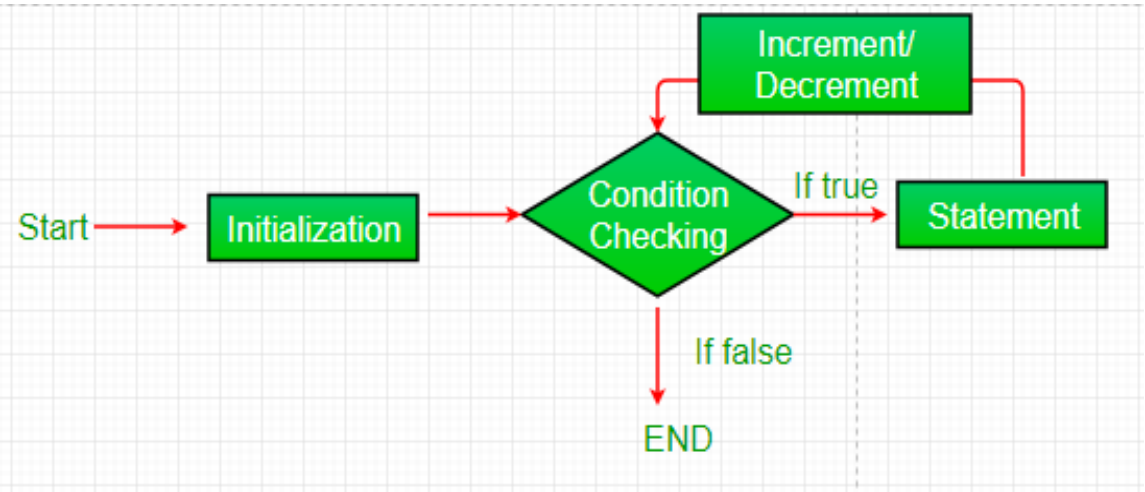
While loop starts with the checking of Boolean condition. If it evaluated to true, then the loop body statements are executed otherwise first statement following the loop is executed. Once the condition is evaluated to true, the statements in the loop body are executed. Normally the statements contain an update value for the variable being processed for the next iteration. When the condition becomes false, the loop terminates which marks the end of its life cycle.

For loop: It provides a concise way of writing the loop structure. Unlike a while loop, a for statement consumes the initialization, condition and increment/decrement in one line thereby providing a shorter, easy to debug structure of looping.

Syntax:

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

Flow Chart:

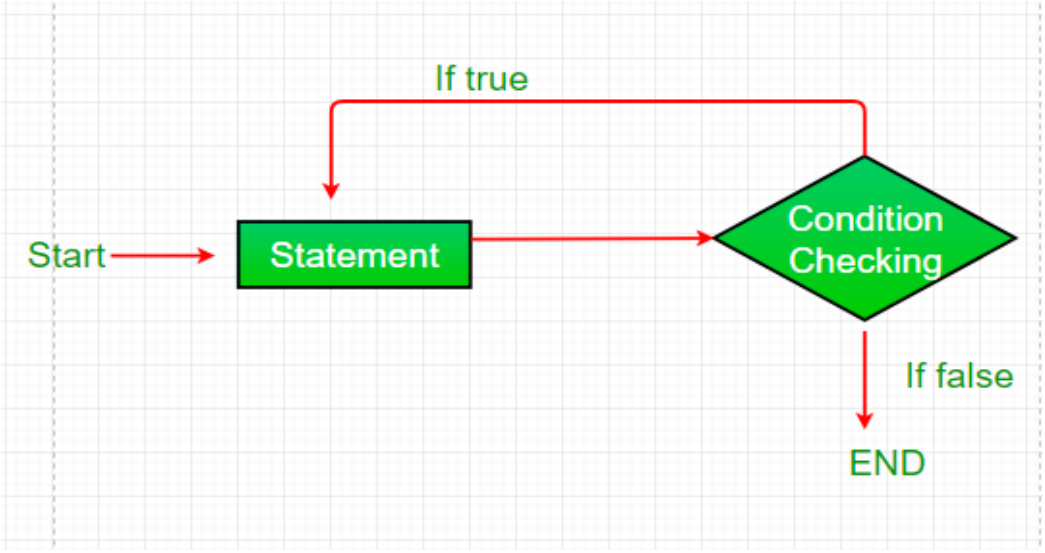


- Initialization condition: Here, we initialize the variable in use. It marks the start of a for loop. An already declared variable can be used or a variable can be declared, local to loop only.
- Testing condition: it is used for testing the exit condition for a loop. It must return a Boolean value. It is also an Entry Control Loop as the condition is checked prior to the execution of the loop statements.
- Statement execution: Once the condition is evaluated to true, the statements in the loop body are executed.
- Increment/ Decrement: It is used for updating the variable for next iteration.
- Loop termination: When the condition becomes false, the loop terminates marking the end of its life cycle.

Do while loop: It is similar to while loop with only difference that it checks for condition after executing the statements.

Syntax:

```
do
{
        Statements…
}
While(condition);
```



Flowchart:

- Do while loop starts with the execution of the statements, and there is no checking of any condition for the first time.
- After the execution of the statements, and update of the variable value, the condition is checked for true or false value. If it is evaluated to true, next iteration of loop starts.
- When the condition becomes false, the loop terminates which marks the end of its life cycle.
- It is important to note that the do-while loop will execute its statements at least once before any condition is checked, and therefore is an example of exit control loop.

Program:

```
import java.io.*; import
java.math.*; import
java.security.*; import
java.text.*; import
```

```
java.util.*;
import java.util.concurrent.*; import
java.util.regex.*;
public class Solution {
    public static void main(String[] args) throws IOExceptio
n {
        Scanner in = new Scanner(System.in); int
        N = in.nextInt();
        for (int i = 1; i <= 10; i++) { System.out.println(N
            + " x " + i + " = " + N * i
);
        }
        in.close();
    }
}
```

Output:

Sample Input

2

Sample Output

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20

Conclusion:

With the help of looping statements, we generated the output for

1. 2's multiplication table
2. using a, b and n, created series S_0 , S_1, …, S_{n-1} .

Practical No.: 6 JAVA

DATA TYPES

Aim: Program to hold integer values and must determine which primitive data types are capable of properly storing that input.

Tools used: www.hackerrank.com

Theory:

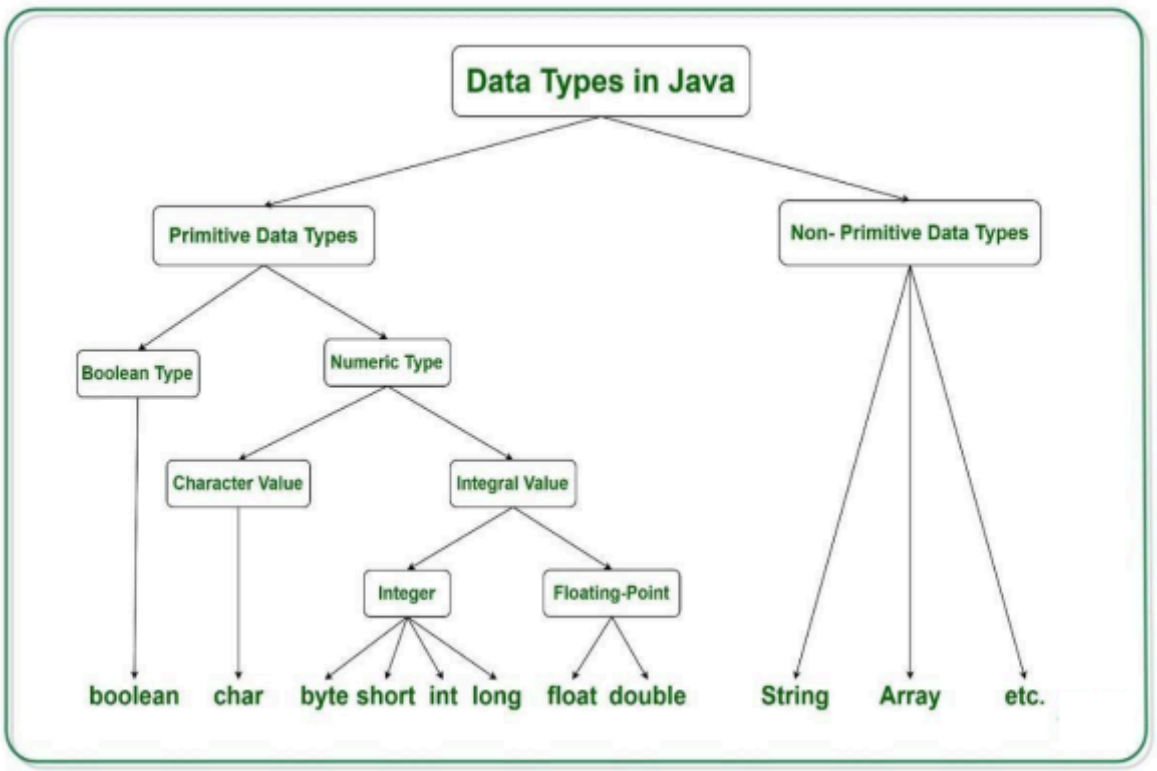
In java, each type of data is predefined as part of the programming language and all constants or variables defined for a given program must be described with one of the data types.

What is data type?

A data type, in programming, is a classification that specifies which type of value a variable has and what type of mathematical, relational or logical operations can be applied to it without causing an error.

Java Has two categories in it:

1. Primitive Data Type: Such as Boolean, Char, Int, Short, Byte, Long, float, and double
2. Non-primitive Data Type: Such as String, Array etc.



Primitive Data Type: These are only single values and have no special capabilities. There are 8 primitive data types i.e. as follows:

Type	Default Value	Size	Range of Values
Boolean	False	1 bit	True, false
Byte	0	8 bits	-128 to 127
Char	\u0000	16 bits	Characters representation of ASCII values 0 to 255
Short	0	16 bits	-32,768 to 32,767
Int	0	32 bits	-2,147,483,648 to 2,147,483,647
Long	0	64 bits	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	0.0	32 bits	Up to 7 decimal digits
Double	0.0	64 bits	Up to 16 decimal digits

1. Boolean Data type: It represents only one bit of information i.e. true or false, which is intended to represent the two truth values of logic and Boolean algebra. The values of type Boolean are not converted implicitly or explicitly to any other type. But the programmer can easily write conversion code.
Syntax: Boolean booleanVar;

2. Byte Data Type: It is an 8-bit signed two's complement integer. The byte data type is useful for saving memory in large arrays.
Syntax: byte byteVar;

3. Short Data Type: It is a 16-bit signed two's complement integer. It is similar to byte, use of a short is to save memory in large arrays, in situations where the memory savings actually matters.
Syntax: short shortVar;

4. Integer Data Type: It is a 32-bit signed two's complement integer. Syntax: int intVar;

5. Long Data Type: is a 64-bit two's complement integer and is useful for those occasions where an int type is not large enough to hold the desired

value.

Syntax: long longVar

6. Float Data Type: Use a float (instead of double) if you need to save memory in large arrays of floating-point numbers. The size of the float data type is 4 bytes (32 bits).

Syntax: float floatVar;

7. Double Data Type: For decimal values, this data type is generally the default choice. The size of the double data type is 8 bytes or 64 bits.

Syntax: double doubleVar;

8. Char Data Type: The char data type is a single 16-bit Unicode character with the size of 2 bytes (16 bits).

Syntax: char charVar;

Non-Primitive Data Type or Reference Data Types

The **Reference Data Types** will contain a memory address of variable values because the reference types won't store the variable value directly in memory. They are strings, objects, arrays, etc.

1. Strings

Strings are defined as an array of characters. The difference between a character array and a string in Java is, that the string is designed to hold a sequence of characters in a single variable whereas, a character array is a collection of separate char-type entities. Unlike C/C++, Java strings are not terminated with a null character.

Syntax: Declaring a string

<String_Type> <string_variable> = "<sequence_of_string>";

2. Class

A class is a user-defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components i.e. **Modifiers, Class name, Superclass, Interfaces.**

3. Object

An Object is a basic unit of Object-Oriented Programming and represents real-life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of: **State, Behavior, and Identity.**

4. Interface

Like a class, an interface can have methods and variables, but the methods declared in an interface are by default abstract.

5. Array

An Array is a group of like-typed variables that are referred to by a common name.

Program:

```
import java.util.*;
import java.io.*;
class Solution{
public static void main(String []args)
{
      Scanner sc = new Scanner(System.in); int
      t = sc.nextInt();
      for (int i = 0; i < t; i++)
      {
        try {
long x = sc.nextLong();
System.out.println(x + " can
be fitted in:"); if (x >=
-128 && x <= 127)
System.out.println("*
byte");
 if (x >= (Math.pow(2, 16 - 1))
&& x <= (Math.pow(2, 16 - 1)
- 1)) System.out.println("*
short");
if (x >= -(Math.pow(2, 32 - 1))
&& x <= (Math.pow(2, 32 - 1)
- 1)) System.out.println("*
int");
if (x >= -(Math.pow(2, 64 - 1))
```

```
&& x <= (Math.pow(2, 64 - 1)
) System.out.println("*
 long");
}
```

```
        catch (Exception e) {
             System.out.println(sc.next() + " can't be fitte d
anywhere.");
            }
      }
         sc.close();
    }
}
```

Output:

Sample Input

5
-150
150000
1500000000
21333333333333333333333333333333333
-100000000000000

Sample Output

-150 can be fitted in:
* short
* int
* long
150000 can be fitted in:
* int
* long
1500000000 can be fitted in:
* int
* long
21333333333333333333333333333333333 can't be fitted anywhere.
-100000000000000 can be fitted in:
* long

Conclusion:

Here for each input variable and appropriate datatype is determined and it prints datatypes that are capable of storing it.

Practical No.: 7

JAVA END-OF-FILE

Aim: Program to read n lines of input until you reach *EOF,* then number and print all n. lines of content.

Tools used: www.hackerrank.com

Theory:

What is end of file in Java?

In computing, end-of-file (EOF) is a condition in a computer operating system where no more data can be read from a data source. The data source is usually called a file or stream.

Here read n lines of input until we reach EOF, then number and print all n lines of content.

Input Format:

Read some unknown n lines of input from stdin(System.in) until it reach EOF; each line of input contains a non-empty String.

Output Format:

For each line, here print the line number, followed by a single space, and then the line content received as input.

Program:

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*; public
class Solution {
    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Prin t
output to STDOUT. Your class should be named Solution. */
        Scanner sc = new Scanner(System.in); int
        i = 1;
        while (sc.hasNext()) {
            System.out.println(i++ + " " + sc.nextLine());
        }
```

```
            sc.close();
        }
}
```

Output:

Sample Input

Hello world I
am a file
Read me until end-of-file.

Sample Output

1 Hello world
2 I am a file
3 Read me until end-of-file.

Conclusion:

Here each line, print the line number, followed by a single space, and then the line content received as input.

Practical No.: 8

JAVA STATIC INITIALIZER BLOCK

Aim: Program to find the Parallelogram and Constraints are -100 ≤B ≤ 100, -100 ≤ H≤ 100.

Tools used: www.hackerrank.com

Theory:

Java supports a special block, called a static block (also called static clause) that can be used for static initialization of a class. The code inside the static block is executed only once: the first time the class is loaded into memory.

What is the static initialization block used for?

Static blocks can be used to initialize static variables or to call a static method.

How to use static initializer block in Java?

To create a static block in java we need to use the static keyword with the block. We can create a static block only at the class level. We can't create a static block inside a method or constructor. It creates with a keyword "static" and uses the curly braces "{"to start and end "}" the block.

Calling of static block in java?

Now comes the point of how to call this static block. So in order to call any static block, there is no specified way as static block executes automatically when the class is loaded in memory. Refer to the below illustration for understanding how static block is called.

Input Format

There are two lines of input. The first line contains B: the breadth of the parallelogram. The next line contains H: the height of the parallelogram.

Constraints

- -100 ≤ B ≤ 100
- -100 ≤ H ≤ 100

Output Format

If both values are greater than zero, then the main method must output the area of the parallelogram. Otherwise, print "java.lang.Exception: Breadth and height must be positive" without quotes.

Program:

```
import java.io.*;
import java.util.*;
```

```
import java.text.*;
import java.math.*;
import java.util.regex.*; public
class Solution {
static boolean flag = true;
static int B,H;
static{
Scanner scan = new Scanner(System.in); B
= scan.nextInt();
scan.nextLine();
H = scan.nextInt();
scan.close(); if(B>0
&& H>0){
    flag = true;
}else if(B<=0 || H<=0){
    flag=false;
    System.out.println("java.lang.Exception: Breadth and hei
ght must be positive");
}
}
public static void main(String[] args){
        if(flag){
            int area=B*H;
            System.out.print(area);
        }
    }//end of main
}//end of class
```

Output:

Sample input 1

1

3

Sample output 1

3

Sample input 2

-1

2

Sample output 2

java.lang.Exception: Breadth and height must be positive

Conclusion:

Here both values are greater than zero, then the main method prints the area of the parallelogram.

Practical No.: 9

JAVA INT TO STRING

Aim: Program to convert integer n into string.

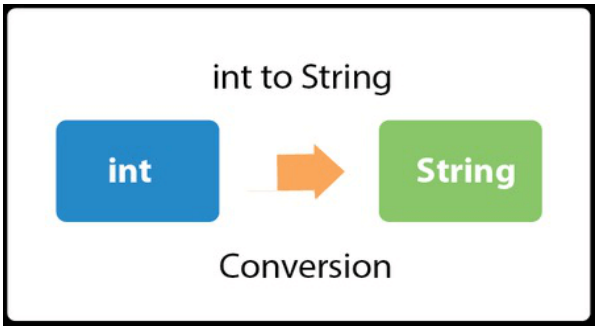
Tools used: www.hackerrank.com

Theory:

We generally counter with such conversion articles because many operations can be performed over a string while we are limited to when it comes to integers. We have a wide varied list of in-built method in the String class that helps us perform hassle-free operations.

Suppose we are required to concatenate two integers then it would become a tedious job as we need to go through as we need to deal with the number system corresponding to which we will be playing mathematics within the number system. But i**n Java, in order to convert integer to string, we have some inbuilt methods and classes which make out work too easy.**

We can convert **int to String in java** using *String.valueOf()* and *Integer.toString()* methods. Alternatively, we can use *String.format()* method, string concatenation operator etc.



Different Methods For Integer to String Conversions

1. Using toString() method of Integer class
2. Using valueOf() method of String class
3. Using Integer(int).toString() method of Integer class
4. Using DecimalFormat Class
5. Using StringBuffer class
6. using StringBuilder class
7. Using concatenation with an empty

string Scenario

1) String.valueOf()

The String.valueOf() method converts int to String. The valueOf() is the static method of String class. The **signature** of valueOf() method is given below:

public static String valueOf(int i)

2) Integer.toString()

The Integer.toString() method converts int to String. The toString() is the static method of Integer class. The **signature** of toString() method is given below:

public static String toString(int i)

3) String.format()

The String.format() method is used to format given arguments into String. It is introduced since Jdk 1.5.

public static String format(String format, Object... args)

Program:

```
import java.util.*; import
java.security.*; public
class Solution
{
public static void main(String[] args)
{
  DoNotTerminate.forbidExit();
  try {
   Scanner in = new Scanner(System.in); int
   n = in .nextInt();
   in.close(); String
   s=n+"";
   if (n == Integer.parseInt(s))
   {
    System.out.println("Good job");
   } else
   {
    System.out.println("Wrong answer.");
```

```
    }
   } catch (DoNotTerminate.ExitTrappedException e)
   {
    System.out.println("Unsuccessful Termination!!");
   }
  }
}

class DoNotTerminate {
public static class ExitTrappedException extends SecurityExc
eption
{
  private static final long serialVersionUID = 1;
}
 public static void forbidExit() {
  final SecurityManager securityManager = new SecurityManage
r() {
   @Override
   public void checkPermission(Permission permission) { if
    (permission.getName().contains("exitVM")) {
     throw new ExitTrappedException();
    }
     }
    };
  System.setSecurityManager(securityManager);
 }
}
```

Output:

Sample Input 0

100

Sample Output 0 Good

job

Conclusion:

Here conversion from integer to string is done

Practical No.:10

JAVA DATE AND TIME

Aim: Program for input containing the space separated month, day and year, respectively, in MM DD YYYY format.

Tools used: www.hackerrank.com

Theory:

In Java, there is a built-in class known as the Date class and we can import java.time package to work with date and time API. Here we are supposed to print current date and time. There can be multiple ways to print the current date and time.

Different Ways to Get Current Date and Time

1. **Using Date Class**
2. **Using get() method of Calendar class**
3. **Using calendar and formatter class to print the current dates in a specific format.**

Method 1: Using Date Class

```
Date d1 = new Date();
```

Method 2: Using get() method of Calendar class

getInstance() method is generally used to get the time, date or any required some belonging to Calendar year.

```
Calendar c = Calendar.getInstance();
```

Method 3: Using calendar and formatter class to print the current dates in a specific format.

SimpleDateFormat ft = new SimpleDateFormat("dd-MM-yyyy");

Input Format

A single line of input containing the space separated month, day and year, respectively, in MM DD YYYY format.

Output Format

It should display Day

Program:

```
import java.io.*; import
java.math.*; import
java.security.*; import
```

```
java.text.*; import
java.util.*;
import java.util.concurrent.*;
import java.util.regex.*; import
java.io.*;
import java.math.*; import
java.security.*; import
java.text.*; import
java.util.*;
import java.util.concurrent.*;
import java.util.regex.*; class
Result
{
      public static String findDay(int month, int day, int y
ear)       {

         Calendar cal = Calendar.getInstance();
         cal.set(Calendar.MONTH,month-1);
         cal.set(Calendar.DAY_OF_MONTH,day);
         cal.set(Calendar.YEAR,year);
String dayOfWeek = cal.getDisplayName(Calendar.DAY_OF_WEEK,
Calendar.LONG, Locale.US).toUpperCase();
         return dayOfWeek;
    }
}

public class Solution
{
    public static void main(String[] args) throws IOExceptio
n
     {
         BufferedReader bufferedReader = new BufferedReader(n
ew InputStreamReader(System.in));
         BufferedWriter bufferedWriter = new BufferedWriter(n ew
FileWriter(System.getenv("OUTPUT_PATH")));
```

```
        String[] firstMultipleInput = bufferedReader.readLin
e().replaceAll("\\s+$", "").split(" ");

        int month = Integer.parseInt(firstMultipleInput[0]);
        int day = Integer.parseInt(firstMultipleInput[1]); int
        year = Integer.parseInt(firstMultipleInput[2]); String
        res = Result.findDay(month, day, year);
        bufferedWriter.write(res);
        bufferedWriter.newLine();
        bufferedReader.close();
        bufferedWriter.close();
    }
}
```

Output:

Sample

Input 08 05

2015

Sample Output

WEDNESDAY

Conclusion:

Here it displayed Day

Practical No.: 11

JAVA CURRENCY FORMATTER

Aim: Program for a single double-precision number denoting payment.

Tools used: www.hackerrank.com

Theory:

Here given a double-precision number, payment, denoting an amount of money, use the NumberFormat class' getCurrencyInstance method to convert payment into the US, Indian, Chinese, and French currency formats. Then print the formatted values as follows: US: formattedPayment India:
formattedPayment China:
formattedPayment France:
formattedPayment
Where formattedPayment is payment formatted according to the appropriate Locale's currency.

Input Format

A single double-precision number denoting payment.

Constraints

- $0 \leq \text{payment} \leq 10^9$

Output Format

On the first line, print US: u where u is payment formatted for US currency.
On the second line, print India: i where i is payment formatted for Indian currency. On the third line, print China: c where c is payment formatted for Chinese currency. On the fourth line, print France: f, where f is payment formatted for French currency.

Program:

```
import java.util.*; import
java.text.*;
public class Solution
{
    public static void main(String[] args) { Scanner
        scanner = new Scanner(System.in); double
```

```
        payment = scanner.nextDouble();
        scanner.close();

        // Write your code here.
         String us = NumberFormat.getCurrencyInstance(Locale
.US).format(payment);
         String india = NumberFormat.getCurrencyInstance(new
 Locale("en","in")).format(payment);
          String china = NumberFormat.getCurrencyInstance(Lo
cale.CHINA).format(payment);
           String france = NumberFormat.getCurrencyInstance(
Locale.FRANCE).format(payment);
        System.out.println("US: " + us);
        System.out.println("India: " + india);
        System.out.println("China: " + china);
        System.out.println("France: " + france);
    }
}
```

Output:

Sample Input

12324.134

Sample Output

US: $12,324.13 India:

Rs.12,324.13

China: ￥12,324.13 France:

12 324,13 €

Conclusion:

Here, On the first line, print US: u where u is payment formatted for US currency. On the second line, print India: i where i is payment formatted for Indian currency. On the third line, print China: c where c is payment formatted for Chinese currency. On the fourth line, print France: f, where f is payment formatted for French currency.

Practical No.: 12

JAVA STRINGS INTRODUCTION

Aim: Program for a String where the first line contains a string A, the second line contains another string B. The strings are comprised of only lowercase English letters. **Tools used:** www.hackerrank.com

Theory:

Why Java uses the concept of string literal?

To make Java more memory efficient (because no new objects are created if it exists already in the string constant pool).

Using new keyword

- String s = new String("Welcome");
- In such a case, JVM will create a new string object in normal (non-pool) heap memory and the literal "Welcome" will be placed in the string constant pool. The variable s will refer to the object in the heap (non-pool).

Syntax:

<String_Type> <string_variable> = "<sequence_of_string>";

Memory allotment of String

Whenever a String Object is created as a literal, the object will be created in the String constant pool. This allows JVM to optimize the initialization of String literal.

Example:

String str = "CSE";

The string can also be declared using a **new** operator i.e. dynamically allocated. In case of String are dynamically allocated they are assigned a new memory location in the heap. This string will not be added to String constant pool.

Example:

String str = new String("CSE");

Program:

```
import java.io.*; import
java.util.*; public class
Solution {
public static void main(String[] args) { Scanner
        sc=new Scanner(System.in); String
```

```
        A=sc.next(); String
        B=sc.next();
        System.out.println(A.length()+B.length());
System.out.println(A.compareTo(B)>0?"Yes":"No");
System.out.println(A.substring(0,    1).toUpperCase()+A.substri
ng(1, A.length())+" "+B.substring(0, 1).toUpperCase()+B.subs
tring(1, B.length()));
    }
}
```

Output:

Sample Input 0

hell

o

java

Sample Output 0

9

No

Hello Java

Conclusion:

Here sum of length is displayed, and it compares both the given string and prints whether greater than or not, and displays concatenated string when capitalize the first letter of both string.

Practical No.: 13

JAVA SUBSTRING

Aim: Print a substring consisting of all characters in the inclusive range from start to end-1.

Tools used: www.hackerrank.com

Theory:

A part of String is called **substring**. In other words, substring is a subset of another String. Java String class provides the built-in substring() method that extract a substring from the given string by using the index values passed as an argument. In case of substring() method startIndex is inclusive and endIndex is exclusive. Suppose the string is "**computer**", then the substring will be com, compu, ter, etc.

The two methods in Substring are:

1. public String substring(int startIndex):

This method returns new String object containing the substring of the given string from specified startIndex (inclusive).

2. public String substring (int startIndex, int endIndex):

This method returns new String object containing the substring of the given string from specified startIndex to endIndex.

Input Format

The first line contains a single string denoting s. The second line contains two space-separated integers denoting the respective values of start and end.

Output Format

Print the substring in the inclusive range from start to end-1.

Program:

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*; public
```

```
class Solution {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String S = in.next(); int
        start = in.nextInt(); int
        end = in.nextInt();
        System.out.print(S.substring(start,end));
    }
}
```

Output:

Sample Input

Helloworld 3 7

Sample

Output lowo

Conclusion:

Here we printed the substring in the inclusive range from start to end -1.

Practical No.: 14

JAVA SUBSTRING COMPARISONS

Aim: Program to Print Lexicographical order.

Tools used: www.hackerrank.com

Theory:

There are three ways to compare String in Java:

1. By Using equals() Method: The String class equals() method compares the original content of the string. It compares values of string for equality. String class provides the following two methods:
    - **public boolean equals(Object another)** compares this string to the specified object.
    - **public boolean equalsIgnoreCase(String another)** compares this string to another string, ignoring case.
2. By Using == Operator: The == operator compares references not values.
3. By compareTo() Method: The String class compareTo() method compares values lexicographically and returns an integer value that describes if first string is less than, equal to or greater than second string.

Given a string, s, and an integer, k, complete the function so that it finds the lexicographically smallest and largest substrings of length k.

Function Description

Complete the getSmallestAndLargest function in the editor below. getSmallestAndLargest has the following parameters:

- string s: a string
- int k: the length of the substrings to find

Returns

- string: the string ' + "\n" + ' where and are the two substrings.

Input Format

The first line contains a string denoting s. The second line contains an integer denoting k.

Constraints

- $1 \leq |s| \leq 1000$
- s consists of English alphabetic letters only (i.e., [a-z, A-Z]).

Program:

```
import java.util.Scanner; public
class Solution {
    public static String getSmallestAndLargest(String s, int k)
                {                                   S
                                                    t
                                                    r
                                                    i
                                                    n
<>();                                               g
                                                    s
                                                    m
                                                    a
                                                    l
                                                    l
                                                    e
                                                    s
                                                    t
                                                    =
                                                    "
                                                    "
                                                    ;
                                                    S
                                                    t
                                                    r
                                                    i
                                                    n
                                                    g
                                                    l
                                                    a
                                                    r
                                                    g
                                                    e
                                                    s
                                                    t
```

```
"
;
  java.util.List<String> a = new
  java.util.ArrayList

for(int
        i
        =
        0
        ;
        i
        <
        s
        .
        l
        e
        n
        g
        t
        h
        (
        )
        -
        k
        +
        1
        ;
        i
                                    +
                                    +
                                    )
                                    {
                                    a
                                    .
                                    a
                                    d
                                    d
                                    (
                                    s
                                    .
                                    s
                                    u
                                    b
                                    s
                                    t
                                    r
                                    i
                                    n
                                    g
                                    (
                                    i
                                    ,
                                    i
                                    +
                                    k
                                    )
                                    )
                                    ;
                              }
                              java
                               .ut
                               il.
                               Col
                          }    lec
                               tio
                               ns.
```

Output:

```
sor                                    0);
YSPU(a                                 largest
);                                     =
sma                                    a.get(a.s
lle                                    ize()-1);
st                                     return
=                                      smallest
a.g                                    + "\n" +
et(                                    largest;
```

Sample Input 0

welcometojava 3

Sample Output 0

ava wel

Conclusion:

Here we printed lexicographically smallest and largest substrings of entered length.

Practical No.: 15

JAVA STRING REVERSE

Aim: Program to print Yes if it is a palindrome, print No otherwise.

Tools used: www.hackerrank.com

Theory:

There are many ways to reverse String in Java. We can reverse String using StringBuffer, StringBuilder, iteration etc. Let's see the ways to reverse String in Java.

By StringBuilder / StringBuffer

Using built in reverse() method of the StringBuilder class: String class does not have reverse() method, we need to convert the input string to StringBuilder, which is achieved by using the append method of StringBuilder. After that, print out the characters of the reversed string by scanning from the first till the last index.

Using StringBuffer: String class does not have reverse() method, we need to convert the input string to StringBuffer, which is achieved by using the reverse method of StringBuffer.

A palindrome is a word, phrase, number, or other sequence of characters which reads the same backward or forward.

Given a string A, print Yes if it is a palindrome, print No otherwise.

Constraints

A will consist at most 50 lower case english letters.

Program:

```
import java.io.*;
import java.util.*;
import java.util.Scanner;
public class Solution {
    public static void main(String[] args) {
         Scanner sc=new Scanner(System.in);
        String A=sc.next();
        /* Enter your code here. Print output to STDOUT. */
```

```
        String rev="";
        for(int i=A.length()-1;i>=0;i--)
        {
            rev = rev+A.charAt(i);
        }
        if(rev.equals(A))
         System.out.print("Yes"); else
         System.out.print("No");

    }
}
```

Output:

Sample Input

madam

Sample Output Yes

Conclusion:

Here we printed yes for the given string is palindrome

Practical No.: 16

JAVA ANAGRAMS

Aim: Program to Check Two strings, a and b, are called anagrams if they contain all the same characters in the same frequencies.

Tools used: www.hackerrank.com

Theory:

Anagram

The dictionary meaning of the word **anagram** is a word or phrase formed by rearranging the letters.

Two strings are said to be **anagrams** if they make a meaningful word by rearranging or shuffling the letters of the string. In other words, we can say that two strings are **anagrams** if they contain the same characters but in a different order.

Example of Anagram

There are several anagram words some of them are:

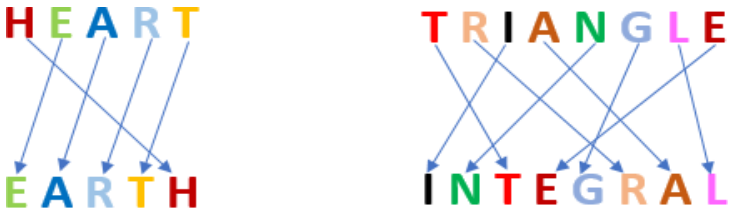
LISTEN - > SILENT

HEART - > EARTH

LIVES - > ELVIS

KEEP -> PEEK TABLE

- >BLEAT



How to check two strings are anagram or not?

- Read or initialize two strings **str1** and **str2**.
- Find the length of both the strings.
- Compare the length of the strings.
    i. If length is not equal, print strings are not an anagram.
    ii. Else, do the following:

a. Convert the string into a character array.
b. Sort both the arrays by using the **sort()** method.
c. After sorting, compare the strings by using the **equals()** method. Store the value in a Boolean variable (**status**) returned by the equals() method.

iii. Pass the variable in the if statement. If it returns true, the given strings are **anagram**. Else, not an **anagram**.

Input Format

The first line contains a string a.

The second line contains a string b.

Constraints

- 1≤length(a), length(b)≤ 50
- Strings a and b consist of English alphabetic characters.
- The comparison should NOT be case sensitive.

Program:

```
import java.util.Scanner; public
class Solution {
    static boolean isAnagram(String a, String b) { String
        s1 = a;
        String s2 = b;
        s1=s1.toLowerCase();
        s2=s2.toLowerCase();
        if(s1.length()==s2.length())
        {
            int[] arr = new int[256];
            int[] brr = new int[256];
            for (int i = 0; i < s1.length(); i++) { arr[(int)
                s1.charAt(i)] += 1;
                brr[(int) s2.charAt(i)] += 1;
            }
            for (int i = 0; i < 256; i++) { if
                (arr[i] != brr[i])
                    return false;
```

```
                                        }
                                        return true;
        }
        else
        {
                                        return false;
        }
                }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String a = scan.next();
        String b = scan.next();
        scan.close();
        boolean ret = isAnagram(a, b);
        System.out.println( (ret) ? "Anagrams" : "Not Anagra
ms");
    }
}
```

Output:

Sample

Input

anagram

margana

Sample Output Anagrams

Conclusion:

Here two strings contain all the same letters in the same frequencies, so we print "Anagrams".

Practical No.: 17

JAVA STRING TOKENS

Aim: Print the number of tokens, followed by each token on a new line.

Tools used: www.hackerrank.com

Theory:

What is a token in Java string?

String tokenization is a process where a string is broken into several parts. Each part is called a token.

For example, if "I am going" is a string,

the discrete parts—such as "I", "am", and "going"—are the tokens. Java provides ready classes and methods to implement the tokenization process.

Constructors of StringToken: Let us consider 'str' as the string to be tokenized:

1. StringTokenizer(String str): default delimiters like newline, space, tab, carriage return, and form feed.

2. StringTokenizer(String str, String delim): delim is a set of delimiters that are used to tokenize the given string.

Methods of String Tokenizer class

Method	Action performed
countTokens()	Returns the number of tokens present
hasMoreToken()	Tests if tokens are present for the String tokenizer's string
nextElement()	Returns an object rather than String
hasMoreElements()	Returns the same value as hasMore Token
nextToken()	Returns the next token from the given StringTokenizer.

Given a string, s, matching the regular expression [A-Za-z !,?._'@]+, split the string into *tokens*. We define a token to be one or more consecutive English alphabetic letters. Then, print the number of tokens, followed by each token on a new line.

Note: You may find the String.split method helpful in completing this challenge.

Input Format

A single string, s.

Constraints

- $1 \leq$ length of s $\leq 4.10^5$
- S is composed of *any* of the following: English alphabetic letters, blank spaces, exclamation points (!), commas (,), question marks (?), periods (.), underscores (_), and apostrophes ('), and at symbols (@).

Output Format

On the first line, print an integer, n, denoting the number of tokens in string s (they *do not* need to be unique). Next, print each of the n tokens on a new line in the same order as they appear in input string s.

Program:

```
import java.io.*; import
java.util.*; public class
Solution {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String s = scan.nextLine();
        // Write your code here. s
        = s.trim();
        if (s.length() == 0) {
            System.out.println(0);
        } else {
            String[] strings = s.split("['!?,._@ ]+");
            System.out.println(strings.length);
            for (String str : strings)
                System.out.println(str);
        scan.close();
    }
    }
```

}

Output:

Sample Input

He is a very very good boy, isn't he?

Sample Output

10

He

is a

very

very

good

boy

isn t

he

Conclusion:

We print the number of tokens, followed by each token on a new line.

Practical No.: 18

JAVA PATTERN SYNTAX CHECKER

Aim: Program for pattern syntax checker.

Tools used: www.hackerrank.com

Theory:

Regular Expressions or Regex in Java is an API for defining String patterns that can be used for searching, manipulating, and editing a string in java. Email validation and passwords are a few areas of strings where Regex is widely used to define the constraints. Regular Expressions are provided under java.util.regex package. This consists of 3 classes and 1 interface as depicted below in tabular format as follows:

Sr. No.	Class/Interface	Description
1.	Pattern Class	Used for defining patterns
2.	Matcher Class	Used for performing match operations on text using patterns
3.	PatternSyntaxException Class	Used for indicating syntax error in a regular expression pattern
4.	MatchResult Interface	Used for representing the result of a match operation

Pattern Class:

This class is a compilation of regular expressions that can be used to define various types of patterns, providing no public constructors. This can be created by invoking the compile() method which accepts a regular expression as the first argument, thus returns a pattern after execution.

The methods in Pattern Class:

1. compile(String regex): It is used to compile the given regular expression into a pattern.
2. compile(String regex, int flags): It is used to compile the given regular expression into pattern with the given flags.

Input Format

The first line of input contains an integer N, denoting the number of test cases. The next N lines contain a string of any printable characters representing the pattern of a regex.

Output Format

For each test case, print Valid if the syntax of the given pattern is correct. Otherwise, print

Invalid. Do not print the quotes.

Program:

```
import    java.util.Scanner;
import    java.util.regex.*;
public class Solution
{
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        int testCases = Integer.parseInt(in.nextLine());
        while(testCases>0){
            String pattern = in.nextLine(); try
                { Pattern.compile(pattern);
                System.out.println("Valid");
            } catch(PatternSyntaxException e) {
                System.out.println("Invalid");
            }
            testCases--;
        }
    }
}
```

Output:

Sample Input

3

([A-Z])(.+)

[AZ[a-z](a-z)

batcatpat(nat

Sample

Output Valid

Invali

d

Invali

d

Conclusion:

Here we printed valid for the syntax of the given pattern is correct and invalid for the syntax of the given pattern is not correct.

Practical No.: 19

JAVA REGEX

Aim: Program to write a regular expression to find the valid IPs for a provided string containing any combination of ASCII.

Tools used: www.hackerrank.com

Theory:

Here we supposed to create a class called MyRegex, which will contain a string pattern. And here we need to write a regular expression and assign it to the pattern such that it can be used to validate an IP address.

Use the following definition of an IP address:

IP address is a string in the form "A.B.C.D ", where the value of A, B, C, and D may range from 0 to 255. Leading zeros are allowed. The length of A, B, C, or D can't be greater than 3.

Some valid IP address:

000.12.12.034

121.234.12.12

23.45.12.56

Some invalid IP address:

000.12.234.23.23

666.666.23.23

.213.123.23.32

23.45.22.32.

I.Am.not.an.ip

In this problem, we have to write a regular expression to find the valid IPs for a provided string containing any combination of ASCII.

Program:

```
import java.util.regex.Matcher; import
java.util.regex.Pattern; import
java.util.Scanner;
class Solution{
```

```
    public static void main(String[] args){ Scanner in = new
        Scanner(System.in); while(in.hasNext()){
           String IP = in.next();
System.out.println(IP.matches(new MyRegex().pattern));
}
}
}
class MyRegex {
String num = "([01]?\\d{1,2}|2[0-4]\\d|25[0-5])";
String pattern = num + "." + num + "." + num + "." + num;
}
```

Output:

Sample Input

000.12.12.034

121.234.12.12

23.45.12.56

00.12.123.123123.123

122.23

Hello.IP

Sample

Output true

true

true

false

false

false

Conclusion:

We write regular expression and assign it to the pattern such that it can be used to validate an IP address and IP address is correct, it printed true.

Practical No.: 20

JAVA REGEX 2

Aim: Program to remove instances of words that are repeated more than once but retain the first occurrence of any case-insensitive repeated word.

Tools used: www.hackerrank.com

Theory:

Here we use regular expressions (Regex) to remove instances of words that are repeated more than once, but retain the first occurrence of any case-insensitive repeated word. For example, the words love and to are repeated in the sentence I love Love to To tO code. Here we supposed to convert I love Love to To tO code into I Love to code?

To solve this challenge, complete the following three lines:

1. Write a RegEx that will match any repeated word.
2. Complete the second *compile* argument so that the compiled RegEx is case- insensitive.
3. Write the two necessary arguments for replaceAll such that each repeated word is replaced with the very first instance the word found in the sentence. It must be the exact first occurrence of the word, as the expected output is case-sensitive.

Input Format

The following input is handled for you the given stub code:

The first line contains an integer, n, denoting the number of sentences.

Each of the n subsequent lines contains a single sentence consisting of English alphabetic letters and whitespace characters.

Constraints

- Each sentence consists of at most 10^4 English alphabetic letters and whitespaces.
- $1 \leq n \leq 100$

Output Format

Stub code in the editor prints the sentence modified by the replaceAll line to stdout. The modified string must be a modified version of the initial sentence where all repeat occurrences of each word are removed.

Program:

```
import java.util.Scanner; import
java.util.regex.Matcher; import
java.util.regex.Pattern; public
class DuplicateWords {
    public static void main(String[] args) { String
        regex = "\\b(\\w+)(?:\\W+\\1\\b)+";
        Pattern p = Pattern.compile(regex, Pattern.CASE_INSE
NSITIVE);
        Scanner in = new Scanner(System.in);
        int  numSentences  =  Integer.parseInt(in.nextLine());
        while (numSentences-- > 0) {
            String  input  =  in.nextLine();
            Matcher  m  =  p.matcher(input);
            while (m.find()) {
                input = input.replaceAll(m.group(), m.group(
1));
                                                    }
                                                    System.out.println(input);
                                                }
        in.close();
    }
}
```

Output:

Sample Input

5

Goodbye bye bye world world world Sam

went went to to to his business

Reya is is the the best player in eye eye game

in inthe

Hello hello Ab aB

Sample Output

Goodbye bye world

Sam went to his business

Reya is the best player in eye game in

inthe

Hello Ab

Conclusion:

Here Stub code in the editor prints the sentence modified by the replaceAll line to stdout

Practical No.: 21

VALID USERNAME REGULAR EXPRESSION

Aim: Program to check the valid username regular expression.

Tools used: www.hackerrank.com

Theory:

Here we have to update the username policy on your company's internal networking platform. According to the policy, a username is considered valid if all the following constraints are satisfied:

- The username consists of 8 to 30 characters inclusive. If the username consists of less than 8 or greater than 30 characters, then it is an invalid username.
- The username can only contain alphanumeric characters and underscores (_). Alphanumeric characters describe the character set consisting of lowercase characters [a-z], uppercase characters [A-Z], and digits [0-9].
- The first character of the username must be an alphabetic character, i.e., either lowercase character [a-z] or uppercase character[A-Z].

For example:

UserName	Validity
Julia	Invalid; UserName length<8 characters
Samantha	Valid
Samantha_21	Valid
1Samantha	Invalid; UserName begins with non-alphabetic character
Samantha?10_2A	Invalid; '?' character not allowed

Update the value of regularExpression field in the UsernameValidator class so that the regular expression only matches with valid usernames.

Input Format

The first line of input contains an integer n, describing the total number of usernames. Each of the next n lines contains a string describing the username. The locked stub code reads the inputs and validates the username.

Constraints

- $1 \leq n \leq 100$
- The username consists of any printable characters.

Output Format

For each of the usernames, the locked stub code prints Valid if the username is valid; otherwise Invalid each on a new line.

Program:

```
import java.util.Scanner;
class UsernameValidator {
    public static final String regularExpression = "^[A-Za-
z]\\w{7,29}$";
}
```

Output:

Sample Input

8
Julia Samantha
Samantha_21
1Samantha
Samantha?10_2
A JuliaZ007
Julia@007
_Julia007

Sample

Output Invalid
Valid
Valid
Invali
d
Invali
d
Valid
Invali
d
Invali
d

Conclusion:

We printed valid for each of the usernames.

Practical No.: 22

TAG CONTENT EXTRACTOR

Aim: Program to print the content enclosed within valid tags.

Tools used: www.hackerrank.com

Theory:

In a tag-based language like *XML* or *HTML*, contents are enclosed between a *start tag* and an *end tag* like `<tag>contents</tag>`. Note that the corresponding *end tag* starts with a `/`.

Given a string of text in a tag-based language, parse this text and retrieve the contents enclosed within sequences of well-organized tags meeting the following criterion:

1. The name of the *start* and *end* tags must be same. The HTML code `<h1>Hello World</h2>` is *not valid*, because the text starts with an `h1` tag and ends with a non-matching `h2` tag.
2. Tags can be nested, but content between nested tags is considered *not valid*. For example,

   in `<h1><a>contents</a>invalid</h1>`, `contents` is *valid* but `invalid` is *not valid*.
3. Tags can consist of any printable characters.

Input Format

The first line of input contains a single integer, N (the number of lines). The N subsequent lines each contain a line of text.

Constraints

- $1 \leq N \leq 100$
- Each line contains a maximum of 10^4 printable characters.
- The total number of characters in all test cases will not exceed 10^6.

Output Format

For each line, print the content enclosed within valid tags. If a line contains multiple instances of valid content, print out each instance of valid content on a new line; if no valid content is found, print None.

Program:

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
```

```
import java.util.regex.*; public
class Solution{
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
         int t = Integer.parseInt(in.nextLine());
      while(t-->0){
          String line = in.nextLine();
          Matcher m = Pattern.compile("<(.+)>(([^<>]+))</\\1
>").matcher(line);
          if (!m.find()) {
              System.out.println("None");
              continue;
          }
          m.reset();
          while (m.find()){
              System.out.println(m.group(2));
          }
       }
    }
 }
```

Output:

Sample Input

4

<h1>Nayeem loves counseling</h1>

<h1><h1>Sanjay has no watch</h1></h1><par>So wait for a while</par>

<Amee>safat codes like a ninja</amee>

<SA premium>Imtiaz has a secret crush</SA premium>

Sample Output

Nayeem loves
counseling Sanjay has no
watch
So wait for a while
None Imtiaz has a
secret crush

Conclusion:

We printed the content enclosed within valid tags.

Practical No.: 23

JAVA BIGDECIMAL

Aim: Here each number must be printed in the exact same format as it was read from stdin, meaning that .1 is printed as .1, and 0.1 is printed as 0.1.

Tools used: www.hackerrank.com

Theory:

Java's BigDecimal class can handle arbitrary-precision signed decimal numbers. Here given an array s of n real number strings, sort them in descending order – but there is more! Each number must be printed in the exact same format as it was read from stdin, meaning that .1 is printed as .1, and 0.1 i.e. printed as 0.1. If two numbers represent numerically equivalent values for example .1 ≡ 0.1, then it must be listed in the same order as they were received as input. Here to complete the code we must rearrange array s's elements according to the instructions above.

Input Format

The first line consists of a single integer, n, denoting the number of integer strings. Each line i of the n subsequent lines contains a real number denoting the value of s_i.

Constraints

- $1 \leq n \leq 200$
- Each s_i has *at most* digits.

Output Format

Locked stub code in the editor will print the contents of array s to stdout. You are only responsible for reordering the array's elements.

Program:

```
Import    java.math.BigDecimal;
import java.util.*;
class Solution{
    public  static  void  main(String  []args){
        Scanner  sc=  new  Scanner(System.in);
        int n=sc.nextInt();
        String  []s=new  String[n+2];
        for(int i=0;i<n;i++){
            s[i]=sc.next();
        }
```

```
        sc.close();
        for(int i=1; i<n ; i++){ for(int
            j=i; j>=1; j--){
                if(new BigDecimal(s[j]).compareTo(new BigDec
imal(s[j-1]))>0){
                    String temp = s[j]; s[j]
                    = s[j-1];
                    s[j-1] = temp;
                }else{
                    break;
                }
            }
        }
        for(int i=0;i<n;i++)
        {
            System.out.println(s[i]);
        }
    }
}
```

Output:

Sample Input

9
-100
50
0
56.6
90
0.12
.12
02.34
000.000

Sample Output

90
56.6
50
02.34
0.12
.12
0
000.000
-100

Conclusion:

We print the contents of array to stdout.

Practical No.: 24

JAVA PRIMALITY TEST

Aim: Program to print if the given number is a prime number, then print prime; otherwise, print not prime.

Tools used: www.hackerrank.com

Theory:

The java.math.BigInteger isProbablePrime(int certainty) method is used to tell if this BigIntegr is probably prime or if its definitely composite. This method checks for prime or composite upon the current BigInteger by which this method is called and returns a Boolean value. It returns true if this BigInteger is probably prime, false if its definitely composite. If certainty is <= 0, true is returned.

Syntax:

Public Boolean isProbalePrime(int certainty)

Parameters: this method accepts a mandatory parameters certainty which is a measure of the uncertainty that is accepatable to the user. This is due to the fact the BigInteger is very very large number and finding exactly if it is prime is very difficult and expensive. Hence it can be said that this method checks for the prime of this BigInteger based on a threshold value $(1-1/2^{certainty})$

Return Value: This method returns a Boolean value stating whether this BigInteger is prime or not. It returns true if this BigInteger is probably prime, false if its definitely composite.

A prime number is a natural number greater than 1 whose only positive divisors are 1 and itself. For example, the first six prime numbers are 2, 3, 5, 7, 11, and 13. Given a large integer, n, use the Java BigInteger class, isProbablePrime method to determine and print whether its prime or not prime

Input Format

A single line containing an integer, n(the number to be checked).

Constraints

n contains at most 100 digits.

Output Format

If n is a prime number, print prime; otherwise, print not prime.

Program:

```
import java.io.*; import
java.math.*; import
java.security.*; import
java.text.*; import
java.util.*;
import java.util.concurrent.*;
import java.util.regex.*; public
class Solution {
    public static void main(String[] args) throws IOExceptio
n {
          try (Scanner scanner = new Scanner(System.in);)
        {
            System.out.println(scanner.nextBigInteger().isPr
obablePrime(100) ? "prime" : "not prime");
        }
    }
}
```

Output:

Sample Input

13

Sample

Output prime

Conclusion:

We proved that the given number is prime.

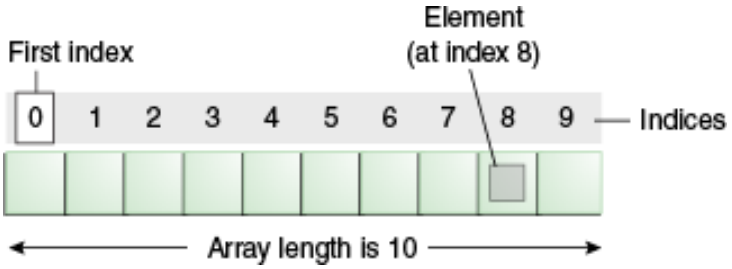
Practical No.: 25
JAVA 1D ARRAY

Aim: Program to print sequential elements of the given array

Tools used: www.hackerrank.com

Theory:

Java array is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array. Array in Java is index-based, the first element of the array is stored at the 0^{th} index, 2nd element is stored on 1st index and so on. In Java, array is an object of a dynamically generated class. Java array inherits the Object class and implements the Serializable as well as Cloneable interfaces. We can store primitive values or objects in an array.



Advantages

- **Code Optimization:** It makes the code optimized, we can retrieve or sort the data efficiently.
- **Random access:** We can get any data located at an index position.

Disadvantages

- **Size Limit:** We can store only the fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in Java which grows automatically.

There are two types of array.

- Single Dimensional Array
- Multidimensional Array

Single Dimensional Array in Java

Syntax :

dataType[] arr; (or) dataType []arr; (or) dataType arr[];

Instantiation of an Array in Java

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

Multidimensional Array in Java

In such case, data is stored in row and column based index (also known as matrix form).

Syntax:

dataType[][] arrayRefVar; (or) dataType [][]arrayRefVar; (or) dataType arrayRefVa r[][]; (or) dataType []arrayRefVar[];

Task

The code in your editor does the following:

1. Reads an integer from stdin and saves it to a variable, n, denoting some number of integers.
2. Reads n integers corresponding to $a_0, a_1, \ldots, a_{n-1}$ from stdin and saves each integer a_i to a variable, val.
3. Attempts to print each element of an array of integers named a.

Input Format

The first line contains a single integer, n, denoting the size of the array. Each line i of the n subsequent lines contains a single integer denoting the value of element a_i.

Output Format

You are not responsible for printing any output to stdout. Locked code in the editor loops through array a and prints each sequential element on a new line.

Program:

```
import java.util.*;
public class Solution {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
int [] a= new int[n];
```

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

Output:

Sample Input

5

10

20

30

40

50

Sample Output

10

20

30

40

50

Conclusion:

We printed the sequential elements of the given array.