Module 1

Assignment Topic: Fundamental

- **Question-Answer**

B1. Features of Java. (or) Java buzzwords

Answer: - java features are as follows:

Simple: - Java was developed by taking the best points from other programming languages, primarily C and C++. Java, therefore, utilizes algorithms and methodologies that are already proven. Error-prone tasks such as pointers and memory management have either been eliminated or handled automatically by the Java environment rather than by the programmer. Since Java is primarily a derivative of C++ which most programmers are conversant with, it implies that Java has a familiar feel rendering it easy to use.

1. Object-oriented: - Java is a fully Object-oriented programming language. & Object-oriented concepts are following.
   - Class
   - Object
   - Inheritance
   - Encapsulation
   - Polymorphism

2. Portable: - The feature Write-once-run-anywhere makes the java language portable provided that the system must have an interpreter for the JVM. Java also has the standard data size irrespective of the operating system or the processor. This feature mathe java as a portable language.

3. Distributed: - The widely used protocols like HTTP and FTP are developed in java. Internet programmers can call functions on these protocols and can get access the files from any remote machine on the internet rather than writing codes on their local system.

4. High performance: - Java uses native code usage, and a lightweight process called threads. In the beginning interpretation of bytecode resulted in the performance being slow but the advanced version of JVM uses the adaptive and just-in-time compilation technique that improves the performance.

5. Multithreaded: - As we all know several features of Java-like Secure, Robust, Portable, dynamic, etc; you will be more delighted to know another feature of Java which is Multithreaded. Java is also a multi-threaded programming language. Multithreading means a single program having different threads executing independently at the same time. Multiple threads execute instructions according to the program code in a process or a program. Multithreading works in a similar way as multiple processes run on one computer. Multithreading programming is a very interesting concept in Java. In multithreaded programs not even, a single thread disturbs the execution of other threads. Threads are obtained from the pool of available ready-to-run threads and they run on the system CPUs. This is how Multithreading works in Java which you will soon come to know in detail in later chapters.

6. Robust: - Java has the strong memory allocation and automatic garbage collection mechanism. It provides the powerful exception handling and type checking mechanism as compare to other programming languages. Compiler checks the program whether there any error and interpreter check any run time error and makes the system secure from crash. All of the above features make the java language robust.

7. Dynamic: - While executing the java program the user can get the required files dynamically from a local drive or from a computer thousands of miles away from the user just by connecting with the Internet.

8. Secure: - Java does not use memory pointers explicitly. All the programs in java are run under an area known as the sandbox. Security manager determines the accessibility options of a class like reading and writing a file to the local disk. Java uses the public key encryption system to allow the java applications to transmit over the internet in the secure encrypted form. The bytecode Verifier checks the classes after loading.

B2. Difference between JDK, JRE, and JVM.

	JDK	JRE	JVM
Full Form	Java Development Kit	Java Runtime Environment	Java Virtual Machine
Definition	It's a software development kit used to develop applications in Java.	It's a software bundle that provides Java class libraries with the necessary components to run Java code.	It's an abstract machine that provides and environment to run and execute Java byte code.
What it Contains	It contains tools for developing, debugging, and monitoring java code.	It contains class libraries and other supporting files that the JVM requires to execute the program.	Software development tools are not included in the JVM.
What it Enables	The JDK enables developers to create Java programs that can be executed and run by the JRE and JVM.	The JRE is the part of the JDK that creates the JVM.	It is the Java platform component that executes source code.
Architecture	Superset	Subset of JDK	Subset of JRE

B3. Java is Platform independent.?

Answer: - Platform independent language means that you can execute the program on any platform (OS) once compiled. Java is platform-independent. Because the Java compiler converts the source code **to** bytecode, which is Intermediate Language. Bytecode can be executed on any platform (OS) using JVM (Java Virtual Machine).

B4. Three flavors of Java.

Answer: - Programs are written in three basic flavors: applets, applications, and servlets/JSP pages.

B5. How many types of memory areas are allocated by JVM?

Answer: - JVM (Java Virtual Machine) areas allocated list are as follows:

1. Class (Method) Area
2. Heap
3. Stack
4. Program Counter Register
5. Native Method Stack

Let's see about them in brief:

1. Class (Method) Area
The class method area is the memory block that stores the class code, variable code(static variable, runtime constant), method code, and the constructor of a Java program. (Here method means the function which is written inside the class). It stores class-level data of every class such as the runtime constant pool, field and method data, the code for methods.

2. Heap
The Heap area is the memory block where objects are created or objects are stored. Heap memory allocates memory for class interfaces and arrays (an array is an object). It is used to allocate memory to objects at run time

3. Stack
Each thread has a private JVM stack, created at the same time as the thread. It is used to store data and partial results which will be needed while returning value for method and performing dynamic linking.

Java Stack stores frames and a new frame is created each time at every invocation of the method. A frame is destroyed when its method invocation completes

4. Program Counter Register:
Each JVM thread that carries out the task of a specific method has a program counter register associated with it. The non-native method has a PC that stores the address of the available JVM instruction whereas, in a native method, the value of the program counter is undefined. PC register is capable of storing the return address or a native pointer on some specific platform.

5. Native method Stacks:
Also called C stacks, native method stacks are not written in Java language. This memory is allocated for each thread when it's created and it can be of a fixed or dynamic nature.

B6. What is the latest version of Java?

Answer: - The latest version of Java is **Java 18 or JDK 18** released on March, 22nd 2022 (follow this article to check Java version on your computer). JDK 18 is a regular update release, and JDK 17 is the latest Long-Term Support (LTS) release of Java SE platform (about 8 years of support from Oracle).

B7. What is Write Once, Run Anywhere (WORA)?

Answer: - Write once, run anywhere (WORA) is a term that refers to a particular program's supposed ability to run on all common OSs (operating systems). The term, sometimes also expressed as write once, run everywhere (WORE), was originally coined by Sun Microsystems in reference to Java.

B8. Is Java a pure/fully object-oriented language?

Answer: - 1) First we will understand what object-oriented language is? When we talk about everything in terms of objects. (i.e., data representation based on objects, methods representation based on objects).

2) Java is not pure object-oriented it means we can represent data with or without objects (i.e., it is possible to represent a few data without objects means there is no need for an object).

3) Java is not a pure object-oriented programming language just because of primitive data types like byte, short, int, char, float, double, long, Boolean, etc. we can work with primitive types if we don't want to work with Objects types.

4) We can represent static data directly there is no need of object instantiation (i.e., it does not mean that you can't create an object still we can create an object if required).

5) We can work with primitives and objects in java (java provides facility to represent data in terms of primitives or objects whatever you want).

6) Object contains variables and methods (i.e., we can call variable or methods with the help of dot operator that is not possible in case of primitives).

7) Primitives are not an object.

B9. What is bytecode?

Answer: - Java bytecode is the instruction set for the Java Virtual Machine. It acts similar to an assembler, an alias representation of a C++ code.

B10. What is Heap space in Java?

Answer: - The Java heap is the area of memory used to store objects instantiated by applications running on the JVM**.** When the JVM is started, heap memory is created and any objects in the heap can be shared between threads as long as the application is running. The size of the heap can vary, so many users restrict the Java heap size to 2-8 GB in order to minimize garbage collection pauses.

Types of Applications where Heap Space Matters

- In-Memory Computing
- NoSQL Databases
- Big Data Applications
- Analytics
- Web Personalization

- eCommerce

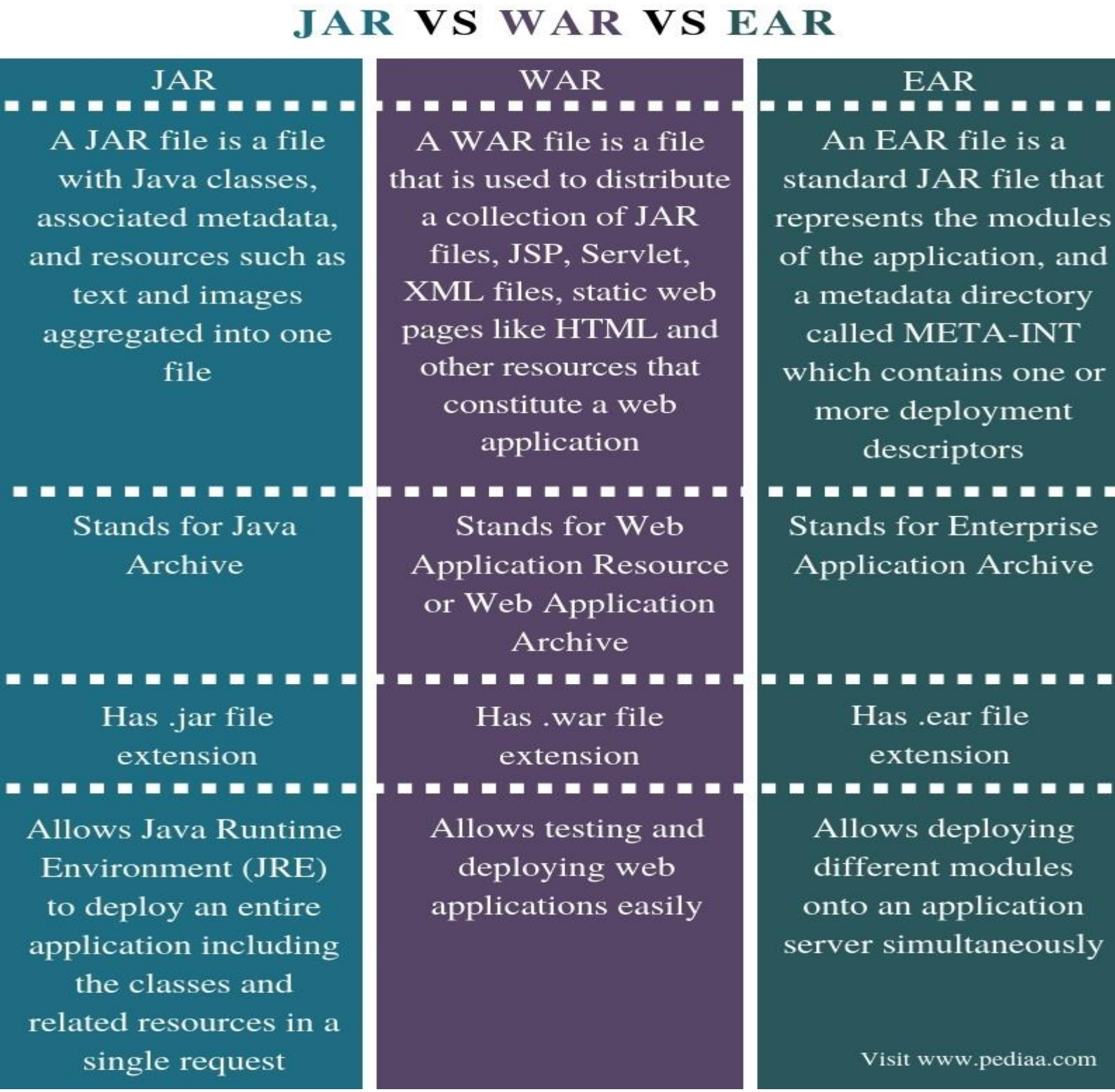
A larger Java memory heap

- Does not require working data to be divided across multiple JVM instances
- Allows more objects to be created
- Takes longer to fill
- Allows the application to run longer between garbage collection (GC) events

A smaller Java memory heap

- Holds fewer objects
- Fills more quickly
- Is garbage collected more frequently (but the pauses are shorter)
- May lead to out-of-memory errors

B11. Difference between EAR, JAR and WAR file in J2EE



B12. Explain memory leak in Java.

Answer: - A memory leak is a situation where unused objects occupy unnecessary space in memory. Unused objects are typically removed by the Java Garbage Collector (GC), but when objects are still being referenced, they are not eligible to be removed.

B13. How did the Garbage collection work in Java?

Answer: - Garbage collection in Java is the process by which Java programs perform automatic memory management. Java programs compile to bytecode that can be run on a Java Virtual Machine, or JVM for short. When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program.

Java garbage collection is an automatic process. Automatic garbage collection is the process of looking at heap memory, identifying which objects are in use and which are not, and deleting the unused objects. An in-use object, or a referenced object, means that some part of your program still maintains a pointer to that object. An unused or unreferenced object is no longer referenced by any part of your program. So the memory used by an unreferenced object can be reclaimed. The programmer does not need to mark objects to be deleted explicitly. The garbage collection implementation lives in the JVM.

- Allocated memory that is no longer needed should be de-allocated.

- In some languages, such as C++, dynamically allocated objects must be manually released by use of a delete operator. Java takes a different approach; it handles de-allocation for you automatically. TOPS Technologies Version_Jan_2019 Page 12

- The technique that accomplishes this is called garbage collection. It works like this: when no references to an object exist, that object is assumed to be no longer needed, and the memory occupied by the object can be reclaimed.

- Garbage collection only occurs sporadically (if at all) during the execution of your program. It will not occur simply because one or more objects exist that are no longer used. Garbage collection has the following characteristics:

- Checks for and frees memory no longer needed.

- Is done automatically.

- Can vary dramatically across JVM implementations.

B14. Does Java garbage collector clean both heap and stack memory?

Answer: - Garbage Collector in Java works only on Heap memory and not on stack memory, because the main principle that stack works on which is (Last in First Out).

B15. Why garbage collection is required in Java?

Answer: - The name "garbage collection" implies that objects no longer needed by the program are "garbage" and can be thrown away. A more accurate and up-to-date metaphor might be "memory recycling." When an object is no longer referenced by the program, the heap space it occupies can be recycled so that the space is made available for subsequent new objects. The garbage collector must somehow determine which objects are no longer referenced by the program and make available the heap space occupied by such unreferenced objects. In the process of freeing unreferenced objects, the garbage collector must run any finalizers of objects being freed.

In addition to freeing unreferenced objects, a garbage collector may also combat heap fragmentation. Heap fragmentation occurs through the course of normal program execution. New objects are allocated, and unreferenced objects are freed such that free portions of heap memory are left in between portions occupied by live objects. Requests to allocate new objects may have to be filled by extending the size of the heap even though there is enough total unused space in the existing heap. This will happen if there is not enough contiguous free heap space available into which the new object will fit. On a virtual memory system, the extra paging (or swapping) required to service an ever-growing heap can degrade the performance of the executing program. On an embedded system with low memory, fragmentation could cause the virtual machine to "run out of memory" unnecessarily.

Garbage collection relieves you from the burden of freeing allocated memory. Knowing when to explicitly free allocated memory can be very tricky. Giving this job to the Java virtual machine has several advantages. First, it can make you more productive. When programming in non-garbage-collected languages you can spend many late hours (or days or weeks) chasing down an elusive memory problem. When programming in Java you can use that time more advantageously by getting ahead of schedule or simply going home to have a life.

A second advantage of garbage collection is that it helps ensure program integrity. Garbage collection is an important part of Java's security strategy. Java programmers are unable to accidentally (or purposely) crash the Java virtual machine by incorrectly freeing memory.

B16. Write A Java Program to Take three numbers from the user and print the greatest number.

Answer: -

```
package BASIC;

import java.util.Scanner;

public class greaternumber {


        public static void main(String args[]) {

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

```
            System.out.print("Enter First Number: ");
            int a = in.nextInt();
            System.out.print("Enter Second Number: ");
            int b = in.nextInt();
            System.out.print("Enter Third Number: ");
            int c = in.nextInt();

            int g = Math.max(a, b);
            g = Math.max(g, c);


            System.out.println("Greatest Number = " + g);
        }
}
```

```
Output: -
Enter First Number: 33
Enter Second Number: 89
Enter Third Number: 44
Greatest Number = 89
```

B17. Write a Java program that takes the user to provide a single character from the alphabet. Print Vowel or Consonant, depending on the user input. If the user input is not a letter (between an and z or A and Z), or is a string of length > 1, print an error message.

```
import java.util.Scanner;
public class Vowel {


  public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);

        System.out.print("Input an alphabet: ");
        String input = in.next().toLowerCase();

        boolean uppercase = input.charAt(0) >= 65 &&
input.charAt(0) <= 90;
        boolean lowercase = input.charAt(0) >= 97 &&
input.charAt(0) <= 122;
        boolean vowels = input.equals("a") ||
input.equals("e") || input.equals("i")
                || input.equals("o") || input.equals("u");

        if (input.length() > 1)
        {
            System.out.println("Error. Not a single
character.");
        }
```

```
        else if (!(uppercase || lowercase))
        {
            System.out.println("Error. Not a letter. Enter
uppercase or lowercase letter.");
        }
        else if (vowels)
        {
            System.out.println("The Input letter is Vowel");
        }
        else
        {
            System.out.println(" The Input letter is
Consonant");
        }
    }
}
```

Output:

```
Input an alphabet: a
The input letter is Vowel

Input an alphabet: s
The input letter is Consonant
```

B18. Write a Java program that takes a year from the user and prints whether that year is a leap year or not.

Answer: -

```
import java.util.Scanner;

public class Leapyear
{
       public static void main(String[] args)
         {
            Scanner in = new Scanner(System.in);

            System.out.print("Input the year: ");
            int year = in.nextInt();

            boolean x = (year % 4) == 0;
            boolean y = (year % 100) != 0;
            boolean z = ((year % 100 == 0) && (year % 400 ==
0));

            if (x && (y || z))
            {
                System.out.println(year + " is a leap year");
```

```
                }
                else
                {
                    System.out.println(year + " is not a leap
year");
}
          }
}
```

Output:

```
Input the year: 2022

2022 is not a leap year
```

B19. Write a program in Java to display the first 10 natural numbers using a while loop.

Answer: -

```
public class Natural_numbers
{
            public static void main(String[] args)
              {
              int i;
               System.out.println ("The first 10 natural
numbers are:\n");
               for (i=1;i<=10;i++)
               {
                    System.out.println (i);
               }
          System.out.println ("\n");
          }
}
```

OUTPUT: -

```
The first 10 natural numbers are:

1
2
3
4
5
6
7
8
9
10
```

B20. Write a program in Java to input 5 numbers from the keyboard and find their sum and average using for loop.

Answer: -

```
import java.util.Scanner;

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

     {
         int i,n=0,s=0;
          double avg;
          {

             System.out.println("Input the 5 numbers : ");

          }
               for (i=0;i<5;i++)
               {
                    Scanner in = new Scanner(System.in);
                    n = in.nextInt();

               s +=n;
          }
          avg=s/5;
          System.out.println("The sum of 5 no is : " +s+"\nThe
Average is : " +avg);

     }
}
```

OUTPUT: -

```
Input the 5 numbers:
3
6
9
4
2
The sum of 5 no is: 24
The Average is: 4.0
```

B21. Write a program in Java to display the pattern like a right-angle triangle with a number.

1

12

123

1234

12345

123456

1234567

12345678

Answer: -

```
import java.util.Scanner;

public class Pattern
{


   public static void main(String[] args)
{
   int i,j,n;
   System.out.print("Input number of rows : ");
   Scanner in = new Scanner(System.in);
              n = in.nextInt();

              for(i=1;i<=n;i++)
              {
               for(j=1;j<=i;j++)
                    System.out.print(j);

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

}
```

Output: -

```
Input numbers pf rows: 8
1

12

123

1234

12345
```

```
123456

1234567

12345678
```

B22. Write a program in Java to make such a pattern like right-angle triangle with a number increased by

1. **The pattern like:**
   1
   2 3
   4 5 6
   7 8 9 10

Answer: -

```
import java.util.Scanner;

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

     {
                int i,j,n,k=1;

                System.out.print("Input number of rows : ");

                Scanner in = new Scanner(System.in);
                    n = in.nextInt();

                for(i=1;i<=n;i++)
                {
                for(j=1;j<=i;j++)
                System.out.print(k++);
                System.out.println("");
                }
           }
}
```

Output: -

```
Input number of rows: 4
1
23
456
```

```
78910
```

B23. Write a program in Java to print Floyd's Triangle.

1

01

101

0101

10101

ANSWER: -

```
public class pattern
{

          public static void main(String[] args) {
               for(int i=1;i<=6;i++) {
                    for(int j=1;j<=i;j++) {
                         if((i+j)%2 == 0) {
                              System.out.print(1);
                         }
                         else {
                              System.out.print(0);
                         }
                    }
                    System.out.println();
               }

          }
}
```

Output: -

```
1
01
101
0101
10101
010101
```

B24. Write a Java program to display Pascal's triangle.

1

1 1

1 2 1
1 3 3 1
1 4 6 4 1

Answer: -

```
import java.util.Scanner;

public class pascle_triangle
{
       public static void main(String[] args)
     {
         int no_row,c=1,blk,i,j;
         System.out.print("Input number of rows: ");
         Scanner in = new Scanner(System.in);
                   no_row = in.nextInt();
         for(i=0;i<no_row;i++)
         {
             for(blk=1;blk<=no_row-i;blk++)
             System.out.print(" ");
             for(j=0;j<=i;j++)
             {
                 if (j==0||i==0)
                     c=1;
                 else
                    c=c*(i-j+1)/j;
                 System.out.print(" "+c);
             }
             System.out.print("\n");
         }
     }
}
```

OUTPUT: -

```
Input number of rows: 5
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
```

B25. Write a Java program that reads a positive integer and counts the number of digits the number. Input an integer number less than ten billion: 125463 Number of digits in the number: 6

Answer: -

```
import java.util.Scanner;

public class TEN_BILLION
{       public static void main(String[] args)
         {
             Scanner in = new Scanner(System.in);
             System.out.print("Input an integer number less
than ten billion: ");

             if (in.hasNextLong())
             {

                 long n = in.nextLong();

                 if (n < 0)
                 {
                     n *= -1;
                 }
                 if (n >= 10000000000L)
                 {
                     System.out.println("Number is greater or
equals 10,000,000,000!");
                 }
                 else
                 {
                     int digits = 1;
                     if (n >= 10 && n < 100)
                     {
                         digits = 2;
                     }
                     else if (n >= 100 && n < 1000)
                     {
                         digits = 3;
                     }
                     else if (n >= 1000 && n < 10000)
                     {
                         digits = 4;
                     }
                     else if (n >= 10000 && n < 100000)
                     {
                         digits = 5;
                     }
                     else if (n >= 100000 && n < 1000000)
                     {
                         digits = 6;
                     }
                     else if (n >= 1000000 && n < 10000000)
                     {
                         digits = 7;
```

```
                    }
                    else if (n >= 10000000 && n < 100000000)
                    {
                        digits = 8;
                    }
                    else if (n >= 100000000 && n <
1000000000)
                    {
                        digits = 9;
                    }
                    else if (n >= 1000000000 && n <
10000000000L)
                    {
                        digits = 10;
                    }
                    System.out.println("Number of digits in
the number: " + digits);
                }
            }
            else
            {
                System.out.println("The number is not an
integer");
            }
        }
}
```

Output: -

```
Input an integer number less than ten billion: 123456
Number of digits in the number: 6
```

B26. Write a Java program to count the letters, spaces, numbers, and other characters of an input string.

Answer: -

```
import java.util.Scanner;

public class Count_values_numbers
{
    public static void main(String[] args) {
            String test = "This is a, demo Programme! 09.10
";
            count(test);

        }
```

```
        public static void count(String x){
            char[] ch = x.toCharArray();
            int letter = 0;
            int space = 0;
            int num = 0;
            int other = 0;
            for(int i = 0; i < x.length(); i++){
                if(Character.isLetter(ch[i])){
                    letter ++ ;
                }
                else if(Character.isDigit(ch[i])){
                    num ++ ;
                }
                else if(Character.isSpaceChar(ch[i])){
                    space ++ ;
                }
                else{
                    other ++;
                }
            }
            System.out.println("The string is : This is a,
demo Programme! 09.10  ");
            System.out.println("letter: " + letter);
            System.out.println("space: " + space);
            System.out.println("number: " + num);
            System.out.println("other: " + other);
                }

}
```

Output: -

```
The string is: This is a demo Programme! 09.10
letter: 20
space: 6
number: 4
other: 3
```

B27. Write a Java program to print the ascii value of a given character.

Answer: -

```
public class Ancii_tablevalues {

        public static void main(String[] String) {
            int chr = 'S';
```

```
                System.out.println("The ASCII value of S
is:"+chr);
            }
}
```

Output: -

```
The ASCII value of S is :83
```

B28. Write a Java program that accepts an integer (n) and computes the value of n+nn+nnn.

Input number: 5 + 55 + 555

Answer: -

```
import java.util.Scanner;

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

        int n;
        char s1, s2, s3;
        Scanner in = new Scanner(System.in);
        System.out.print("Input number: ");
        n = in .nextInt();
        System.out.printf("%d + %d%d  + %d%d%d\n", n, n, n, n,
n, n);
      }
}
```

OUTPUT: -

```
Input number: 5
5 + 55  + 555
```

B29. Write a Java program to display the system time.

Answer: -

```
public class CURRENT_TIME
{
            public static void main(String[] args){
                    System.out.format("\nCurrent Date time:
%tc%n\n", System.currentTimeMillis());
                }
```

```
}
```

Output: -

```
Current Date time: Wed May 04 10:40:48 IST 2022
```

B30. Write a Java program to print numbers between 1 to 100 which are divisible by 3, 5, and by both.

Answer: -

```
public class divide_by_3_5
{
     public static void main(String args[])
     {
          System.out.println("\nDivided by 3: ");
          for (int i=1; i<100; i++) {
               if (i%3==0)
               System.out.print(i +", ");
          }

          System.out.println("\n\nDivided by 5: ");
          for (int i=1; i<100; i++) {
               if (i%5==0) System.out.print(i +", ");

          }

          System.out.println("\n\nDivided by 3 & 5: ");

          for (int i=1; i<100; i++) {
               if (i%3==0 && i%5==0) System.out.print(i +",
");
          }
          System.out.println("\n");
   }
}
```

OUTPUT: -

```
Divided by 3:
3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48,
51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93,
96, 99,

Divided by 5:
5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80,
85, 90, 95,

Divided by 3 & 5:
15, 30, 45, 60, 75, 90,
```

B31. Write a Java program to calculate the sum of two integers and return true if the sum is equal to a third integer.

Input the first number: 5

Input the second number: 10

Input the third number: 15

The result is: true

Answer: -

```
import java.util.*;

public class Equal_to_five
{
      public static void main(String[] args)
         {
             Scanner in = new Scanner(System.in);
             System.out.print("Input the first number : ");
             int x = in.nextInt();
               System.out.print("Input the second number: ");
               int y = in.nextInt();
               System.out.print("Input the third number : ");
             int z = in.nextInt();
             System.out.print("The result is: "+sumoftwo(x, y,
z));
               System.out.print("\n");
         }

         public static boolean sumoftwo(int p, int q, int r)
           {
              return ((p + q) == r || (q + r) == p || (r + p)
== q);
             }
}
```

OUTPUT: -

If the value is equal to 5

```
Input the first number : 15
Input the second number: 10
Input the third number : 5
The result is: true
```

When value is not equal to 5

```
Input the first number : 12
Input the second number: 7
Input the third number : 9
The result is: false
```

B32.Write a Java program that accepts two integer values between 25 to 75 and returns true if there is a common digit in both numbers.

Input the first number: 35

Input the second number: 45

Result: true

Answer: -

```
import java.util.*;

public class number_between_25_75
{
      public static void main(String[] args)
         {
             Scanner in = new Scanner(System.in);
             System.out.print("Input the first number : ");
             int a = in.nextInt();
               System.out.print("Input the second number: ");
               int b = in.nextInt();
               System.out.println("Result: "+common_digit(a,
b));
         }

          public static boolean common_digit(int p, int q)
          {
             if (p<25 || q>75)
                  return false;
             int x = p % 10;
             int y = q % 10;
             p /= 10;
             q /= 10;
             return (p == q || p == y || x == q || x == y);
          }
}
```

OUTPUT: -

```
Input the first number : 35
```

```
Input the second number: 45
Result: true
```

B33.Write a Java program to compute the sum of the first 100 prime numbers.

Answer: -

```
import java.util.*;

public class PRIME_NUMBER
{
      public static void main(String[] args)
      {
         int sum = 1;
           int ctr = 0;
           int n = 0;

               while (ctr < 100) {
                    n++;
                    if (n % 2 != 0) {
                    // check if the number is even
                         if (is_Prime(n)) {
                              sum += n;

                         }
                    }
                         ctr++;
               }
               System.out.println("\nSum of the prime numbers
till 100: "+sum);
           }

          public static boolean is_Prime(int n) {
               for (int i = 3; i * i <= n; i+= 2) {
                    if (n % i == 0) {
                         return false;
                    }
               }
               return true;
          }
}
```

OUTPUT: -

```
Sum of the prime numbers till 100: 1060
```

B34. Write a Java program that accepts two double variables and test if both are strictly between 0 and 1 and false otherwise.

Input first number: 5

Input second number: 1

False

Answer: -

```
import java.util.Scanner;

public class Double_values
{

          public static void main(String[] args)
          {
                Scanner in = new Scanner(System.in);
                System.out.print("Input first number: ");
              double n1 = in.nextDouble();
              System.out.print("Input second number: ");
              double n2 = in.nextDouble();
                System.out.println(n1 > 0 && n1 < 1 && n2 > 0
&& n2 < 1);
          }
}
```

OUTPUT: -

```
Input first number: 0
Input second number: 1
false
```

B35.Write a Java program to break an integer into a sequence of individual digits. TestData Input six non-negative digits: 123456

ExpectedOutput: 1 2 3 4 5 6

Answer: -

```
import java.util.Scanner;

public class break_integer
```

```
{
        public static void main(String[] args)
        {
            Scanner in = new Scanner(System.in);
            System.out.print("Input six non-negative digits:
");
            int input = in.nextInt();
            int n1 = input / 100000 % 10;
            int n2 = input / 10000 % 10;
            int n3 = input / 1000 % 10;
            int n4 = input / 100 % 10;
            int n5 = input / 10 % 10;
            int n6 = input % 10;
          System.out.println(n1 + " " + n2 + " " + n3 + " " +
n4 + " " + n5 + " " + n6);

        }
}
```

OUTPUT: -

```
Input six non-negative digits: 12345678
3 4 5 6 7 8
```