



FULL STACK DEVELOPMENT: WORKSHEET A

Table of Contents

Introduction.....	2
Ques 1. Write a java program Add two Numbers.....	3
Ques 2. Write a java program Check Whether a Number is Even or Odd.	5
Ques 3. Write a java program Check if a given number is palindrome or not.	7
Ques 4. Write a java program to find the sum of n natural numbers.....	10
Ques 5. Write a java program to Check Prime Number or not.....	12
Conclusion	15

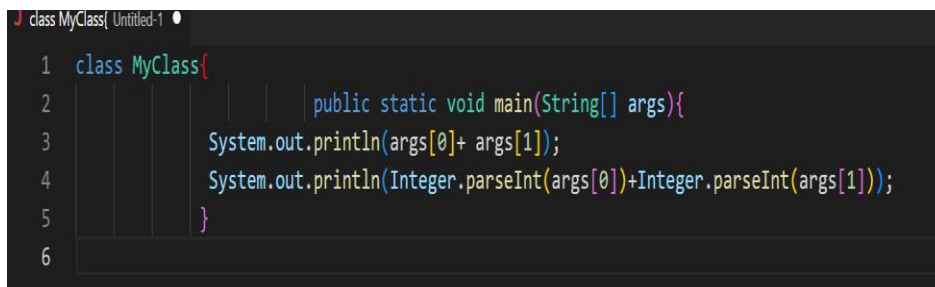
Introduction

Numerous fundamental ideas in the field of Java programming are shown through clear-cut and informative code snippets. These include summing natural numbers, determining palindromes, comparing even and odd numbers, adding two numbers, and validating prime numbers. Each programme provides a thorough introduction to essential Java programming concepts by demonstrating fundamental coding principles, processing user input, and logical constructions.

Ques 1. Write a java program Add two Numbers.

Ans.

```
class MyClass{  
  
    public static void main(String[] args){  
  
        System.out.println(args[0]+ args[1]);  
  
        System.out.println(Integer.parseInt(args[0])+Integer.parseInt(args[1]));  
  
    }  
}
```



The Java code that is supplied constructs a class with a main function called MyClass. Let's break down the code piece by piece to understand its functionality:

- **class MyClass:** A class named MyClass is defined in this line. A class in Java acts as a blueprint for creating objects and encapsulating behaviour.
- **public static void main(String[] args):** This is the class's main function and the place where the Java programme starts. When the programme is run it is the method that is executed. Any command-line parameters given to the programme are included in the array of strings known as the String [] args parameter.
- **System.out.println(...):** This command prints an expression's result to standard output, which is often the console.
- **parseInt Integer (args [0]):** This section of code uses the Integer.parseInt () function to transform the first command-line parameter, which is a string and is located at index 0 of the args array, into an integer. Similar to that, the second command-line argument (at index 1) is taken and converted to an integer by Integer.parseInt (args [1]).
- **+:** This is the addition operator. The two numbers that were acquired from the command-line inputs are added.
- **args [0] + args [1]:** This section of the code concatenates strings rather than adding them. The two strings that represent the command-line parameters are concatenated, which is not what we want in this situation.

- **System.out.println (Integer.parseInt (args [0]) + Integer.parseInt (args [1])):** This line combines the two numbers derived from the command-line parameters properly and publishes the result as an integer to the console.

Ques 2. Write a java program Check Whether a Number is Even or Odd.

Ans. import java.util.Scanner;

```
class EvenOddChecker {  
  
    public static void main(String[] args) {  
  
        Scanner scanner = new Scanner(System.in);  
  
        System.out.print("Enter a number: ");  
  
        int number = scanner.nextInt();  
  
        if (number % 2 == 0) {  
  
            System.out.println(number + " is even.");  
  
        } else {  
  
            System.out.println(number + " is odd.");  
  
        }  
  
        scanner.close();  
  
    }  
  
}
```

A screenshot of a Java IDE window titled 'EvenOdd.java'. The code is displayed with syntax highlighting: imports in blue, class and method declarations in black, and logic in black. Line numbers 1 through 19 are visible on the left. The code matches the one provided in the previous block.

```
1  import java.util.Scanner;  
2  
3  class EvenOddChecker {  
4      public static void main(String[] args) {  
5          Scanner scanner = new Scanner(System.in);  
6  
7          System.out.print(s:"Enter a number: ");  
8          int number = scanner.nextInt();  
9  
10         if (number % 2 == 0) {  
11             System.out.println(number + " is even.");  
12         } else {  
13             System.out.println(number + " is odd.");  
14         }  
15  
16         scanner.close();  
17     }  
18 }  
19
```

The Java programme in this code determines if a given integer is even or odd. Let's walk through the code piece by piece and describe each component:

- **import java.util.Scanner:** This line imports the Scanner class, which is used to read input from the user, from the java.util package.
- **class EvenOddChecker:** This specifies the EvenOddChecker class. Each class in a Java programme has methods that specify the program's behaviour, and a Java programme normally consists of one or more classes.
- **public static void main(String[] args):** This is the class's main function and the place where the Java programme starts. When the programme is run, it is the method that is executed.
- **Scanner scanner = new Scanner(System.in):** This statement prompts the user to enter a number by printing the message "Enter a number:".
- **int number = scanner.nextInt():** This line uses the nextInt() function of the Scanner class to read an integer input from the user and stores it in the number variable.
- **If (number % 2 == 0):** The if statement is introduced by this line. It determines if the criteria for an even number—that the remainder of the number divided by two equals zero—is met.
- **System.out.println(number + " is even."):** If the condition in the if statement is true, this line is executed. The input number is even, according to the message that is printed.
- **else:** This line starts the else block, which includes code that will run if the if statement's condition is false.
- **System.out.println(number + "is odd."):** This line is performed if the if statement's condition is false. The oddness of the input number is indicated by a message that is printed.
- **Scanner.close():** This line closes the Scanner object and releases all of its resources.

This programme will ask you for a number when you run it. When a number is entered, the programme checks to see if it's even or odd, and then it displays the relevant message.

Ques 3. Write a java program Check if a given number is palindrome or not.

Ans. import java.util.Scanner;

```
class PalindromeCheck{

    public static void main(String args[])

    {

        Scanner in = new Scanner(System.in);

        System.out.print("Input a number: ");

        int n = in.nextInt();

        int sum = 0, r;

        int temp = n;

        while(n>0)

        {

            r = n % 10;

            sum = (sum*10)+r;

            n = n/10;

        }

        if(temp==sum)

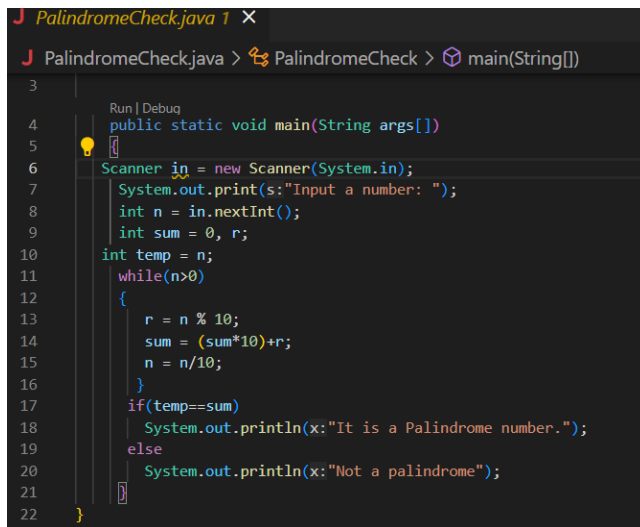
            System.out.println("It is a Palindrome number.");

        else

            System.out.println("Not a palindrome");

        }

}
```

```
3
4 public static void main(String args[])
5
6 Scanner in = new Scanner(System.in);
7 System.out.print(s:"Input a number: ");
8 int n = in.nextInt();
9 int sum = 0, r;
10 int temp = n;
11 while(n>0)
12 {
13     r = n % 10;
14     sum = (sum*10)+r;
15     n = n/10;
16 }
17 if(temp==sum)
18     System.out.println(x:"It is a Palindrome number.");
19 else
20     System.out.println(x:"Not a palindrome");
21 }
22 }
```

The revealed a programme that determines whether or not a given number is a palindrome is written in Java code. Let's analyse the code step by step to see how it operates:

- **Import java.util.scanner:** This line imports the Java.util package's Scanner class, allowing the programme to read user input.
- **class PalindromeCheck:** This defines the main method and additional code for the class PalindromeCheck.
- **public static void main(String args[]):** This is the main method where the program's execution starts.
- **Scanner in = new Scanner(System.in):** This statement creates an instance of the Scanner class with the name in to read user input.
- **System.out.println("Input a number: "):** This line shows the prompt asking a numeric entry from the user.
- **int n = in.nextInt():** This line receives a user-provided integer input and stores it in the variable n.
- **int sum = 0,r:** This sets up two integer variables, sum and r, to record the reversal of the number and the remainder of the division, respectively.
- **int temp = n:** This creates a copy of the value n into the temp variable.
- The digits of the number are reversed using the 'while' loop described below:

```
while(n > 0) {
    r = n % 10;
    sum = (sum * 10) + r;
```

```
n = n / 10;  
}
```

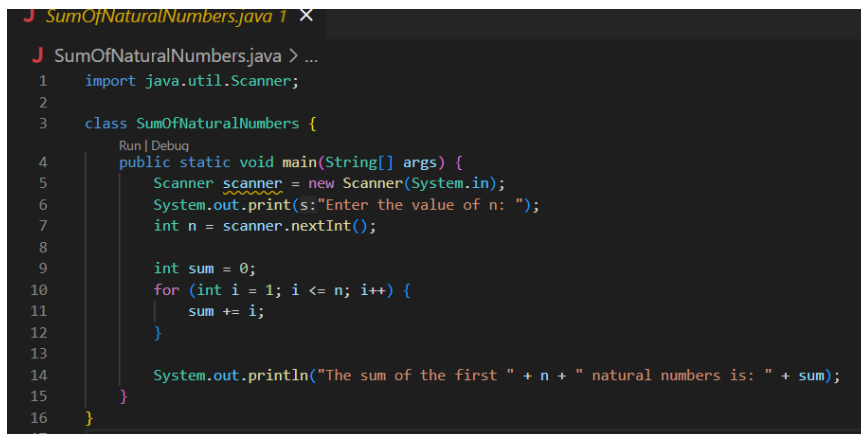
- Following the loop, the reversal is stored in the sum of the variable.
- The if clause determines whether the original number temp and the reversal number sum are same. The number is a palindrome if they are equal; otherwise, they are not.
- Finally, based on whether or not the number is a palindrome, the programme produces the appropriate message.

In order to determine if a number is a palindrome or not, the programme reverses the digits of an input integer and comparing the result to the original number.

Ques 4. Write a java program to find the sum of n natural numbers.

Ans. `import java.util.Scanner;`

```
class SumOfNaturalNumbers {  
  
    public static void main(String[] args) {  
  
        Scanner scanner = new Scanner(System.in);  
  
        System.out.print("Enter the value of n: ");  
  
        int n = scanner.nextInt();  
  
        int sum = 0;  
  
        for (int i = 1; i <= n; i++) {  
  
            sum += i;  
  
        }  
  
        System.out.println("The sum of the first " + n + " natural numbers is: " + sum);  
  
    }  
  
}
```

A screenshot of a Java IDE window titled 'SumOfNaturalNumbers.java'. The code is displayed with syntax highlighting: imports in blue, class and method declarations in black, and variable/constant declarations in green. The code matches the one shown in the previous block. The IDE interface includes a 'Run | Debug' button and line numbers on the left margin.

- **import java.util.Scanner:** This line imports the Java.util.Scanner class, which is used to receive user input.

- **class SumOfNaturalNumber:** This line introduces a new class called SumOfNaturalNumbers. The primary logic of the programme will be included in this class, which is how Java organises programmes into classes.

- **public static void main(String [] args):** This is where the program's execution begins. The main method is where the program's execution starts.
- **Scanner scanner = new Scanner(System.in):** Scanner is the name of a newly created Scanner object. It reads user input and processes it.
- **System.out.print("Enter the value of n:"):** This line prompts the user for input by printing the message "Enter the value of n:" to the console. Then, an integer value is obtained from the user and placed in the variable n using the nextInt() function of the scanner class.
- **int sum = 0:** The initial value of the integer variable sum is 0. To process through all integers from 1 to n, a for loop is needed. The sum variable is increased by the value of i throughout each iteration. The sum of the first "n" natural numbers is calculated using this loop.
- **System.out.println("The sum of the first n natural numbers is:" + sum):** Prints the computed sum to the console along with a message explaining what it stands for. The + operator is used to concatenate the message's various components.

In summary, this Java programme asks for the user to provide a positive integer "n" computes the sum of the first "n" natural integers using a loop and then outputs the result along with a helpful message.

Ques 5. Write a java program to Check Prime Number or not.

Ans. import java.util.Scanner;

```
class primeNumberCheck{

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a number: ");

        int num = scanner.nextInt();

        if (isPrime(num)) {

            System.out.println(num + " is a prime number.");

        } else {

            System.out.println(num + " is not a prime number.");

        }

    }

    public static boolean isPrime(int num) {

        if (num <= 1) {

            return false;

        }

        if (num <= 3) {

            return true;

        }

        if (num % 2 == 0 || num % 3 == 0) {

            return false;

        }

        for (int i = 5; i * i <= num; i += 6) {

            if (num % i == 0 || num % (i + 2) == 0) {
```

```
        return false;
    }
}

    return true;
}
}
```



```
J primeNumberCheck.java 1 x
J primeNumberCheck.java > ...
1  import java.util.Scanner;
2
3  public class primeNumberCheck{
4      Run|Debug
5      public static void main(String[] args) {
6          Scanner scanner = new Scanner(System.in);
7          System.out.print(s:"Enter a number: ");
8          int num = scanner.nextInt();
9
10         if (isPrime(num)) {
11             System.out.println(num + " is a prime number.");
12         } else {
13             System.out.println(num + " is not a prime number.");
14         }
15     }
16
17     public static boolean isPrime(int num) {
18         if (num <= 1) {
19             return false;
20         }
21
22         if (num <= 3) {
23             return true;
24         }
25
26         if (num % 2 == 0 || num % 3 == 0) {
27             return false;
28         }
29
30         for (int i = 5; i * i <= num; i += 6) {
31             if (num % i == 0 || num % (i + 2) == 0) {
32                 return false;
33             }
34         }
35
36         return true;
37     }
38 }
```

- **import java.util.Scanner:** This line imports the Java.util.Scanner class, which is used to receive user input.
- **primeNumberCheck Class:** This class has the functionality to determine if a given number is a prime number.
- **main Method:** The program's entry point is the main method. It performs the following:
 - to read user input from the console, creates the Scanner object with the name scanner.
 - requests a number from the user.
 - scans the entered number to read it and saves it in the variable num using nextInt().
 - determines whether num is a prime number by calling the isPrime function.
 - depending on whether num is a prime number, prints the outcome.

- **is Prime Method:** The return value of this method, which accepts an integer num as an input, is a boolean value (true if the number is a prime, false otherwise). To determine if a number is prime, the following procedures are taken:
 - Given that prime numbers are more than 1, the function returns false if num is less than or equal to 1.
 - As they are prime numbers, if num is 2 or 3, it returns true.
 - Since prime numbers have only two divisors: 1 and themselves, if num is divisible by 2 or 3, it returns false.
 - To optimise the algorithm, the loop from $i = 5$ to $i * i = \text{num}$ iterates through odd values (skipping even ones). It determines if num is divisible by i or $i + 2$ within the loop. The procedure returns false if each of these circumstances holds true, indicating that num is not prime.
 - The method returns true, indicating that num is prime, if none of the following requirements are satisfied during the loop.
- Overall, the programme uses an algorithm that is optimised to eliminate unnecessary divisibility checks for some integers while determining if a given number is prime or not.

Conclusion

These Java programmes ultimately provide information on essential coding techniques and approaches to problem-solving. The examples provided offer a strong basis for developing programmers, including everything from arithmetic operations and conditional expressions to user input management. The ability to comprehend and use these ideas will enable developers to take on increasingly challenging tasks with ease and proficiency.