

Batch:B4**Roll No.:16010122221****Experiment / assignment / tutorial No.01****Grade: AA / AB / BB / BC / CC / CD /DD****Signature of the Staff In-charge with date****TITLE : Perfect Number**

AIM: Define a class Perfect which accepts the range of numbers from the user. Create a static function check_per , which checks if the number is a perfect number or not and sends the result back to the main function which counts and displays the perfect numbers within that range.

Variations :

Implementation of Program with One class

Accessibility with static and non-static methods within class and outside class.

Expected OUTCOME of Experiment:

CO2: Explore arrays, vectors, classes and objects in C++ and Java

Books/ Journals/ Websites referred:

1. E. Balagurusamy , “Programming with Java” McGraw-Hill.
2. Sachin Malhotra, Saurabh Choudhary, “Programming in Java”, Oxford Publications.

Pre Lab/ Prior Concepts:

The Scanner class is a class in java.util, which allows the user to read values of various types. There are far more methods in class Scanner than you will need in this course. We only cover a small useful subset, ones that allow us to read in numeric values from either

the keyboard or file without having to convert them from strings and determine if there are more values to be read.

```
Scanner in = new Scanner(System.in); // System.in is an InputStream
```

Numeric and String Methods

Method	Returns
int nextInt()	Returns the next token as an int. If the next token is not an integer, <code>InputMismatchException</code> is thrown.
long nextLong()	Returns the next token as a long. If the next token is not an integer, <code>InputMismatchException</code> is thrown.
float nextFloat()	Returns the next token as a float. If the next token is not a float or is out of range, <code>InputMismatchException</code> is thrown.
double nextDouble()	Returns the next token as a long. If the next token is not a float or is out of range, <code>InputMismatchException</code> is thrown.
String next()	Finds and returns the next complete token from this scanner and returns it as a string; a token is usually ended by whitespace such as a blank or line break. If not token exists, <code>NoSuchElementException</code> is thrown.
String nextLine()	Returns the rest of the current line, excluding any line separator at the end.
void close()	Closes the scanner.

The Scanner looks for tokens in the input. A token is a series of characters that ends with what Java calls whitespace. A whitespace character can be a blank, a tab character, a carriage return. Thus, if we read a line that has a series of numbers separated by blanks, the scanner will take each number as a separate token.

The numeric values may all be on one line with blanks between each value or may be on separate lines. Whitespace characters (blanks or carriage returns) act as separators. The next method returns the next input value as a string, regardless of what is keyed. For example, given the following code segment and data

- `int number = in.nextInt();`
- `float real = in.nextFloat();`
- `long number2 = in.nextLong();`
- `double real2 = in.nextDouble();`
- `String string = in.next();`

Class Diagram:

<u>CLASS</u>	<u>METHOD</u>
Perfect	<code>int calc (int n)</code>
Main	<code>public static void main (string arg sc)</code>

Perfect		Main
-n : int		-a, x : int
-sum : int	←	-s : perfect
-i : int		
+ calc (n:int):int		+ calc(a): int

Algorithm:

1. Define a class named "Perfect".
2. Inside the "Perfect" class:

2.1 Define instance variables "start" and "end" to store the range of numbers.

2.2 Define a constructor that accepts "start" and "end" as parameters and initializes the instance variables.

2.3 Define a static method named "check_per" that accepts a number as a parameter and checks if it is a perfect number:

2.3.1 Initialize a variable "sumDivisors" to 0.

2.3.2 Loop from 1 to (number / 2):

2.3.2.1 If the current number divides "number" evenly, add it to "sumDivisors".

2.3.3 If "sumDivisors" is equal to "number", return true (perfect number), otherwise, return false.

2.4 Define a non-static method named "countAndDisplayPerfectNumbers" to count and display perfect numbers within the range:

2.4.1 Initialize a variable "perfectCount" to 0.

2.4.2 Loop through the range of numbers:

2.4.2.1 Call the static method "check_per" with the current number to check if it's a perfect number.

2.4.2.2 If the number is perfect, increment "perfectCount" and display it.

2.4.3 Display the total count of perfect numbers.

3. In the main part of the code:

3.1 Create a scanner object to get user input.

3.2 Prompt the user to enter the start and end of the range.

3.3 Read the input for "startRange" and "endRange".

3.4 Create an instance of the "Perfect" class with the provided range.

3.5 Call the "countAndDisplayPerfectNumbers" method on the instance to display perfect numbers.

3.6 Close the scanner.

Implementation details:

variation-1

```
import java.util.Scanner;
```

```
class Perfect {  
  
    public static boolean check_per(int number) {  
  
        int sumDivisors = 0;  
  
        for (int i = 1; i <= number / 2; i++) {  
  
            if (number % i == 0) {  
  
                sumDivisors += i;  
  
            }  
  
        }  
  
    }  
  
}
```

```
    }

    return sumDivisors == number;
}

public void countAndDisplayPerfectNumbers(int start, int end) {

    int perfectCount = 0;

    for (int num = start; num <= end; num++) {

        if (check_per(num)) {

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

            perfectCount++;

        }

    }

    System.out.println("Total perfect numbers in the range: " + perfectCount);

}

public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter start of range: ");

    int startRange = scanner.nextInt();

    System.out.print("Enter end of range: ");
```

```
int endRange = scanner.nextInt();

Perfect perfect = new Perfect();

perfect.countAndDisplayPerfectNumbers(startRange, endRange);

scanner.close();

}

}
```

variation-2

```
import java.util.ArrayList;

import java.util.Scanner;

class Perfect {

    private int start;

    private int end;

    public Perfect(int start, int end) {
```

```
this.start = start;  
  
this.end = end;  
  
}
```

```
public static boolean checkPer(int number) {  
  
    int sum = 0;  
  
    for (int i = 1; i <= number / 2; i++) {  
  
        if (number % i == 0) {  
  
            sum += i;  
  
        }  
  
    }  
  
    return sum == number;  
  
}
```

```
public ArrayList<Integer> findPerfectNumbers() {  
  
    ArrayList<Integer> perfectNumbers = new ArrayList<>();  
  
    for (int num = start; num <= end; num++) {  
  
        if (checkPer(num)) {  
  
            perfectNumbers.add(num);  
  
        }  
  
    }  
  
    return perfectNumbers;  
  
}
```



```
}
```

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

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

```
    System.out.print("Enter the start of the range: ");
```

```
    int startRange = scanner.nextInt();
```

```
    System.out.print("Enter the end of the range: ");
```

```
    int endRange = scanner.nextInt();
```

```
    Perfect perfectObj = new Perfect(startRange, endRange);
```

```
    ArrayList<Integer> perfectNumbers = perfectObj.findPerfectNumbers();
```

```
    int count = perfectNumbers.size();
```

```
    System.out.println("Number of perfect numbers within the range: " + count);
```

```
    System.out.println("Perfect numbers within the range:");
```

```
    for (int num : perfectNumbers) {
```

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

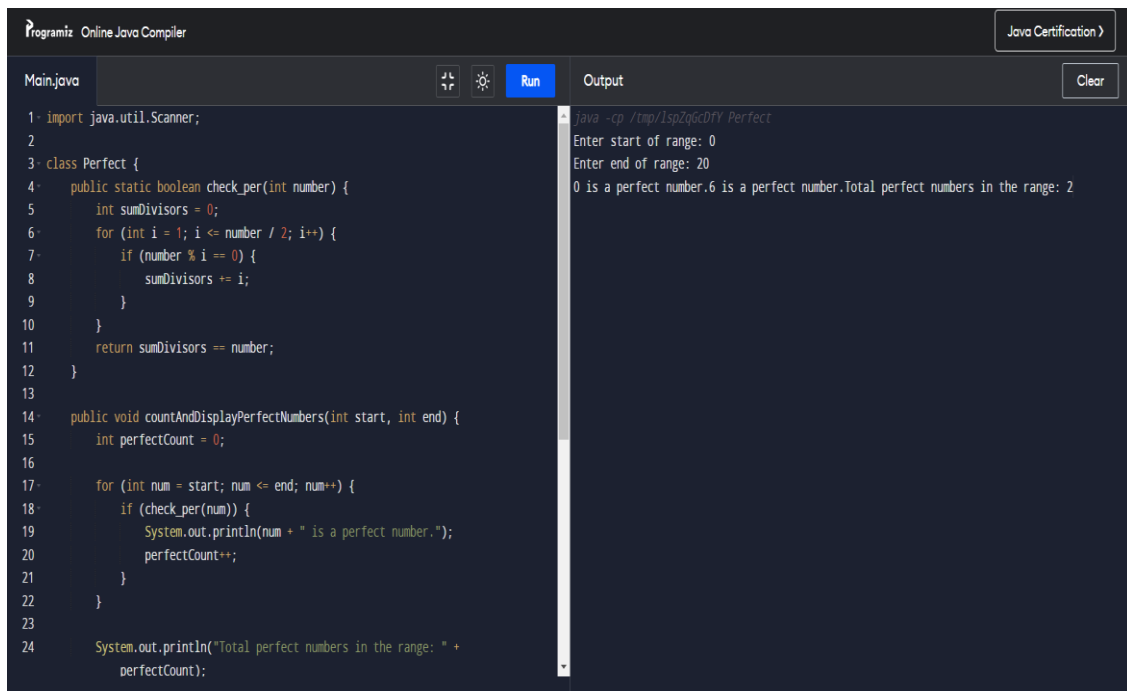
```
    }
```

```
}
```

```
}
```

Output:

VARIATION-1



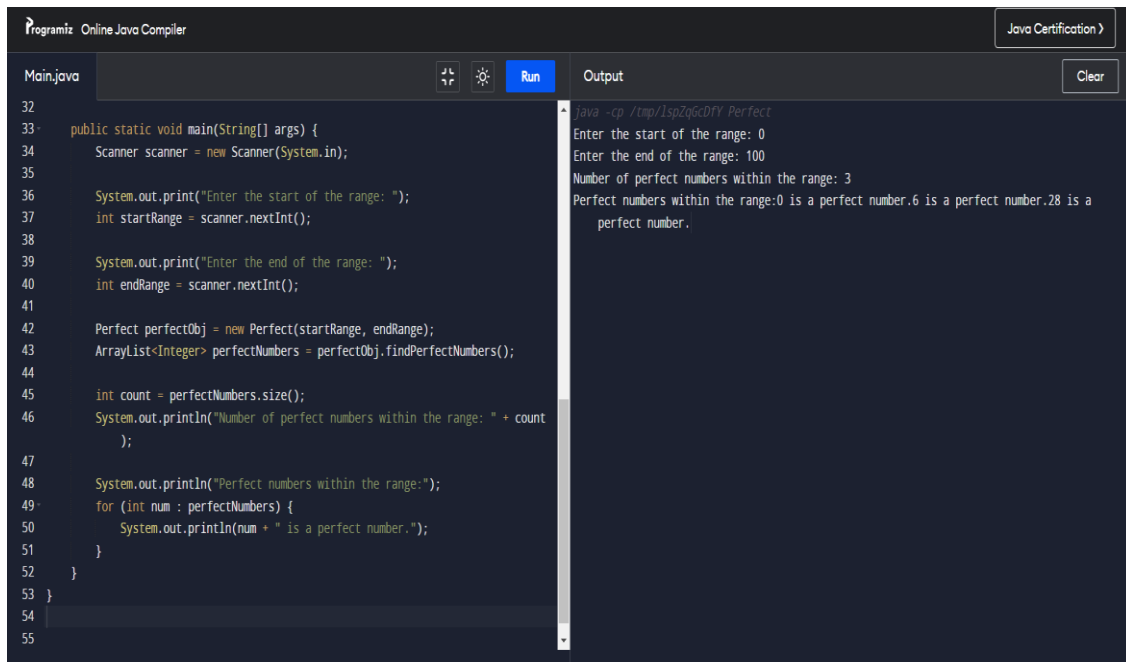
The screenshot shows a web-based Java IDE with a dark theme. The editor contains a Java program named 'Main.java' that defines a 'Perfect' class. The class has two methods: 'check_per' which checks if a number is perfect by summing its divisors, and 'countAndDisplayPerfectNumbers' which iterates through a range of numbers and prints perfect ones. The output window shows the execution results for the range 0 to 20, identifying 0 and 6 as perfect numbers.

```
Main.java  Run  Output  Clear
```

```
1 import java.util.Scanner;
2
3 class Perfect {
4     public static boolean check_per(int number) {
5         int sumDivisors = 0;
6         for (int i = 1; i <= number / 2; i++) {
7             if (number % i == 0) {
8                 sumDivisors += i;
9             }
10        }
11        return sumDivisors == number;
12    }
13
14    public void countAndDisplayPerfectNumbers(int start, int end) {
15        int perfectCount = 0;
16
17        for (int num = start; num <= end; num++) {
18            if (check_per(num)) {
19                System.out.println(num + " is a perfect number.");
20                perfectCount++;
21            }
22        }
23
24        System.out.println("Total perfect numbers in the range: " +
25            perfectCount);
26    }
27 }
```

```
java -cp /tmp/lspZqGcDTY Perfect
Enter start of range: 0
Enter end of range: 20
0 is a perfect number.6 is a perfect number.Total perfect numbers in the range: 2
```

VARIATION-2



The screenshot shows a web-based Java IDE. The code in the editor is as follows:

```
32
33 public static void main(String[] args) {
34     Scanner scanner = new Scanner(System.in);
35
36     System.out.print("Enter the start of the range: ");
37     int startRange = scanner.nextInt();
38
39     System.out.print("Enter the end of the range: ");
40     int endRange = scanner.nextInt();
41
42     Perfect perfectObj = new Perfect(startRange, endRange);
43     ArrayList<Integer> perfectNumbers = perfectObj.findPerfectNumbers();
44
45     int count = perfectNumbers.size();
46     System.out.println("Number of perfect numbers within the range: " + count
47         );
48
49     System.out.println("Perfect numbers within the range:");
50     for (int num : perfectNumbers) {
51         System.out.println(num + " is a perfect number.");
52     }
53 }
54
55
```

The output window shows the following text:

```
java -cp /tmp/1spZqGcDFY Perfect
Enter the start of the range: 0
Enter the end of the range: 100
Number of perfect numbers within the range: 3
Perfect numbers within the range:0 is a perfect number.6 is a perfect number.28 is a
perfect number.
```

Conclusion:

In this experiment, we learnt how to implement a program with one class and multiple classes. Along with accessibility with static and non-static methods within class and outside class.

Date: _____

Signature of faculty in-charge

Post Lab Descriptive Questions:

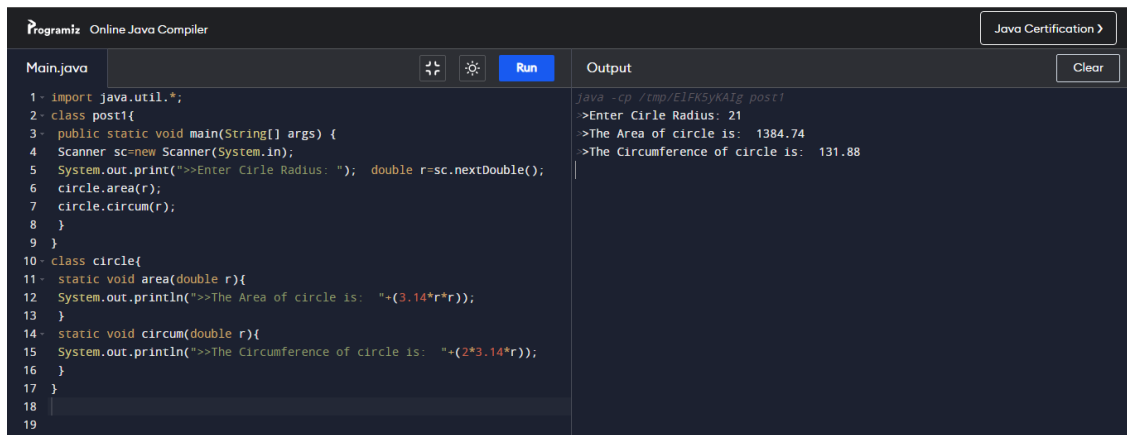
Q.1 Write a program to find the area and circumference of a circle using two classes.

CODE-:

```
import java.util.*;

class post1 {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print(">>Enter Circle Radius: "); double r=sc.nextDouble();
        circle.area(r);
        circle.circum(r);
    }
}

class circle{
    static void area(double r){
        System.out.println(">>The Area of circle is: "+(3.14*r*r));
    }
    static void circum(double r){
        System.out.println(">>The Circumference of circle is: "+(2*3.14*r));
    }
}
```



The screenshot shows a web-based Java IDE. The editor contains a Java program named 'Main.java' that defines a 'post1' class with a 'main' method. It uses a 'Scanner' to take input for a circle's radius (21) and then calls static methods 'area' and 'circum' from a 'circle' class. The 'circle' class has static methods 'area' and 'circum' that calculate the area and circumference using the formulae $A = \pi r^2$ and $C = 2\pi r$ respectively, with π approximated as 3.14. The output window shows the program's execution: it prompts for the radius, then prints 'The Area of circle is: 1384.74' and 'The Circumference of circle is: 131.88'.

```
1- import java.util.*;
2- class post1{
3- public static void main(String[] args) {
4- Scanner sc=new Scanner(System.in);
5- System.out.print(">>Enter Circle Radius: "); double r=sc.nextDouble();
6- circle.area(r);
7- circle.circum(r);
8- }
9- }
10- class circle{
11- static void area(double r){
12- System.out.println(">>The Area of circle is: "+(3.14*r*r));
13- }
14- static void circum(double r){
15- System.out.println(">>The Circumference of circle is: "+(2*3.14*r));
16- }
17- }
18-
19-
```

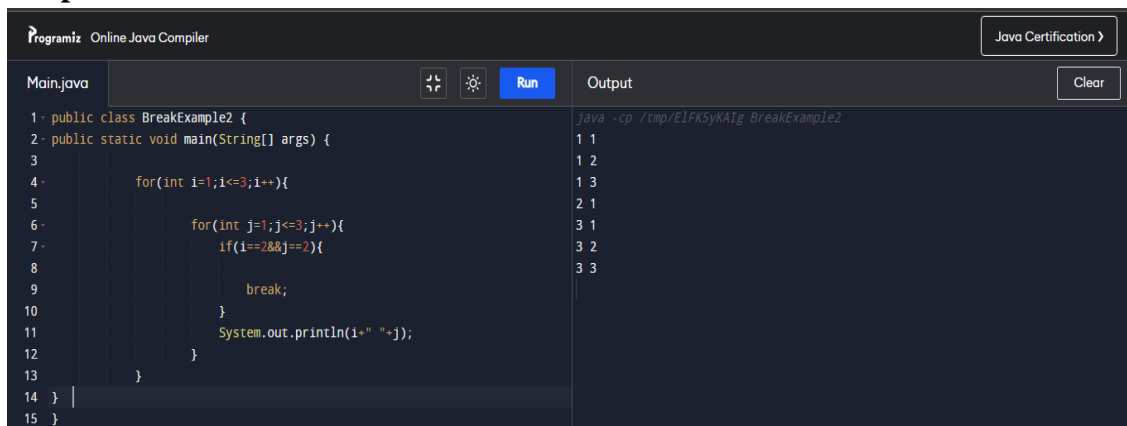
Output

```
java -cp /tmp/ElFKSyKAig post1
>Enter Circle Radius: 21
>The Area of circle is: 1384.74
>The Circumference of circle is: 131.88
```

Q.2 Write the output of following program

```
1. public class BreakExample2 {
2. public static void main(String[] args) {
3. //outer loop
4. for(int i=1;i<=3;i++){
5. //inner loop
6. for(int j=1;j<=3;j++){
7. if(i==2&&j==2){
8. //using break statement inside the inner loop
9. break;
10. }
11. System.out.println(i+" "+j);
12. }
13. }
14. }
15. }
```

Output:



The screenshot shows a web-based Java IDE. The editor contains a Java program named 'Main.java' that defines a 'BreakExample2' class with a 'main' method. It uses nested 'for' loops: an outer loop for 'i' from 1 to 3, and an inner loop for 'j' from 1 to 3. Inside the inner loop, there is an 'if' condition 'if(i==2&&j==2)' which triggers a 'break' statement, exiting the inner loop. The program prints the values of 'i' and 'j' for each iteration. The output window shows the program's execution, displaying the pairs (1,1), (1,2), (1,3), (2,1), (2,2), (2,3), and (3,3). The pair (2,2) is the last one printed because the break statement exits the inner loop before it can print (2,3).

```
1- public class BreakExample2 {
2- public static void main(String[] args) {
3-
4- for(int i=1;i<=3;i++){
5-
6- for(int j=1;j<=3;j++){
7- if(i==2&&j==2){
8- break;
9- }
10- }
11- System.out.println(i+" "+j);
12- }
13- }
14- }
15- }
```

Output

```
java -cp /tmp/ElFKSyKAig BreakExample2
1 1
1 2
1 3
2 1
3 1
3 2
3 3
```

Q.3 Why is Java known as a platform independent language?

Java is based on the Write-Once-Run-Anywhere concept that makes it Platform independent. It is platform independent because the program written in it is not directly converted into machine code but instead is converted into byte code by Java compiler, this byte code is then converted into machine readable code by Java Virtual Machine (JVM). JDK including JVM must be installed in a platform.

Q.4 Write a recursive static method for calculation of gcd of a number.

```
class post2 {  
    public static void main(String[] args) { int a = 15, b = 150;  
        System.out.println("G.C.D is: "+gcd(a,b)); }  
    public static int gcd(int a, int b) {  
        if (b != 0)  
            return gcd(b, a % b);  
        else  
            return a;  
    }  
}
```



The screenshot shows the 'Programiz Online Java Compiler' interface. On the left, a code editor displays a Java program named 'Main.java'. The code defines a class 'post2' with a 'main' method that takes two integers, 'a' and 'b', and prints their Greatest Common Divisor (G.C.D). A 'gcd' method is also defined, which uses a recursive algorithm to find the G.C.D. The 'main' method calls 'gcd(a, b)' and prints the result. On the right, the 'Output' pane shows the command 'java -cp ./tmp/ElFKSyKAig post2' and the output 'G.C.D is: 15'. A 'Run' button is visible between the code editor and the output pane.

```
1 class post2 {  
2     public static void main(String[] args) { int a = 15, b = 150;  
3         System.out.println("G.C.D is: "+gcd(a,b)); }  
4     public static int gcd(int a, int b) {  
5         if (b != 0)  
6             return gcd(b, a % b);  
7         else  
8             return a;  
9     }  
10 }  
11
```

Output
Clear

java -cp ./tmp/ElFKSyKAig post2
G.C.D is: 15

