योगः कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS

Grade A++ Accredited Institution by NAAC

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;
Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE
An ISO 9001:2015 Certified Institution

SCHOOL OF ENGINEERING & TECHNOLOGY

BTECH Programme: CSE (B)

Course Title: Advanced Java Programming Lab

Course Code: CIE-306P

Submitted By:

Name: Nidhi Rawat
Enrollment No: 36017702722

योगः कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS
Grade A++ Accredited Institution by NAAC
NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;
Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE
An ISO 9001:2015 Certified Institution

SCHOOL OF ENGINEERING & TECHNOLOGY

VISION OF INSTITUTE

To be an educational institute that empowers the field of engineering to build a sustainable future by providing quality education with innovative practices that supports people, planet and profit.

MISSION OF INSTITUTE

To groom the future engineers by providing value-based education and awakening students' curiosity, nurturing creativity and building
capabilities to enable them to make significant contributions to the world.

VIPS
योगः कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS
Grade A++ Accredited Institution by NAAC
NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;
Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE
An ISO 9001:2015 Certified Institution

SCHOOL OF ENGINEERING & TECHNOLOGY

INDEX

S.No	EXP.	Date	Marks			Remark	Updated Marks	Faculty Signature
			Lab. Assess. (15 Marks)	Class Part. (5 Marks)	Viva (5 Marks)			

EXPERIMENT 1

AIM

Write a Java program to print numbers from 1 to n. For multiples of 3, print "Fizz" instead of the number, and for multiples of 5, print "Buzz". For numbers that are multiples of both 3 and 5, print "FizzBuzz".

CODE

```
import java.util.*;
public class multiple {
  public static void main(String[] args){
     Scanner sc= new Scanner(System.in);
     System.out.print("Enter the total number of elements : ");
     int n = sc.nextInt();
     for(int i =1;i<=n;i++){
        if(i%3==0){
           System. out.println("Fizz");
        }
        else if(i%5==0){
           System.out.println("Buzz");
        }
        else if(i%3==0 && i%5==0){
           System.out.println("FizzBuzz");
        }
        else{
           System.out.println(i);
        }
     }
  }
}
```

OUTPUT

```
nidhirawat@Nidhis-MacBook-Pro  java lab % cd "/Us
java multiple
Enter the total number of elements : 10
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
```

LEARNING OUTCOME

EXPERIMENT 3

AIM

Write an efficient code in java to check all the prime numbers in a given range of numbers.

CODE

```
import java.util.Scanner;
public class prime{
 public static boolean checkPrime(int n) {
    if (n <= 1)
    return false;
    if (n == 2)
    return true;
    if (n % 2 == 0)
    return false;
   for (int i = 3; i <= Math.sqrt(n); i += 2) {
     if (n % i == 0) {
     return false;
    }
  }
  return true;
}
 public static void main(String[] args) {
   Scanner num = new Scanner(System.in);
   System.out.print("Enter start : ");
   int start = num.nextInt();
   System.out.print("Enter end : ");
   int end = num.nextInt();
   System.out.print("Prime numbers between " + start + " and " + end+ " are : ");
   for (int i = start; i <= end; i++) {
     if (checkPrime(i)) {
     System.out.print(i + " ");
     }
    }
    System.out.println();
 num.close();
}
}
```

OUTPUT

```
nidhirawat@Nidhis-MacBook-Pro  java lab % cd "/User
a prime
Enter start : 1
Enter end : 10
Prime numbers between 1 and 10 are : 2 3 5 7
nidhirawat@Nidhis-MacBook-Pro  java lab %
```

LEARNING OUTCOME

EXPERIMENT 2

AIM

Create an abstract class BankAccount with the following:

a)An accountNumber (String) and balance (double) as instance variables.

b)A constructor to initialize the account number and balance.

c)Abstract methods:

- deposit(double amount)
- withdraw(double amount)

d)Create two subclasses: SavingsAccount:

- Has an additional variable interestRate (double).
- Overrides the deposit method to add interest to the balance.
- Withdrawals are allowed only if the balance remains above a certain minimum (e.g., 500).

CurrentAccount:

- Has an additional variable overdraftLimit (double).
- Overrides the withdraw method to allow overdraft up to the specified limit.

Write a program that:

e)Creates objects of both subclasses.

f)Performs deposit and withdrawal operations.

g)Displays the final account details for each object.

THEORY

CODE

```
import java.util.*;
abstract class bankAccount{
    String accNo;
    double balance;
   public bankAccount(String accNo , double balance){
       this.accNo = accNo;
       this.balance = balance;
   }
   public abstract void deposit(double amount);
   public abstract void withdraw(double amount);

   public void displayDetail(){
       System.out.println("Account no  "  + accNo);
       System.out.println("Balance  " +balance);
     }
}
class SavingsAccount extends bankAccount{
   double interestRate;
   static final double minBalance = 500;

   public SavingsAccount(String accNo, double balance, double interestRate){
       super(accNo, balance);
       this. interestRate = interestRate;
   }
   public void deposit(double amount){
       balance += amount+(amount*interestRate/100);
   }
     public void withdraw(double amount){
       if(balance-amount>=minBalance){
           balance -= amount;
       }
       else{
           System.out.println("Withdrawal denied");
       }
   }
}
```

```
class  CurrentAccount extends bankAccount{
   double overdraftLimit;

   public CurrentAccount(String accNo, double balance , double overdraftLimit){
      super(accNo, balance);
      this.overdraftLimit = overdraftLimit;
   }
    public void deposit(double amount){
      balance+=amount;
   }
   @Override
   public void withdraw(double amount){
      if(balance- amount>=overdraftLimit){
         balance-=amount;
      }
      else{
         System.out.println("Withdrawn denied");
      }
   }
}
public class bankDemo{
   public static void main(String[] args) {
      SavingsAccount savings = new SavingsAccount("BR101",10000, 20);
      CurrentAccount current = new CurrentAccount("UK201",12300 ,1000);
      System.out.println("Savings Account:");
       savings.deposit(2000);
       savings.withdraw(10000);
       savings.displayDetail();
       System.out.println("");
       System.out.println("Current Account:");
       current.deposit(800);
       current.withdraw(5000);
       current.displayDetail();

   }
}
```

OUTPUT

```
Savings Account:
Account no  BR101
Balance  2400.0

Current Account:
Account no  UK201
Balance  8100.0
nidhirawat@Nidhis-MacBook-Pro
```

LEARNING OUTCOME

EXPERIMENT 4

AIM

N soldiers (or people) stand in a circle. The king gives a sword to the first soldier, who kills the person to their left and passes the sword to the next surviving person. This process continues until only one person remains. Write a java program to find the safe position to stand. Assuming first position is=1.

THEORY

CODE

```
import java.util.Scanner;
public class josephus {
  public static int josephus(int n, int k){
     if(n==1){
     return 1;
     } else {
        return (josephus(n-1, k)+k-1)% n +1;
     }   }
  public static void main(String[] args) {
     Scanner sc = new Scanner(System.in);
     System.out.print("Enter number of people \t");
     int n = sc.nextInt();
     int k = 2; //every 2nd person is killed
     int safePosition = josephus(n, k);
     System.out.println("Safe position is " + safePosition);
     sc.close();
  }}
```

OUTPUT

```
Enter number of people  200
Safe position is 145
nidhirawat@Nidhis-MacBook-Pro  java lab %
```

LEARNING OUTCOME

EXPERIMENT 5

AIM

Create an user defined exception named InvalidCibilScore if the cibil score of a customer is below 8.5.

THEORY

CODE

```
import java.util.*;
class InvalidCibilScore extends Exception{
public InvalidCibilScore(String message){
super(message);
}}
public class five {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.print("Enter your Cibil Score :");
int cs = sc.nextInt();
try{
if(cs < 8.5){
throw new InvalidCibilScore("Cibil Score is below 8.5");
}
```

```
else{
System.out.println("Valid Cibil Score");
}}
catch(InvalidCibilScore e){
System.out.println(e);
}
finally{
sc.close();
System.out.println("Program executed successfully");
}
}}
```

OUTPUT

```
Enter your Cibil Score :6
InvalidCibilScore: Cibil Score is below 8.5
Program executed successfully
nidhirawat@Nidhis-MacBook-Pro  java lab % cd
& java five
Enter your Cibil Score :9
Valid Cibil Score
Program executed successfully
nidhirawat@Nidhis-MacBook-Pro  java lab %
```

LEARNING OUTCOME

EXPERIMENT 6

AIM

Write a program to implement multithreading where thread 1 adds all even numbers and thread 2 adds all odd numbers from a given file which has large numbers of random positive number.

THEORY

CODE

```
import java.io.*;
import java.util.*;
class EvenSumThread extends Thread {
private List<Integer> numbers;
private int sum = 0;
public EvenSumThread(List<Integer> numbers) {
this.numbers = numbers;
}
public void run() {
for (int num : numbers) {
if (num % 2 == 0) {
sum += num;
}}
System.out.println("Sum of even numbers: " + sum);
}}
class OddSumThread extends Thread {
private List<Integer> numbers;
private int sum = 0; public OddSumThread(List<Integer> numbers) {
```

```
this.numbers = numbers;
}
public void run() {
for (int num : numbers) {
if (num % 2 != 0) {
sum += num;
}}
System.out.println("Sum of odd numbers: " + sum);
} }
public class six {
public static void main(String[] args) {
List<Integer> numbers = new ArrayList<>();
try (BufferedReader br = new BufferedReader(new
FileReader("randomNumbers.txt"))) {
String line;

while ((line = br.readLine()) != null) {
numbers.add(Integer.parseInt(line.trim()));
}
} catch (IOException e) {
System.out.println("Error reading file: " + e.getMessage());
return;
}
EvenSumThread evenThread = new EvenSumThread(numbers);
OddSumThread oddThread = new OddSumThread(numbers);
evenThread.start();
oddThread.start();
 try {
evenThread.join();
oddThread.join();
} catch (InterruptedException e) {
System.out.println("Thread interrupted: " + e.getMessage());
}
} }
```

randomNumbers.txt

6th sem > java lab > randomNumbers.txt

```
100
10000
39
59
30
20
500
904
100
401
```

OUTPUT

```
nidhirawat@Nidhis-MacBook-Pro  java lab %
 java six
Sum of odd numbers: 499
Sum of even numbers: 11654
nidhirawat@Nidhis-MacBook-Pro  java lab %
```

LEARNING OUTCOME

EXPERIMENT 7

AIM

Write a Java program to demonstrate the concept of socket programming.

THEORY

CODE

Server Side:

```
package seven;
import java.net.*;
import java.io.*;
public class Server {

    private Socket s = null;
    private ServerSocket ss = null;
    private DataInputStream in = null;
    public Server(int port) {

        // Starts server and waits for a connection
        try
        {
            ss = new ServerSocket(port);
            System.out.println("Server started");
            System.out.println("Waiting for a client ...");
            s = ss.accept();
            System.out.println("Client accepted");

            in = new DataInputStream(
                new BufferedInputStream(s.getInputStream()));
            String m = "";
            while (!m.equals("BYE"))
            {
                try
                {
                    m = in.readUTF();
                    System.out.println(m);
```

```
            }
            catch(IOException i)
            {
                System.out.println(i);
            }
        }
        System.out.println("Closing connection");
        // Close connection
        s.close();
        in.close();
    }
    catch(IOException i)
    {
        System.out.println(i);
    }
}
public static void main(String args[])
{
    Server s = new Server(5001);
}
}
```

Client Side:

```
package seven;
import java.io.*;
import java.net.*;
public class client {

    private Socket s = null;
    private DataInputStream in = null;
    private DataOutputStream out = null;
    public client(String addr, int port)
    {
        try {
            s = new Socket(addr, port);
            System.out.println("Connected");
            in = new DataInputStream(System.in);
            out = new DataOutputStream(s.getOutputStream());
        }
        catch (UnknownHostException u) {
            System.out.println(u);
            return;
        }
        catch (IOException i) {
            System.out.println(i);
            return;
        }
        String m = "";
        while (!m.equals("BYE")) {
            try {
```

```
            m = in.readLine();
            out.writeUTF(m);
        }
        catch (IOException i) {
            System.out.println(i);
        }
    }
    try {
        in.close();
        out.close();
        s.close();
    }
    catch (IOException i) {
        System.out.println(i);
    }
  }
  public static void main(String[] args) {
    client c = new client("127.0.0.1", 5001);
  }
}
```

OUTPUT

Server

```
Server started
Waiting for a client ...
Client accepted

Hello Java
This is socket programming
BYE
Closing connection
```

Client

```
Connected

Hello Java
This is socket programming
BYE
```

LEARNING OUTCOME

EXPERIMENT 8

AIM

Design a servlet and an html file for addition of two numbers.

THEORY

CODE

AdditionServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;
@WebServlet("/AdditionServlet") // URL mapping

public class AdditionServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
    // Set response content type
    response.setContentType("text/html");
    // Get numbers from the request
    int num1 = Integer.parseInt(request.getParameter("num1"));
    int num2 = Integer.parseInt(request.getParameter("num2"));
     // Calculate sum
    int sum = num1 + num2;
```

```
        // Display result
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h2>Result</h2>");
        out.println("<p>Number 1: " + num1 + "</p>");
        out.println("<p>Number 2: " + num2 + "</p>");
        out.println("<p><b>Sum: " + sum + "</b></p>");
        out.println("</body></html>");
    }
}
```

addNumbers.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial- scale=1.0">
    <title>Addition</title>
</head>
<body>
    <h1>Addition of two numbers</h1>
    <form action="AdditionServlet" method="post">
        <label for="num1">Enter first number:</label>
        <input type="number" name="num1" id="num1" required>
        <br>
        <label for="num2">Enter second number:</label>
        <input type="number" name="num2" id="num2" required>
        <br>
        <input type="submit" value="Add">
    </form>
</body>
</html>
```

OUTPUT

Addition of two numbers

Enter first number: 5
Enter second number: 6
Add

Result

Number 1: 5

Number 2: 6

Sum: 11

LEARNING OUTCOME