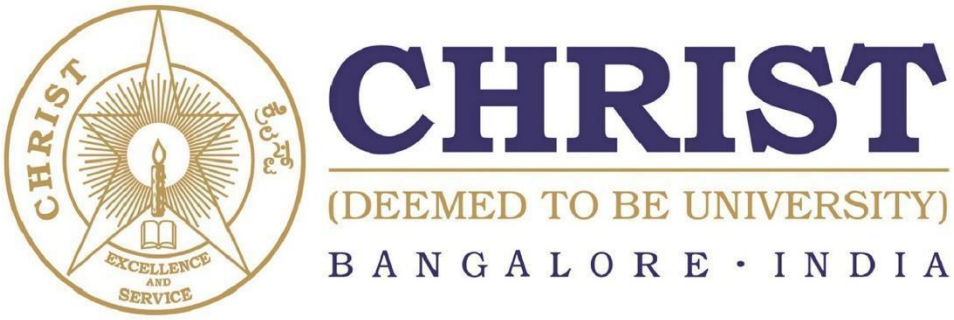

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

Programming Paradigm

CS433P

B. Tech – Computer Science and Engineering
(Artificial Intelligence and Machine Learning)

School of Engineering and Technology,

CHRIST (Deemed to be University),

Kumbalagodu, Bengaluru-560 074

April 2023

CHRIST
(DEEMED TO BE UNIVERSITY)
BANGALORE · INDIA

Certificate

This is to certify that Ashvath Suresh Babu Piriya has successfully completed the record / ~~Mini Project~~ work for (CS433P - Programming Paradigm) in partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineering (Artificial Intelligence & Machine Learning) during the year 2022-2023.

HEAD OF THE DEPARTMENT **FACULTY- IN CHARGE**

EXAMINER 1:

EXAMINER 2:

Name : Ashvath S.P

Register No. : 2162014

Examination Center : SoET, CHRIST (Deemed to be University)

Date of Examination :

DATE: 31-03-2023 **EXPERIMENT NO** 7 **REGISTER NO**: 2162014

IMPLEMENTATION OF EXCEPTIONS

AIM:

Write a java program that uses the try-catch-finally block to handle exceptions during the bank transfer. An exception should be thrown when the transfer amount exceeds the available balance in the customer's account.

PROGRAM:

```
/**
 *
 * @author 2162014
 */
public class excep {

    private double balance;

    public excep(double initialBalance) {
        balance = initialBalance;
    }

    public void transfer(double amount, excep recipient) throws
            InsufficientFundsException {
        try {
            if (amount > balance) {
                throw new InsufficientFundsException("Transfer amount exceeds available
balance");
            } else {
                balance -= amount;
                recipient.balance += amount;
                System.out.println("Transfer successful!");
            }
        } catch (InsufficientFundsException e) {
            System.out.println("Transfer failed: " + e.getMessage());
            throw e;
        }
    }

    public static void main(String[] args) {
        excep account1 = new excep(1000.0);
        excep account2 = new excep(500.0);
        try {
            System.out.println("Acc1 balance before transfer: " + account1.balance);
            System.out.println("Acc2 balance before transfer: " + account2.balance);
            account1.transfer(600.0, account2);
            // account2.transfer(400.0, account1);
```

```
        } catch (InsufficientFundsException e) {
            // Handle the exception here
        } finally {
            System.out.println("Acc1 balance after transfer: " + account1.balance);
            System.out.println("Acc2 balance after transfer: " + account2.balance);
        }
    }
}

class InsufficientFundsException extends Exception {

    public InsufficientFundsException(String message) {
        super(message);
    }
}
```

OUTPUTS:

```
PS C:\Users\ashva\                                                    \exp7> javac excep.java
PS C:\Users\ashva\                                                    \exp7> java excep
Acc1 balance before transfer: 1000.0
Acc2 balance before transfer: 500.0
Transfer failed: Transfer amount exceeds available balance
Acc1 balance after transfer: 1000.0
Acc2 balance after transfer: 500.0
PS C:\Users\ashva\                                                    \exp7> javac excep.java
PS C:\Users\ashva\                                                    \exp7> java excep
Acc1 balance before transfer: 1000.0
Acc2 balance before transfer: 500.0
Transfer successful!
Acc1 balance after transfer: 400.0
Acc2 balance after transfer: 1100.0
```

RESULTS:

The java program was created successfully to demonstrate the use of the try-catch-finally block to handle exceptions during the bank transfer

DATE: 31-03-2023 **EXPERIMENT NO** 8 **REGISTER NO**: 2162014

IMPLEMENTATION OF GENERIC PROGRAMMING

AIM:
Write a java program to implement Generic Programming.

PROGRAM:

```
/**
 *
 * @author 2162014
 */
import java.util.ArrayList;
import java.util.NoSuchElementException;

public class Stack<T> {

    private ArrayList<T> items;

    public Stack() {
        items = new ArrayList<>();
    }

    public void push(T item) {
        items.add(item);
    }

    public T pop() {
        if (isEmpty()) {
            throw new NoSuchElementException("Stack is empty");
        }
        return items.remove(items.size() - 1);
    }

    public T peek() {
        if (isEmpty()) {
            throw new NoSuchElementException("Stack is empty");
        }
        return items.get(items.size() - 1);
    }

    public boolean isEmpty() {
        return items.isEmpty();
    }

    public int size() {
        return items.size();
```

```
    }

    public static void main(String[] args) {
        Stack<Integer> intStack = new Stack<>();
        intStack.push(61);
        intStack.push(87);
        intStack.push(32);
        System.out.println("Top element: " + intStack.peek());
        System.out.println("Size of stack: " + intStack.size());
        while (!intStack.isEmpty()) {
            System.out.println(intStack.pop());
        }
    }
}
```

OUTPUTS:

```
PS C:\Users\ashva\                                          \exp8> javac Stack.java
PS C:\Users\ashva\                                          \exp8> java Stack
Top element: 32
Size of stack: 3
32
87
61
```

RESULTS:

The java program was created successfully implements Generic Programming.

DATE: 31-03-2023 **EXPERIMENT NO** 9 **REGISTER NO**: 2162014

IMPLEMENTATION OF MULTITHREADED PROGRAMS

AIM:
Write a java program to implement Multithreaded programs.

PROGRAM:

```
/**
 *
 * @author 2162014
 */
import java.util.Random;

public class th {

    public static void main(String[] args) {
        Random random = new Random();
        NumberGenerator numberGenerator = new NumberGenerator(random);
        SquareCalculator squareCalculator = new SquareCalculator();
        CubeCalculator cubeCalculator = new CubeCalculator();
        Thread generatorThread = new Thread(numberGenerator);
        Thread squareThread = new Thread(squareCalculator);
        Thread cubeThread = new Thread(cubeCalculator);
        generatorThread.start();
        squareThread.start();
        cubeThread.start();
    }
}

class NumberGenerator implements Runnable {

    private final Random random;

    public NumberGenerator(Random random) {
        this.random = random;
    }

    @Override
    public void run() {
        while (true) {
            int number = random.nextInt(10);
            if (number % 2 == 0) {
                SquareCalculator.handleNumber(number);
            } else {
                CubeCalculator.handleNumber(number);
            }
```

```
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class SquareCalculator implements Runnable {

    public static synchronized void handleNumber(int number) {
        System.out.println("Received an even number: " + number);
        int square = number * number;
        System.out.println("Square of the number: " + square);
    }

    @Override
    public void run() {
        // This thread doesn't need to do anything, as the handleNumber() method
        // is static and synchronized, so it can be called from any thread.
    }
}

class CubeCalculator implements Runnable {

    public static synchronized void handleNumber(int number) {
        System.out.println("Received an odd number: " + number);
        int cube = number * number * number;
        System.out.println("Cube of the number: " + cube);
    }

    @Override
    public void run() {
        // This thread doesn't need to do anything, as the handleNumber() method
        // is static and synchronized, so it can be called from any thread.
    }
}
```

OUTPUTS:

DATE: 31-03-2023 **EXPERIMENT NO** 9 **REGISTER NO**: 2162014

```
PS C:\Users\ashva\                                                \exp9> javac th.java
PS C:\Users\ashva\                                                \exp9> java th
Received an even number: 6
Square of the number: 36
Received an odd number: 1
Cube of the number: 1
Received an odd number: 3
Cube of the number: 27
Received an even number: 4
Square of the number: 16
Received an even number: 2
Square of the number: 4
Received an odd number: 5
Cube of the number: 125
Received an even number: 4
Square of the number: 16
Received an even number: 8
Square of the number: 64
Received an even number: 4
Square of the number: 16
Received an odd number: 7
Cube of the number: 343
Received an odd number: 1
Cube of the number: 1
Received an even number: 8
Square of the number: 64
Received an even number: 4
Square of the number: 16
Received an odd number: 5
Cube of the number: 125
Received an even number: 2
Square of the number: 4
```

RESULTS:

The java program was created successfully implements Multithreaded programs.