[Name:Valaki Jaymin D]
[RollNo: MA075]
[Date of Experiment: 29/03/2023]

Practical-7

Introduction to Exception Handling

1. Create a try block that is likely to generate divide by zero and StringIndexOutOfBoundException exceptions then incorporate necessary catch blocks to catch and handle them appropriately and also implement common exception block and finally block.

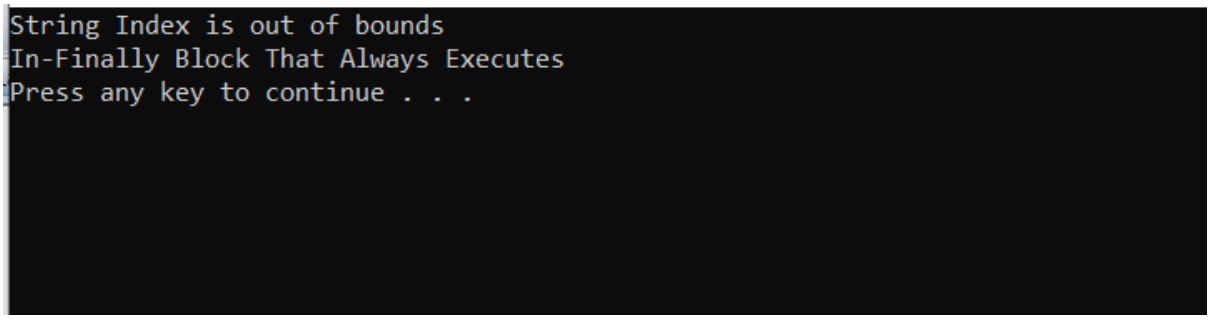
Code:-

```
class l7p1
{
	public static void main(String args[])
	{
		int num1, num2;
		String str = "Welcome";
		try
		{
			str.charAt(50);
			System.out.println("String Index is valid");
			num1 = 0;
			num2 = 20 / num1;
			System.out.println(num2);
			System.out.println("end of try block");
		}
		catch (StringIndexOutOfBoundsException e)
		{
			 System.out.println("String Index is out of bounds");
		}
		catch (ArithmeticException e)
		{
			System.out.println("Not divide a number by zero");
		}
```

```
            catch (Exception e)
            {
                    System.out.println("Exception occurred");
            }
            finally
            {
                        System.out.println("In-Finally Block That Always
                        Executes");
            }
        }

 }
```

Output:-



2. Write a Java program to create a class BankDemo which invokes the deposit() and withdraw() methods of the CheckingAccount class where withdraw() throws an InsufficientFundsException exception if the amount to be withdrawn is more than the available balance. (Use throws keyword)

Code:-

```
class InsufficientFundsException extends Exception
{
        private double amount;
        public InsufficientFundsException(double amount)
        {
                this.amount = amount;
```

```
}
	public double getAmount()
	{
		return amount;
	}
}
class CheckingAccount
{
	private double balance;
	private int number;
	public CheckingAccount(int number)
	{
		this.number = number;
	}
	public void deposit(double amount)
	{
		balance += amount;
	}
public void withdraw(double amount) throws InsufficientFundsException
{
	if(amount <= balance)
	{
		 balance -= amount;
	}
	else
	{
		double needs = amount - balance;
		throw new InsufficientFundsException(needs);
	}
```

```
 }
 public double getBalance()
 {
        return balance;
 }
 public int getNumber()
 {
        return number;
 }
}

public class l7p2
{
        public static void main(String [] args)
        {
                CheckingAccount c = new CheckingAccount(20000);
                System.out.println("Depositing Rs.15000...");
                c.deposit(15000.00);
                try
                {
                        System.out.println("\nWithdrawing Rs.18000...");
                        c.withdraw(18000.00);
                        System.out.println("\nWithdrawing Rs.18000...");
                        c.withdraw(18000.00);
                }
                catch (InsufficientFundsException e)
                {
                         System.out.println("paisa ocha che :" + e.getAmount());
                         e.printStackTrace();
                 }
```

```
        }
}
```

Output:-

```
D:\VS-CODE\JAVA PROGRAMMING>java l7p2
Depositing Rs.15000...

Withdrawing Rs.18000...
paisa ocha che :3000.0
InsufficientFundsException
        at CheckingAccount.withdraw(l7p2.java:37)
        at l7p2.main(l7p2.java:59)

D:\VS-CODE\JAVA PROGRAMMING>
```

3. Define an exception called “NoMatchException” that is thrown when a string is not equal to “Object Oriented Programing with JAVA”. Write a program that uses this exception. (Use throw keyword)

Code:-

```
class NoMatchException extends Exception
{
        NoMatchException(String msg)
        {
            super(msg);
        }
}


public class l7p3 {
   public static void main(String[] args) {
      String str = "jay";
      try {
         if (!str.equals("jaymin")) {
```

```
            throw new NoMatchException(str);
          }
        } catch (NoMatchException e) {
          System.out.println("string does not match");
        }
    }
}
```

Output:-