Name:	Debjit Ghosal
UID:	2023300065
Experiment No.	9A

AIM:	Exception Handling
Program 1	
PROBLEM STATEMENT :	**Write a java program to throw a exception (checked) for an employee details. a) If an employee name is a number, a name exception must be thrown. b) If an employee age is greater than 50, an age exception must be thrown. c) Or else an object must be created for the entered employee details** **Demonstarte different kinds of Exceptions:** **1-AirthmaticException: find HCF and LCM of two no's** **2- Array outofboud exception:** **3-NumberFormatException** **4-StringIndexOutOfBoundsException**
PROGRAM:	/* Write a java program to throw a exception (checked) for an employee details. a) If an employee name is a number, a name exception must be thrown. b) If an employee age is greater than 50, an age exception must be thrown. c) Or else an object must be created for the entered employee details Demonstarte different kinds of Exceptions: 1-ArithmaticException: find HCF and LCM of two no's 2- Array outofboud exception: 3-NumberFormatException

```
4-StringIndexOutOfBoundsException
*/


import java.util.Scanner;

class NameException extends Exception {
    public NameException(String message) {
        super(message);
    }
}

class AgeException extends Exception {
    public AgeException(String message) {
        super(message);
    }
}

class Employee {
    private String name;
    private int age;

    public Employee(String name, int age) throws NameException,
AgeException {
        if (name.matches("[0-9]")) {
            throw new NameException("Employee name cannot contain
numbers.");
        }
        if (age > 50) {
            throw new AgeException("Employee age cannot be greater than
50.");
        }
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }
```

```
    public int getAge() {
        return age;
    }
}

public class Main123 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter employee name: ");
        String name = scanner.nextLine();

        System.out.print("Enter employee age: ");
        int age = scanner.nextInt();

        try {
            Employee employee = new Employee(name, age);
            System.out.println("Employee details:");
            System.out.println("Name: " + employee.getName());
            System.out.println("Age: " + employee.getAge());
        } catch (NameException e) {
            System.out.println("Name Exception: " + e.getMessage());
        } catch (AgeException e) {
            System.out.println("Age Exception: " + e.getMessage());
        }
    }
}
```

RESULT:

```
lenovo@lenovo-ThinkCentre-neo-50s-Gen-3:~/Desktop/2023300065$ javac Main123.java
lenovo@lenovo-ThinkCentre-neo-50s-Gen-3:~/Desktop/2023300065$ java Main123
Enter employee name: Debjit
Enter employee age: 40
Employee details:
Name: Debjit
Age: 40
lenovo@lenovo-ThinkCentre-neo-50s-Gen-3:~/Desktop/2023300065$ java Main123
Enter employee name: 245
Enter employee age: 65
Name Exception: Employee name cannot contain numbers.
lenovo@lenovo-ThinkCentre-neo-50s-Gen-3:~/Desktop/2023300065$ java Main123
Enter employee name: Debjit
Enter employee age: 58
Age Exception: Employee age cannot be greater than 50.
```

Program 2	
PROBLEM STATEMENT :	**There is an abstract class Account** **Attribute:-** **● Name** **● Balance** **● Acc_No** **Method:-** **● Deposit - abstract method** **● withdraw - abstract method** **● display - abstract method** **Saving Account inherits the Account class and provides the implementation for the methods accordingly** **Saving Account class Attribute:-** **● interestRate** **● minBalance** **Method** **● addInterest: handle Arithmetic Exception** **● transfer():** **Note:** **● Balance cannot be less than 0.** **● In a Saving account if minBalance is set then for that the balance**

	cannot go less than that amount. If it goes, an error must be shown. **● let the user deposit to or withdraw from the account. For each transaction, a message is displayed to indicate the status of the transaction: successful or failed. In case of failure, the failure reason is reported.** **● The possible Exceptions are negative-amount-exception (in both deposit and withdraw transaction) and insufficient-amount-exception (in withdraw transaction).** **For the above scenario write an interactive program in Java. Also, show output for different use cases.**
PROGRAM:	/* There is an abstract class Account Attribute:- ● Name ● Balance ● Acc_No Method:- ● Deposit - abstract method ● withdraw - abstract method ● display - abstract method Saving Account inherits the Account class and provides the implementation for the methods accordingly Saving Account class Attribute:- ● interestRate

	● minBalance
	
	Method
	
	● addInterest: handle Arithmetic Exception
	
	● transfer():
	
	Note:
	
	● Balance cannot be less than 0.
	
	● In a Saving account if minBalance is set then for that the balance cannot go less than that amount. If it goes, an error must be shown.
	
	● let the user deposit to or withdraw from the account. For each transaction, a message is displayed to indicate the status of the transaction: successful or failed. In case of failure, the failure reason is reported.
	
	● The possible Exceptions are negative-amount-exception (in both deposit and withdraw transaction) and insufficient-amount-exception (in withdraw transaction).
	
	For the above scenario write an interactive program in Java. Also, show output for different use cases.
	
	*/
	
	import java.util.Scanner;
	
	abstract class Account {
	  String name;
	  double balance;

```
    int acc_No;

    abstract void deposit(double amount) throws
NegativeAmountException;
    abstract void withdraw(double amount) throws
NegativeAmountException, InsufficientAmountException;
    abstract void display();
}
class NegativeAmountException extends Exception {
    public NegativeAmountException(String message) {
        super(message);
    }
}
class InsufficientAmountException extends Exception {
    public InsufficientAmountException(String message) {
        super(message);
    }
}
class SavingAccount extends Account {
    double interestRate;
    double minBalance;

    SavingAccount(String name, double balance, int acc_No, double
interestRate, double minBalance) {
        this.name = name;
        this.balance = balance;
        this.acc_No = acc_No;
        this.interestRate = interestRate;
        this.minBalance = minBalance;
    }
    void deposit(double amount) throws NegativeAmountException {
        try {
            if (amount < 0) {
                throw new NegativeAmountException("Deposit amount
cannot be negative.");
            }
            balance += amount;
            System.out.println("Deposit Successful. Current Balance: " +
balance);
        } catch (NegativeAmountException e) {
```

```
            System.out.println("Error: " + e.getMessage());
        }
    }
    void withdraw(double amount) throws NegativeAmountException,
InsufficientAmountException {
        try {
            if (amount < 0) {
                throw new NegativeAmountException("Withdraw amount
cannot be negative.");
            }
            if (balance - amount < minBalance) {
                throw new InsufficientAmountException("Insufficient
balance to withdraw.");
            }
            balance -= amount;
            System.out.println("Withdrawal Successful. Current Balance: "
+ balance);
        }
 catch (NegativeAmountException | InsufficientAmountException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
    void display() {
        System.out.println("Account Holder: " + name);
        System.out.println("Account Number: " + acc_No);
        System.out.println("Current Balance: " + balance);
    }

    void addInterest() {
        try {
            balance += balance * interestRate / 100;
            System.out.println("Interest Added. Current Balance: " +
balance);
        } catch (ArithmeticException e) {
            System.out.println("Error adding interest: " + e.getMessage());
        }
    }
    void transfer(double amount, Account recipient) throws
NegativeAmountException, InsufficientAmountException {
        try {
```

```
            if (amount < 0) {
                throw new NegativeAmountException("Transfer amount
cannot be negative.");
            }
            if (balance - amount < minBalance) {
                throw new InsufficientAmountException("Insufficient
balance to transfer.");
            }
            withdraw(amount);
            recipient.deposit(amount);
            System.out.println("Transfer Successful.");
        } catch (NegativeAmountException |
InsufficientAmountException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}

public class Main12345 {
    public static void main(String[] args) throws
NegativeAmountException, InsufficientAmountException {
        Scanner scanner = new Scanner(System.in);

        SavingAccount account = new SavingAccount("John Doe", 1000,
123456, 5, 500);

        while (true) {
            System.out.println("\nChoose an option:");
            System.out.println("1. Deposit");
            System.out.println("2. Withdraw");
            System.out.println("3. Display Account Info");
            System.out.println("4. Add Interest");
            System.out.println("5. Transfer");
            System.out.println("6. Exit");

            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter deposit amount: ");
```

```
                double depositAmount = scanner.nextDouble();
                account.deposit(depositAmount);
                break;
            case 2:
                System.out.print("Enter withdrawal amount: ");
                double withdrawAmount = scanner.nextDouble();
                account.withdraw(withdrawAmount);
                break;
            case 3:
                account.display();
                break;
            case 4:
                account.addInterest();
                break;
            case 5:
                System.out.print("Enter transfer amount: ");
                double transferAmount = scanner.nextDouble();
                System.out.print("Enter recipient account number: ");
                int recipientAccountNo = scanner.nextInt();
                SavingAccount recipient = new
SavingAccount("Recipient", 0, recipientAccountNo, 0, 0);
                try {
                    account.transfer(transferAmount, recipient);
                } catch (NegativeAmountException |
InsufficientAmountException e) {
                    System.out.println("Error: " + e.getMessage());
                }
                break;
            case 6:
                System.out.println("Exiting...");
                System.exit(0);
                break;
            default:
      System.out.println("Invalid choice. Please choose a valid option.");
          }
       }
   }
}
```

RESULT:

```
PS C:\Users\DEBJIT GHOSAL\Desktop\SPIT_CODING\PSOOP\PSOOP>  c:; cd 'c:\Users\DEBJIT GHOSAL\Desktop\SPIT_CODING\P
n\java.exe' '-cp' 'C:\Users\DEBJIT GHOSAL\AppData\Roaming\Code\User\workspaceStorage\9e74b28faa244b7149de54e209c
ceptionHandling.Main12345'

Choose an option:
1. Deposit
2. Withdraw
3. Display Account Info
4. Add Interest
5. Transfer
6. Exit
2
Enter withdrawal amount: 200
Withdrawal Successful. Current Balance: 800.0

Choose an option:
1. Deposit
2. Withdraw
3. Display Account Info
4. Add Interest
5. Transfer
6. Exit
3
Account Holder: John Doe
Account Number: 123456
Current Balance: 800.0

Choose an option:
1. Deposit
2. Withdraw
3. Display Account Info
4. Add Interest
5. Transfer
6. Exit
1
Enter deposit amount: 200
Deposit Successful. Current Balance: 1000.0
```

```
Choose an option:
1. Deposit
2. Withdraw
3. Display Account Info
4. Add Interest
5. Transfer
6. Exit
5
Enter transfer amount: 500
Enter recipient account number: 987654
Withdrawal Successful. Current Balance: 500.0
Deposit Successful. Current Balance: 500.0
Transfer Successful.

Choose an option:
1. Deposit
2. Withdraw
3. Display Account Info
4. Add Interest
5. Transfer
6. Exit
4
Interest Added. Current Balance: 525.0

Choose an option:
1. Deposit
2. Withdraw
3. Display Account Info
4. Add Interest
5. Transfer
6. Exit
2
Enter withdrawal amount: 600
Error: Insufficient balance to withdraw.

Choose an option:
1. Deposit
2. Withdraw
3. Display Account Info
```

```
Enter recipient account number: 987654
Withdrawal Successful. Current Balance: 500.0
Deposit Successful. Current Balance: 500.0
Transfer Successful.

Choose an option:
1. Deposit
2. Withdraw
3. Display Account Info
4. Add Interest
5. Transfer
6. Exit
4
Interest Added. Current Balance: 525.0

Choose an option:
1. Deposit
2. Withdraw
3. Display Account Info
4. Add Interest
5. Transfer
6. Exit
2
Enter withdrawal amount: 600
Error: Insufficient balance to withdraw.

Choose an option:
1. Deposit
2. Withdraw
3. Display Account Info
4. Add Interest
5. Transfer
6. Exit
        6
Exiting...
PS C:\Users\DEBJIT GHOSAL\Desktop\SPIT_CODING\PSOOP\PSOOP>
```

CONCLUSION:	I have learnt Exception Handling and keywords throws and finally.