Ex-no:5 Name: M.Rohith
Date: 7-11-22 3122 21 5001 085

SSN College of Engineering

Department of Computer Science and Engineering

UCS2313 – Object Oriented Programming Lab

II Year CSE - B Section (III Semester)

Academic Year 2022-23

Batch: 2021- 2025

Faculty Incharge :S.Rajalakshmi

Exercise – 5 – Exception handling

Objective:

1. To test the working of exception handling mechanism in Java

Sample Learning Outcome:

1. Need of exception handling and it's implementation in Java

2. User defined exception creation and usage

3. Usage of try, catch, finally, throw, throws

Best Practices:

1. Class Diagram usage

2. Naming convention – for file names, variables

3. Comment usage at proper places

4. Prompt messages during reading input and displaying output

5. Incremental program development

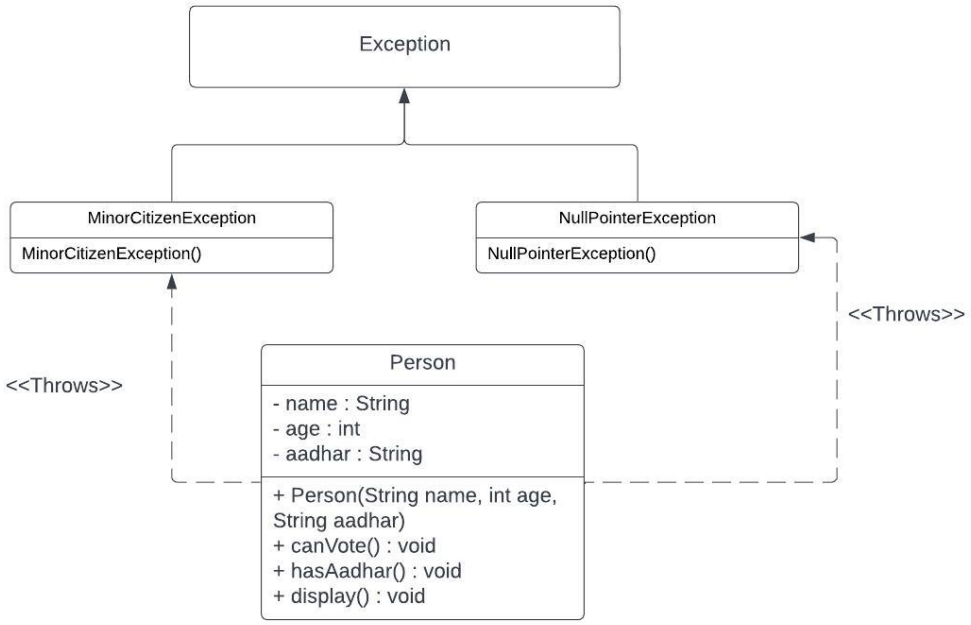
6. Modularity

7. All possible test cases in output

Ex-no:5 Name: M.Rohith
Date: 7-11-22 3122 21 5001 085

1. **Create a class named "Person" which consists of name, age, aadharnumber. Create methods getInput(), display(), canVote(), hasAadhar(). Create and handle the following Exceptions.**
    a. **For age -> if you give alphabets then throw NumberFormatException (Check for the condition explicitly and throw builtin exception)**
    b. **For voting -> if age is less than 18 then throw MinorCitizenException (Check for the condition explicitly and throw user-defined exception)**
    c. **For aadhar -> if no valid aadhar then throw NullPointerException (Check for the condition explicitly and throw builtin exception)**

Class Diagram:



Ex-no:5 Name: M.Rohith
Date: 7-11-22 3122 21 5001 085

Program Code:

```
import java.util.*;
class MinorCitizenException extends Exception
{
    MinorCitizenException()
    {
        super("MinorCitizenException : Age less than 18");
    }

}

class Person
{
    private String name;
    private int age;
    private String aadhar;

    Person(String name, int age, String aadhar)
    {
        this.name = name;
        this.age = age;
        this.aadhar = aadhar;
    }

    void canVote() throws MinorCitizenException
    {
        if(age<18)
            throw new MinorCitizenException();
        else
            System.out.println("Can vote");
    }

    void hasAadhar()
    {
        if(aadhar.length() < 12)
            throw new NullPointerException();
        else
            System.out.println("Has valid aadhar");
    }


    void display()
    {
        System.out.println("***Displaying Details***");
        System.out.println("Name : "+this.name);
        System.out.println("Age :"+this.age);
        System.out.println("Aadhaar :"+this.aadhar);
```

Name: M.Rohith
3122 21 5001 085

```
    }
}


class Main
{
    public static void main(String args[])
    {
        int age;
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter name :");
        String name = scan.next();
        System.out.print("Enter age :");
        try
        {
            age = scan.nextInt();
        }
        catch(InputMismatchException ex)
        {
            System.out.println("Enter valid number!!");
            scan.nextLine();
            System.out.print("Enter age :");
            age = scan.nextInt();
        }

        System.out.print("Enter aadhar :");
        String aadhar = scan.next();


        Person p = new Person(name, age, aadhar);
        p.display();
        try
        {
            p.canVote();
        }
        catch(MinorCitizenException e)
        {
            System.out.println(e);
        }

        try
        {
            p.hasAadhar();
        }
        catch(NullPointerException e)
        {
            System.out.print(e);
        }
    }
```

```
}
```

Output:

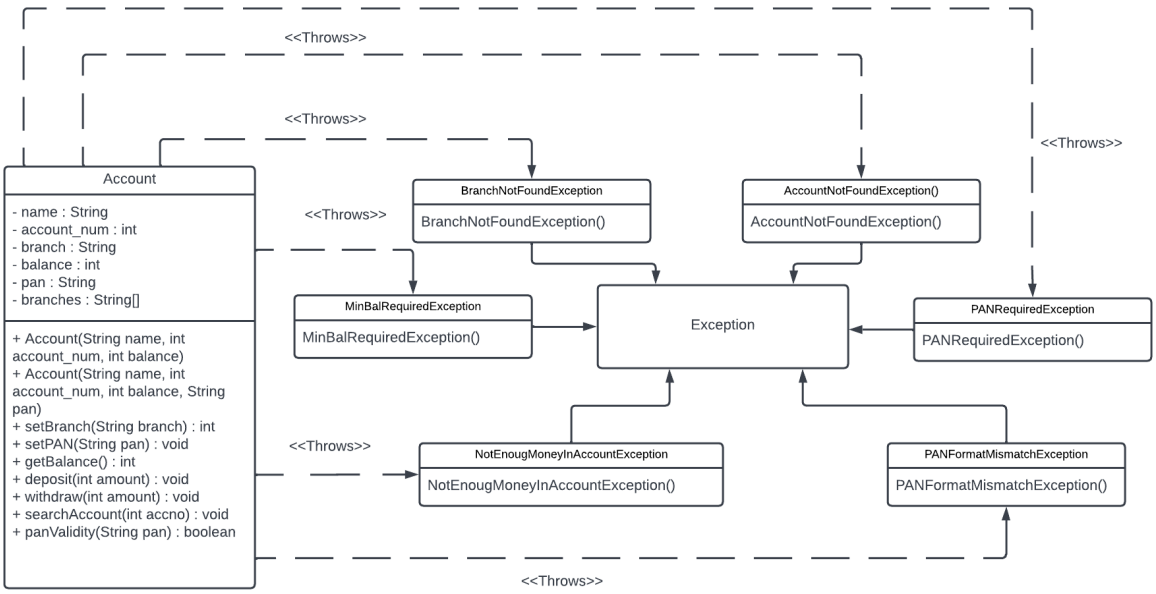
```
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Java programs\Lab programs\Exercise-5> javac Voting.java
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Java programs\Lab programs\Exercise-5> java Main
Enter name :Rohith
Enter age :abc
Enter valid number!!
Enter age :15
Enter aadhar :3122215001085
***Displaying Details***
Name : Rohith
Age :15
Aadhaar :3122215001085
MinorCitizenException: MinorCitizenException : Age less than 18
Has valid aadhar
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Java programs\Lab programs\Exercise-5> javac Voting.java
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Java programs\Lab programs\Exercise-5> java Main
Enter name :Rohith
Enter age :19
Enter aadhar :3122abcd
***Displaying Details***
Name : Rohith
Age :19
Aadhaar :3122abcd
Can vote
java.lang.NullPointerException
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Java programs\Lab programs\Exercise-5>
```

Ex-no:5 Name: M.Rohith
Date: 7-11-22 3122 21 5001 085

2. **Create a class named "Account" which contains name, acct_num, branch, balance, PAN_num. Create functions for deposit and withdrawal. Write <u>user-defined exceptions</u> for the following conditions:**
    a. **In deposit function, if the customer deposits money more than 25000, then throw the user defined exception "PANRequiredException" and get the PAN number and proceed the deposit.**
    b. **In withdrawal function, if the customer requesting some money, check on withdrawal will it satisfy the minimum_bal amount and throw the "MinBalRequiredException" exception. If the withdrawal amount is more than the balance amount then throw "NotEnougMoneyInAccountException".**
    c. **Search for a particular acct_num. If not present then throw "AccountNotFoundException".**
    d. **On PAN number entry check the format of 10 characters. First 5 characters then 4 numbers and then 1 character. If the format not matched then throw "PANFormatMismatchException".**

On account creation if user gives a non branch, then throw "BranchNotFoundException.

<u>Class Diagram:</u>



Ex-no:5 Name: M.Rohith
Date: 7-11-22 3122 21 5001 085

Program code:

```
import java.util.Scanner;

class PANRequiredException extends Exception
{
    PANRequiredException()
    {
        super("PAN required for transcation of amount more then 25,000");
    }
}

class MinBalRequiredException extends Exception
{
    MinBalRequiredException()
    {
        super("Balance less than min. required balance for withdrawal");
    }
}


class NotEnougMoneyInAccountException extends Exception
{
    NotEnougMoneyInAccountException()
    {
        super("Withdrawal amount exceeded current balance");
    }
}

class PANFormatMismatchException extends Exception
{
    PANFormatMismatchException()
    {
        super("Invalid PAN");
    }
}

class BranchNotFoundException extends Exception
{
    BranchNotFoundException()
    {
        super("Invalid Branch");
    }
}

class AccountNotFoundException extends Exception
{
    AccountNotFoundException()
    {
```

Name: M.Rohith
3122 21 5001 085

```
        super("Account not found");
    }
}


class Account
{
    private String name;
    private int account_num;
    private String branch;
    private int balance;
    private String pan;

    public String branches[]= new String[]
{"Tiruppur","Coimbatore","Thoothukudi","Chennai"};

    public Account(String name, int account_num, int balance)
    {
        this.name = name;
        this.account_num = account_num;
        this.balance = balance;
        this.pan = "Null";
    }

    public int setBranch(String branch) throws BranchNotFoundException
    {
        for(String b : this.branches)
        {
            if (branch.equals(b))
            {
                this.branch = branch;
                return 1;
            }
        }
        this.branch = "Null";
        throw new BranchNotFoundException();
    }
    public void setPAN(String pan) throws PANFormatMismatchException
    {
        if(this.panValidity(pan))
            this.pan = pan;
        else
            throw new PANFormatMismatchException();
    }

    public Account(String name, int account_num, int balance, String pan)
throws PANFormatMismatchException
    {
        this(name, account_num, balance);
```

```
        setPAN(pan);

    }

    public int getBalance()
    {
        return this.balance;
    }

    public void deposit(int amount) throws PANRequiredException
    {
        if(amount > 25000)
        {   if(pan.equals("Null"))
            {
                throw new PANRequiredException();
            }
            else if(this.panValidity(pan))
            {
                this.balance = this.balance + amount;
            }
        }
        else
            {
                //System.out.println("Less than 25k");
               this.balance = this.balance + amount/2;
            }
    }

    public void withdraw(int amount) throws MinBalRequiredException,
NotEnougMoneyInAccountException
    {
        if(balance<500)
            throw new MinBalRequiredException();
        if(amount>this.balance)
            throw new NotEnougMoneyInAccountException();
        else
            this.balance = this.balance - amount;
    }

    public void searchAccount(int accno) throws AccountNotFoundException
    {
        if(this.account_num == accno)
        {
            System.out.println("Account name :" + this.name);
            System.out.println("Account number :" + this.account_num);
            System.out.println("Account balance :" + getBalance());
            System.out.println("Branch :" + this.branch);
            System.out.println("PAN :" + this.pan);
```

Name: M.Rohith
3122 21 5001 085

```
        }

    }

    public boolean panValidity(String pan)
    {
        boolean t = true;
        if(pan.length() == 10)
        {
            for(int i=0; i<5; ++i)
            {
                char ch=pan.charAt(i);
                if(!(Character.isLetter(ch)))
                    return false;
                else t=true;
            }
            for(int i=5; i<9; ++i)
            {
                char ch=pan.charAt(i);
                if(!(Character.isDigit(ch)))
                    return false;
                else t = true;
            }
            char ch=pan.charAt(9);
            if(!(Character.isLetter(ch))) return false;
            else t= true;
        }
        return t;
    }
}

class Main
{
    public static void main(String args[])
    {
        //creating account
        String pan = "Null";
        String branch = "Null";

        Scanner scan = new Scanner(System.in);
        System.out.print("Enter name :");
        String name = scan.next();
        System.out.print("Enter account number :");
        int account_num = scan.nextInt();
        System.out.print("Enter balance :");
        int balance = scan.nextInt();
        System.out.print("Enter branch :");
        branch = scan.next();
```

Name: M.Rohith
3122 21 5001 085

```
        Account account = new Account(name, account_num, balance);

        try
        {
            account.setBranch(branch);
        }

        catch(BranchNotFoundException e)
        {
            System.out.print(e);
        }

        System.out.print("Enter amount to deposit :");
        int deposit= scan.nextInt();

        try
        {
        account.deposit(deposit);
        }
        catch(PANRequiredException e)
        {
            System.out.println(e);
            System.out.println("Enter PAN :");
            pan = scan.next();
        }

        try
        {
            account.setPAN(pan);
        }
        catch(PANFormatMismatchException e)
        {
            System.out.println(e);
        }

        try
        {
        account.deposit(deposit);
        }
        catch(PANRequiredException e)
        {
            System.out.println(e);
        }

        System.out.print("After deposit Balance : ");
        System.out.println(account.getBalance());
```

```
        System.out.print("Enter amount to withdraw :");
        int w = scan.nextInt();

        try {
            account.withdraw(w);
        }

        catch(MinBalRequiredException e)
        {
            System.out.println(e);
        }

        catch(NotEnougMoneyInAccountException e)
        {
            System.out.print(e);
        }

        System.out.print(" After withdrawal Balance : ");
        System.out.println(account.getBalance());
    }
}
```

Output:

```
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Java programs\Lab programs\Exercise-5> javac Banking.java
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Java programs\Lab programs\Exercise-5> java Main
Enter name :Roav
Enter account number :12132
Enter balance :10000
Enter branch :Chennai
Enter amount to deposit :10000
After deposit Balance : 20000
Enter amount to withdraw :15000
 After withdrawal Balance : 5000
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Java programs\Lab programs\Exercise-5> javac Banking.java
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Java programs\Lab programs\Exercise-5> java Main
Enter name :Wow
Enter account number :231342
Enter balance :20000
Enter branch :Pondicherry
BranchNotFoundException: Invalid BranchEnter amount to deposit :40000
PANRequiredException: PAN required for transcation of amount more then 25,000
Enter PAN :
12232adasd
PANFormatMismatchException: Invalid PAN
PANRequiredException: PAN required for transcation of amount more then 25,000
After deposit Balance : 20000
Enter amount to withdraw :19000
 After withdrawal Balance : 1000
PS C:\Rohith\Backup\Desktop\SEM 3\OOP-Java\Java programs\Lab programs\Exercise-5>
```

Learning Outcomes:

Thus the working of exception handling in Java has been implemented and executed successfully in various programs.