

**Title of the Experiment (Exception Handling )****Experiment No.** \_\_08\_\_**Date :** \_\_24/12/20\_\_**Problem Statement :**

Assume that you have received a request from the transport authority for automating the task of issuing the permanent license for two wheelers. The mandatory condition to issue the license are: 1) the applicant must over 18 years of age and 2) holder of a valid learner's license and 3) no accident cases in the last one year.

Write a Java program that reads user details as required (use the Scanner class). Create user defined exceptions to check for the three conditions imposed by the transport authority. Based on the inputs entered by the user, decide and display whether or not a license has to be issued or an error message as defined by the user exception.

**Objectives of the Experiment :**

1. Learn the use of Exception Handling in Java
2. Understand the use of Exception Handling in a real-life application
3. To develop user defined exceptions.
4. Learn to Display the result in a readable/proper format

**Program Source Code :**

```
package tw8;

import java.util.Scanner;

class License{// class named License is created

    String gender,name;

    Intno_of_cases,age;

    char valid_LL;

    // necessary variables are declared

    License() // constructor is defined

    {

        Scanner s = new Scanner(System.in);

        System.out.println("Name of the Applicant    :");

        name = s.nextLine();
```

```

System.out.println("Age          :");

    age = Integer.parseInt(s.nextLine());

System.out.println("Gender (M/F)      :");

    gender = s.nextLine();

System.out.println("Is valid LL issued (Y/N)?   :");

Valid_LL = s.nextLine().charAt(0);

System.out.println("Num of accidents in past year :");

no_of_cases =Integer.parseInt(s.nextLine());

System.out.println();

// the variables will be initialised by the users input
}

void validateData() // method to validate the data provided by the user
{
try // try block to check for user defined exceptions
{
    if(age<18)

        throw new UnderAgeException("Invalid Age");

    if(valid_LL!='Y')

        throw new InvalidLLException("Invalid LLR");

    if(no_of_cases>0)

        throw new AccidentException("Involved in Accidents ");

System.out.println("The License can be issued ");

}

catch(UnderAgeException e) /* catch block to provide the message related to under age exception*/

{

System.out.println(e.getMessage());

System.out.println(e);

}

```

```
catch(InvalidLLEException e) /* catch block to provide the message related to Invalidlearning license exception*/
```

```
{  
System.out.println(e.getMessage());  
System.out.println(e);  
}
```

```
catch(AccidentException e) // catch block to provide the message related to accident Exception
```

```
{  
System.out.println(e.getMessage());  
System.out.println(e);  
}  
}  
}
```

```
class UnderAgeException extends Exception /* subclass UnderAgeException is extending the superclass Exception*/
```

```
{  
UnderAgeException(String msg)  
{  
    super(msg);  
}  
public String toString(){ // overriding the toString method.  
    return "The person is underage";  
}  
}
```

```
class InvalidLLEException extends Exception/*subclass InvalidLLEException is extending the superclass Exception*/
```

```
{
```

```

InvalidLLException(String msg)
{
    super(msg);
}

public String toString(){ // overriding the toString method.
    return "The person hasn't issued for the LLR yet";
}
}

class AccidentException extends Exception /*subclass AccidentException is extending the superclass
Exception.*/
{
    AccidentException(String msg)
    {
        super(msg);
    }

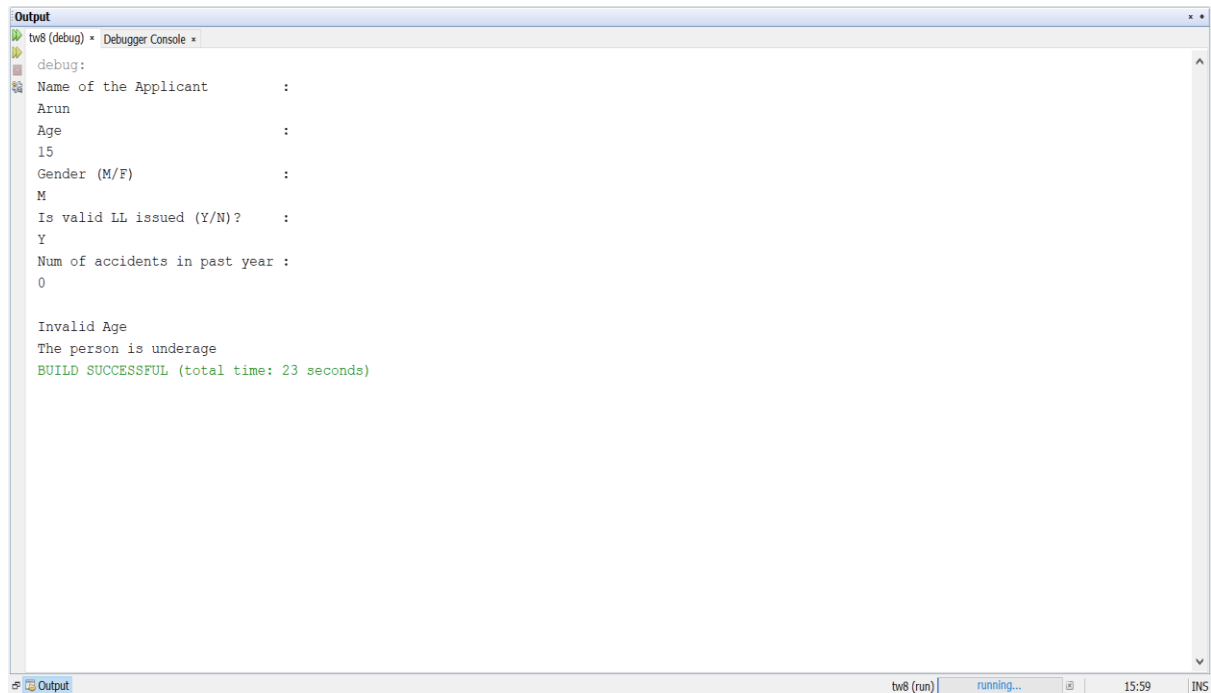
    public String toString(){ // overriding the toString method.
        return "The person has many accidents case on him";
    }
}

public class Tw8 {
    public static void main(String[] args) {
        License l = new License(); // object of class license is instantiated.
        l.validateData(); // method validate data is called
    }
}

```

## Output :

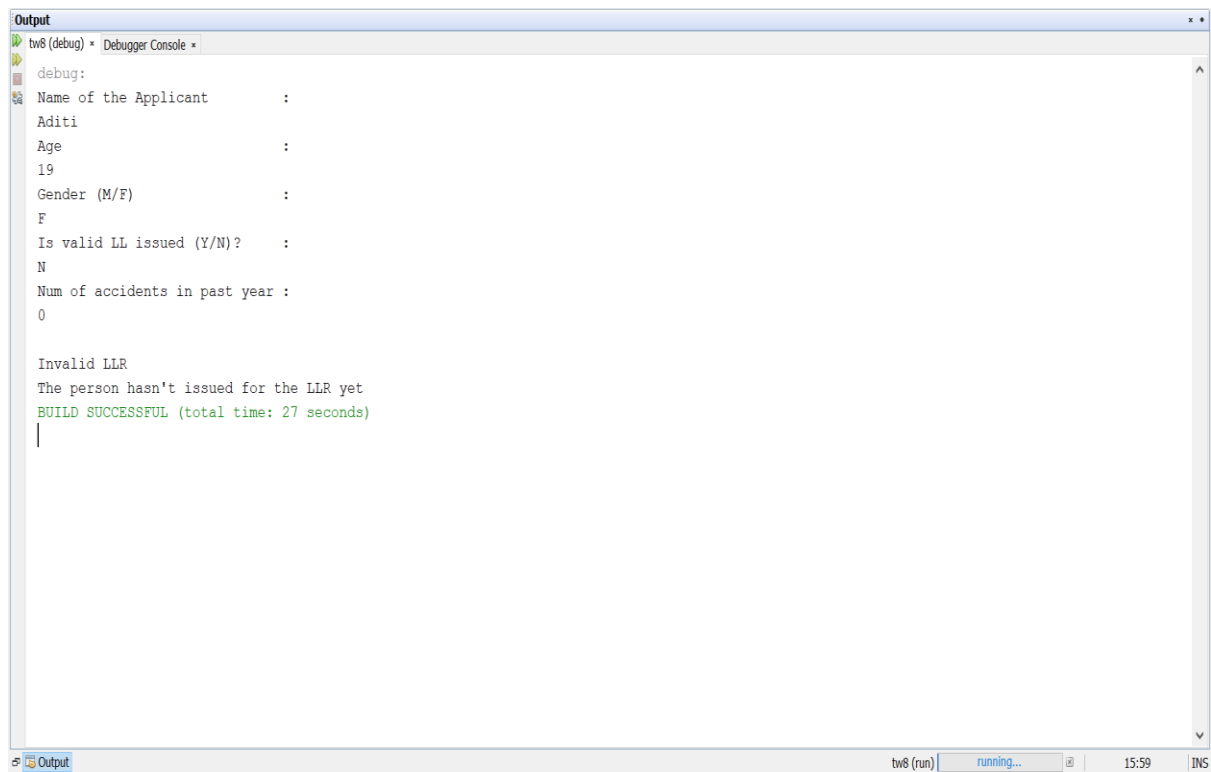
### Case 1:



```
debug:
Name of the Applicant      :
Arun
Age                        :
15
Gender (M/F)              :
M
Is valid LL issued (Y/N)?  :
Y
Num of accidents in past year :
0

Invalid Age
The person is underage
BUILD SUCCESSFUL (total time: 23 seconds)
```

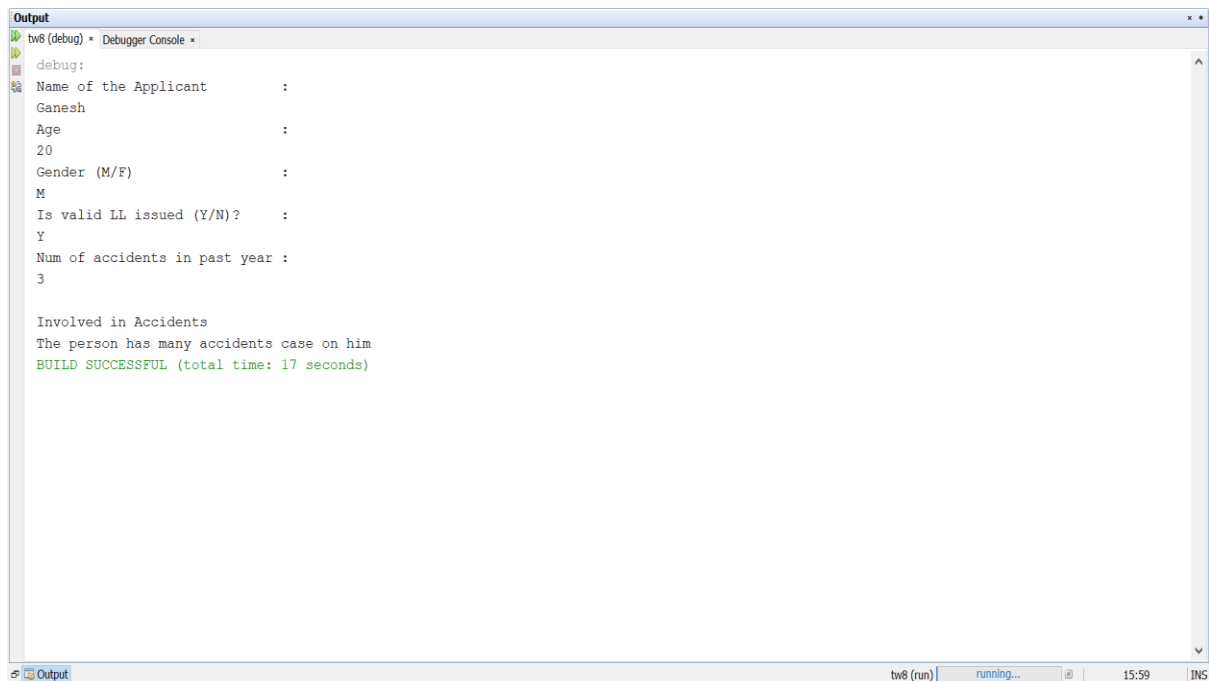
### Case 2:



```
debug:
Name of the Applicant      :
Aditi
Age                        :
19
Gender (M/F)              :
F
Is valid LL issued (Y/N)?  :
N
Num of accidents in past year :
0

Invalid LLR
The person hasn't issued for the LLR yet
BUILD SUCCESSFUL (total time: 27 seconds)
```

### Case 3:



The screenshot shows a Java IDE's Output window. The title bar reads 'Output'. Below it, a tab labeled 'tw8 (debug) \* Debugger Console \*' is active. The main area displays the following text:

```
debug:
Name of the Applicant      :
Ganesh
Age                        :
20
Gender (M/F)              :
M
Is valid LL issued (Y/N)?  :
Y
Num of accidents in past year :
3

Involved in Accidents
The person has many accidents case on him
BUILD SUCCESSFUL (total time: 17 seconds)
```

At the bottom of the window, a status bar shows 'tw8 (run)' in a blue box, followed by 'running...' in a light blue box, a small icon, the time '15:59', and the text 'JNS'.

**Outcomes of the Experiment :** At the end of the laboratory sessions the students should be able to

1. Demonstrate the use Exception Handling in solving real-life problems.
2. Identify appropriate variables and their types
3. Identify appropriate user definedExceptions
4. Identify the control statements needed to meet the problem requirements.

**Conclusions :** From the given problem statement, we could identify the necessary variables of appropriate type, and looping/control statements and the necessary program logic. The program was written in NetBeans IDE(Mention the one you actually used) by creating a project. We understood the usage of the IDE in typing the code, debugging, running the program and observing the output. We came to know about the Exceptions and ways of handling those Exceptions also understand how to define our own Exceptions and implement it on the given problem statement.

We also understood the use of built-in class System and its method println to display the result. The program was executed for two-three sets of input and result obtained were verified to be correct and recorded.

**Practice Problem:**

Demonstrate a class CustomerAccount that has acctNum, custName and balance as member variables and a constructor to initialize these. Implement withdraw and depositAmount methods that accept amount as argument and it must throw a user-defined exception called InsufficientBalance/InvalidAmount exception when amount is greater than balance/ amount is negative respectively. Design two classes InsufficientBalance and InvalidAmount that extend the Exception class and override toString method. Demonstrate the working of the user-defined exceptions by instantiating an object of CustomerAccount class and invoking withdraw and depositAmount in try...catch....finally block.

**Program Source Code :**

```
package tw8p;

import java.util.Scanner;

class CustomerAccount{ // class CustomerAccount is created

    int accNo;

    String custName;

    double balance;// required variables are initialised

    CustomerAccount(int accNo,String custName,double balance)// constructor of the class

    {

        this.accNo = accNo;

        this.custName = custName;

        this.balance = balance;

        // variables are initialised in the constructor

    }

    void withdrawAmt(double Amt) // method used to withdraw the amount

    {

        try{// try block to check for exception when the amt is withdrawn

            if(balance-Amt <0)

                throw new InsufficientBalException("Insufficient Balance");

            balance-=Amt;
```

```
System.out.println("Balance in the account is "+ balance);

    }

    catch (InsufficientBalException e) // catch block to deal with the insufficient Balance Exception

    {

System.out.println(e.getMessage());

System.out.println(e);

System.out.println();

    }

}

void depositAmt(double Amt) // method used to deposit the amount

{

    try // try block to check for exception while depositing the amount

    {

if(Amt<0)

        throw new InvalidAmtException("Invalid amount");

        balance+=Amt;

System.out.println("Balance in the account is "+ balance);

    }

catch( InvalidAmtException e) // catch block to deal with the invalid amount exception

    {

System.out.println(e.getMessage());

System.out.println(e);

System.out.println();

    }

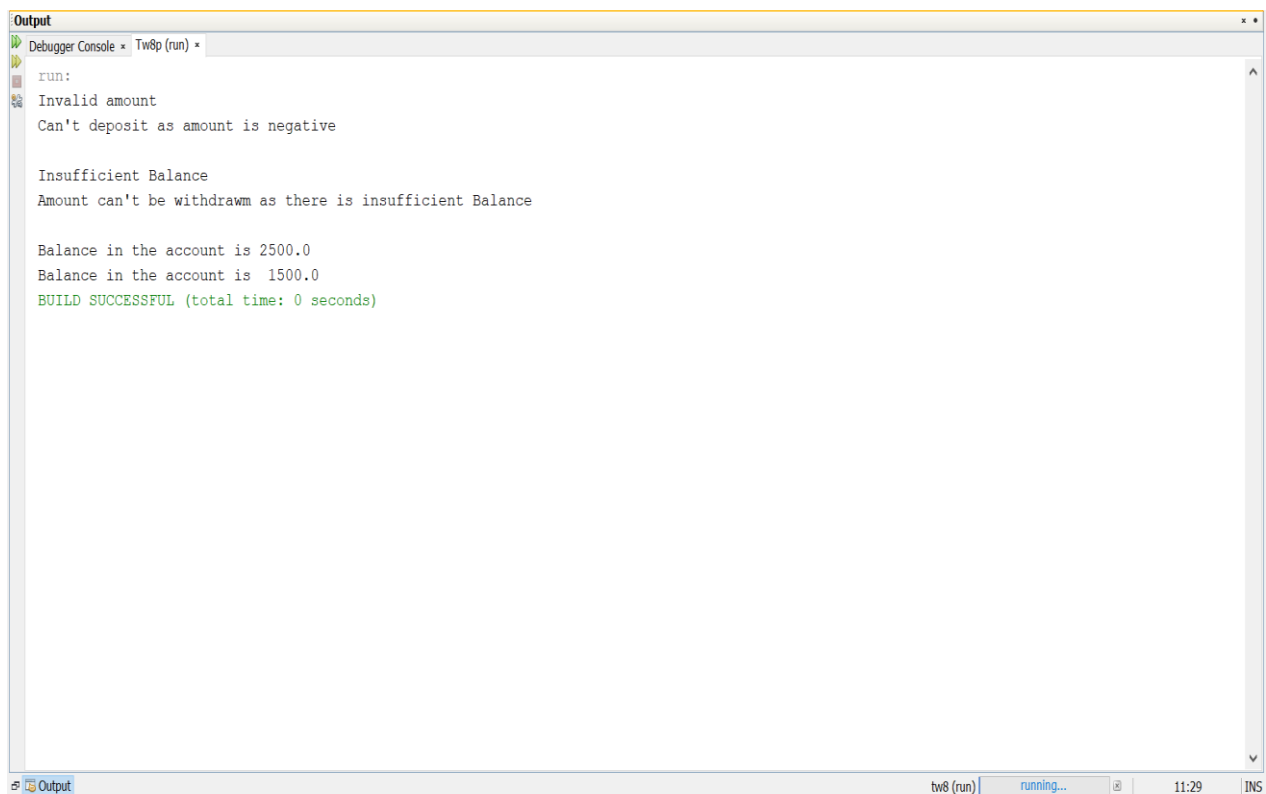
}
```



```
    }  
}  
  
class InsufficientBalException extends Exception{ /* subclass insufficientBalException is extending  
superclass Exception*/  
  
InsufficientBalException(String msg)  
  
    {  
  
        super(msg);  
  
    }  
  
    public String toString()// toString method is overridden  
  
    {  
  
        return "Amount can't be withdrawm as there is insufficient Balance";  
  
    }  
}  
  
class InvalidAmtException extends Exception{/* subclass invalidAmtException is extending  
superclass Exception*/  
  
InvalidAmtException(String msg)  
  
    {  
  
        super(msg);  
  
    }  
  
    public String toString()// toString method is overridden  
  
    {  
  
        return "Can't deposit as amount is negative";  
  
    }  
}
```

```
public class Tw8p {  
  
    public static void main(String[] args) {  
  
        CustomerAccount c =new CustomerAccount(101,"Amit shah",1000);/*object is instantiated of the  
        class CustomerAccount */  
  
        c.depositAmt(-500);// depositAmt method is called  
  
        c.withdrawAmt(1200);// withdrawAmt method is called  
  
        c.depositAmt(1500);  
  
        c.withdrawAmt(1000);  
  
    }  
  
}
```

**Output :**



```
Output  
Debugger Console x Tw8p (run) x  
run:  
Invalid amount  
Can't deposit as amount is negative  
  
Insufficient Balance  
Amount can't be withdrawm as there is insufficient Balance  
  
Balance in the account is 2500.0  
Balance in the account is 1500.0  
BUILD SUCCESSFUL (total time: 0 seconds)  
tw8 (run) running... 11:29 INS
```