Date 7/10/2023 **Experiment no 6**

**Exception Handling**

1. Create a class Doctor with attributes id, name, age and department. Initialize values through parameterized constructor. If age of Doctor is not in between 25 and 65 then generate user-defined exception “AgeNotWithinRangeException”. 2) Write a java program to handle the following Exceptions a) ArithmeticException b) NullPointerException c)NumberFormatException d) ArrayIndexOutOfBoundsException.

**Aim:**

To develop a java program for the given problem statement by using exception handling.

**Algorithm:**

Step1:Create a class called AgeNotWithinRangeException that extends Exception.

Step2:Create a constructor for AgeNotWithinRangeException that takes a message as a parameter.

Step3:Create a class called Doctor with private fields for id, name, age, and department.

Step4:Create a constructor for Doctor that takes id, name, age, and department as parameters.

Step5:Check if the age parameter is less than 25 or greater than 65.

Step6:If the age parameter is not within the valid range, throw an AgeNotWithinRangeException with a suitable message.

Step7:Assign the values of the id, name, age, and department parameters to the corresponding private fields in the Doctor object.

Step8:Create a method called displayDoctorInfo() in the Doctor class that prints the doctor's information to the console.

Step9:Create a public class called Main with a main() method.

Step10:In the main() method, try to create a new Doctor object with the values 1, Dr.Gopinath, 26, and Neurologist as parameters.

Step11:If an AgeNotWithinRangeException is thrown, catch it and print the exception message to the console.

**Program:**

class AgeNotWithinRangeException extends Exception {

public AgeNotWithinRangeException(String message) {

super(message);

}

}

class Doctor {

private int id;

private String name;

private int age;

private String department;

public Doctor(int id, String name, int age, String department) throws AgeNotWithinRangeException {

if (age < 25 && age > 65) {

throw new AgeNotWithinRangeException("Age is not within the valid range (25-65).");

}

this.id = id;

this.name = name;

this.age = age;

this.department = department;

}

public void displayDoctorInfo() {

System.out.println("Doctor ID: " + id);

System.out.println("Doctor Name: " + name);

System.out.println("Doctor Age: " + age);

System.out.println("Doctor Department: " + department);

}

}

public class Main {

public static void main(String[] args) {

try {

Doctor doctor = new Doctor(1, "Dr.Gopinath", 26, " Neurologist");

doctor.displayDoctorInfo();

} catch (AgeNotWithinRangeException e) {

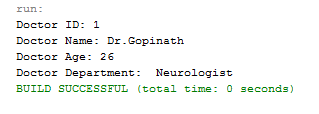
System.out.println("Exception: " + e.getMessage());

}

}

}

**Output:**



**Result:**

The given java program is executed successfully and the output was verified.

Date:7/10/2023 **2) Built in exceptions**

**Aim:**

To develop a java program to execute all the built in exceptions.

**Algorithm:**

**Step 1: Start**

**Step 2: Identify the operation that is likely to cause an exception**

**Step 3: Place the operation inside a** try **block**

**Step 4: Catch the specific exception that is likely to occur**

**Step 5: Print the exception message to the console**

**Step 6: Take appropriate action to recover from the exception**

**Step 7: Repeat steps 2-6 for each operation that is likely to cause an exception**

**Step 8: Stop**

**2(a) Arithmetic exception**

package arithmeticexception;

public class Arithmeticexception {

public static void main(String[] args) {

int i=0;

int j=0;

try{

j=100/i;

System.out.println("Something went wrong da boy");

}

catch(ArithmeticException e){

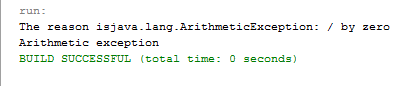
System.out.println("The reason is"+e);

System.out.println("Arithmetic exception");

}

}

}



**2(b) Null pointer exception**

package nullpointerexception;

public class Nullpointerexception {

public static void main(String[] args) {

try

{

String s = null;

System.out.println(s.length());

}

catch(NullPointerException e){

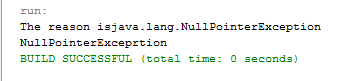
System.out.println("The reason is"+e);

System.out.println("NullPointerExceprtion");

}

}

}



**2(c) Number format exception**

class NumberFormat {

public static void main(String args[])

{

try {

// "akki" is not a number

int num = Integer.parseInt("akki");

System.out.println(num);

}

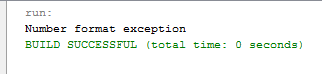
catch (NumberFormatException e) {

System.out.println("Number format exception");

}

}

}



**2(d) array index out of bounds**

package arrayindexoutofboundexception;

public class Arrayindexoutofboundexception {

public static void main(String[] args) {

int a[] = new int[5];

try{

System.out.println(a[2000]);

}

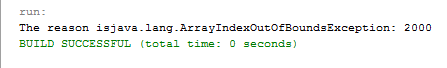
catch(ArrayIndexOutOfBoundsException e){

System.out.println("The reason is"+e);

}

}

}



**Result:**

All the built in exception was successfully executed and the output was verified.

|  |  |
| --- | --- |
| Observation |  |
| Record |  |
| Total |  |

Date 7/10/2023 **Exercise No 7**

**String Handling**

**Team 1**

Write an application that prompts the user for a password that contains at least two uppercase

letters, at least two lowercase letters, and at least two digits. Continuously reprompt the user until

a valid password is entered. After each entry, display a message indicating whether the user was

successful or the reason the user was not successful. Save the file as ValidatePassword.java.

**AIM:**

To develop a java program for the given problem statement using string handling.

**Algorithm:**

Step 1:Prompt the user to input a password.

Step 2 :Check if the password is null or empty.

Step 3:Check if the password length is less than 8 characters.

Step 4: Check if the password contains at least 2 uppercase letters.

Step 5:Check if the password contains at least 2 lowercase letters.

Step 6:Check if the password contains at least 2 digits.

Step 7:Check if the password contains at least 1 special character.

Step 8:Check if the password contains any common passwords from a known list of weak passwords.

Step 9:Check if the password is the same as the username.

Step 10:If the password passes all of the above checks, then it is valid. Otherwise, it is invalid.

**PROGRAM:**

package validatepassword.java;

import java.util.\*;

public class ValidatePasswordJava {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

boolean validPassword = false;

while (!validPassword) {

System.out.println("Enter a password: ");

String password = in.nextLine();

if (isValidPassword(password)) {

System.out.println("Your password is valid. You can proceed.");

validPassword = true;

} else {

System.out.println("Password is not valid. Please follow the requirements:");

System.out.println("- At least two uppercase letters");

System.out.println("- At least two lowercase letters");

System.out.println("- At least two digits");

}

}

}

public static boolean isValidPassword(String password) {

int upperCount = 0;

int lowerCount = 0;

int digitCount = 0;

for (int i = 0; i < password.length(); i++) {

char c = password.charAt(i);

if (Character.isUpperCase(c)) {

upperCount++;

} else if (Character.isLowerCase(c)) {

lowerCount++;

} else if (Character.isDigit(c)) {

digitCount++;

}

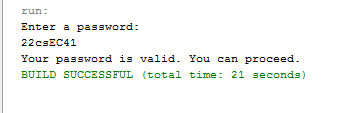
}

return upperCount >= 2 && lowerCount >= 2 && digitCount >= 2;

}

}

**OUTPUT:**



**RESULT:**

|  |  |
| --- | --- |
| Observation |  |
| Record |  |
| Total |  |

The given java program is executed successfully and the output was verified.