MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

Understanding and Implementation of Exception Handling Concepts in C++ and Java.

Practical Report

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 20

October 6, 2022

Contents

1 Aim and Objectives

Implementation and Understanding of Exception handling in Java and C++, and to learn and use the exception handling mechanisms with try and catch blocks.

2 Problem Statements

2.1 Problem 1 in C++

Define a class Employee consisting following:
Data Members

1. Employee ID
2. Name of Employee
3. Age
4. Income
5. City
6. Vehicle

Member Functions

1. To assign initial values.
2. To display.

Accept Employee ID, Name, Age, Income, City and Vehicle from the user. Create an exception to check the following conditions and throw an exception if the condition does not meet.

- Employee age between 18 and 55
- Employee income between Rs. 50,000 - Rs. 1,00,000 per month
- Employee staying in Pune/ Mumbai/ Bangalore / Chennai
- Employee having 4-wheeler

2.2 Problem 2 in Java

Implement the program to handle the arithmetic exception, ArrayIndexOutofBounds . The user enters the two numbers: n1, n2. The division of n1 and n2 is displayed. If n1, n2 are not integers then program will throw number format exception. If n2 is zero the program will throw Arithmetic exception.

2.3 Problem 3 in Java

Validate the employee record with custom exception Create a class employee with attributes eid, name, age and department. Initialize values through parameterized constructor. If age of employee is not in between 25 and 60 then generate user-defined exception "AgeNotWithinRangeException". If name contains numbers or special symbols raise exception "NameNotValidException". Define the two exception classes.

2.4 Problem 4 in Java

Write a menu-driven program for banking system which accept the personal data for Customer(cid, cname, amount). Implement the user-defined/standard exceptions, wherever required to handle the following situations:

1. Account should be created with minimum amount of 1000 Rs.
2. For withdrawal of amount, if withdrawal Amount is greater than the Amount in the Account.
3. Customer Id should be between 1 and 20 only.
4. Entered amount should be positive.

3 Theory

3.1 Concept of Compile time Polymorphism

3.2 Concept of Run Time Polymorphism

3.3 Use of Pure Virtual Functions

4 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : g++ and gcc on linux for C++, and javac, with JDK 18.0.2 for Java

5 Input

For C++

1. Number of Each Type of Employee
2. Name, Age, Address City, and Salary of Each Employee

For Java

1. The Side of the Square
2. The Radius of the Circle
3. The Length and Breadth of the Rectangle.

6 Output

For C++

1. General Information about Each Employee
2. The Weekly, hourly and commisioned Salary for Respective Employees.

For Java

1. The Area of the Shapes
2. The Location of the Hill Stations
3. The Reason the Hill stations are Famous for.

7 Code

7.1 C++ Implementation of Problem A

```cpp
#include <iostream>
#include <string.h>
using namespace std;

class Employee
{
public:
    int emp_id, age, income;
    string name, city;
    bool has_vehicle, data_entered_correctly;

    Employee()
    {
        emp_id = 1;
        age = 30;
        income = 10000;
        name = "William";
        city = "Pune";
        has_vehicle = true;
        cout << "The default values are: " << endl;
        data_entered_correctly = true;
        display();
    }

    int accept()
    {

        cout << "Enter the information of the new Employee : " << endl;
        cout << "Enter the Employee ID: " << endl;
        cin >> emp_id;
        cout << "Enter the Employee Name: " << endl;
        cin >> name;
    age:
        cout << "Enter the Employee Age: " << endl;
        try
        {
```

```
37              cin >> age;
38              if (age <= 18 || age >= 55)
39              {
40                  throw age;
41              }
42          }
43          catch (int e)
44          {
45              cout << "Exception Caught!, Age is not in the valid limit. " << endl;
46              data_entered_correctly = false;
47              goto age;
48          }
49      salary:
50          cout << "Enter the Employee basic Salary: " << endl;
51          try
52          {
53              cin >> income;
54              if (income <= 50000 || income >= 100000)
55              {
56                  throw 'a';
57              }
58          }
59          catch (char e)
60          {
61              cout << "Exception Caught!, Income is not in the valid limit. " <<
    endl;
62              data_entered_correctly = false;
63              goto salary;
64          }
65      city:
66          cout << "Enter the Employee Address City: " << endl;
67          try
68          {
69              cin >> city;
70              if (city != "Mumbai" && city != "Pune" && city != "Bangalore")
71              {
72                  throw 1.1;
73              }
74          }
75          catch (double e)
76          {
77              cout << "Exception Caught!, City Entered Incorrectly " << endl;
78              data_entered_correctly = false;
79              goto city;
80          }
81      vehicle:
82          cout << "Does the Employee have a vehicle? (Y, N) " << endl;
83          try
84          {
85              char inp;
86              cin >> inp;
87              if (inp != 'Y')
88              {
89                  throw has_vehicle;
90              }
91          }
92          catch (bool e)
93          {
94              cout << "Exception Caught!, You must have a vehicle!" << endl;
```

```
            data_entered_correctly = false;
            goto vehicle;
        }
        data_entered_correctly = true;
        return 0;
    }

    void display()
    {
        if (data_entered_correctly)
        {
            cout << "Employee ID is : " << emp_id << endl;
            cout << "Employee Name: " << name << endl;
            cout << "Employee Age: " << age << endl;
            cout << "Employee Income : " << income << endl;
            cout << "Employee Address City: " << city << endl;
            cout << "Does Employee have a vehicle? : " << has_vehicle << endl;
        }
        else
        {
            cout << "You didnt enter the Data Correctly" << endl;
        }
    }
};

int main()
{
    cout << "Welcome to Assignment 4: Error Safe Employee Data Input Program" <<
    endl;
    Employee obj;
    obj.accept();
    obj.display();
    if (!obj.data_entered_correctly)
    {
        cout << "Please Try again";
    }
    else
    {
        cout << "You have entered the data correctly! " << endl;
    }
    return 0;
}
```

Listing 1: Main.Cpp

7.1.1 C++ Output

```
Welcome to Assignment 4: Error Safe Employee Data Input Program
The default values are:
Employee ID is : 1
Employee Name: William
Employee Age: 30
Employee Income : 10000
Employee Address City: Pune
Does Employee have a vehicle? : 1
Enter the information of the new Employee :
Enter the Employee ID:
1
Enter the Employee Name:
```

```
Mike
Enter the Employee Age:
400
Exception Caught!, Age is not in the valid limit.
Enter the Employee Age:
2000
Exception Caught!, Age is not in the valid limit.
Enter the Employee Age:
40
Enter the Employee basic Salary:
1
Exception Caught!, Income is not in the valid limit.
Enter the Employee basic Salary:
5000
Exception Caught!, Income is not in the valid limit.
Enter the Employee basic Salary:
100000
Exception Caught!, Income is not in the valid limit.
Enter the Employee basic Salary:
60000
Enter the Employee Address City:
Mumbai
Does the Employee have a vehicle? (Y, N)
n
Exception Caught!, You must have a vehicle!
Does the Employee have a vehicle? (Y, N)
Y
Employee ID is : 1
Employee Name: Mike
Employee Age: 40
Employee Income : 60000
Employee Address City: Mumbai
Does Employee have a vehicle? : 1
You have entered the data correctly!
```

Listing 2: Output for Problem 1

7.2 Java Implementation of Problem B

```java
package assignment_4;

import java.util.InputMismatchException;
import java.util.Scanner;

// Implement the program to handle the arithmetic exception, ArrayIndexOutofBounds.
// The user enters the two numbers: n1, n2. The division of n1 and n2 is displayed. If n1, n2
// are not integers then program will throw number format exception. If n2 is zero the
// program will throw Arithmetic exception.

// class NumberFormatException extends Exception
// {
//     public NumberFormatException(String s)
//     {
//         super(s);
//     }

```

```
//      @Override
//      public String getMessage()
//      {
//          return "What is this";
//      }
// }

public class Division {
    int n1;
    int n2;
    int ans;
    Scanner input = new Scanner(System.in);

    void accept() {
        System.out.println("Enter the Numbers that you want to divide. ");
        try {
            n1 = input.nextInt();
            n2 = input.nextInt();
        } catch (InputMismatchException e) {
            System.out.println("Number Format Exception, the format you entered
    does not match. ");
        }
    }

    int divide() {
        try {
            ans = n1 / n2;
        } catch (ArithmeticException e) {
            System.out.println("Exception Caught! Cant divide by Zero!");
        }
        return ans;
    }

}
```

Listing 3: Full Time Employee.java

7.2.1 Java Output

```

Enter the Employee ID:
1001
Enter the Employee Name
William
Enter the Employee Age:
1
Age not within range
Age is not within the correct range.


Enter the Employee ID:
1
Enter the Employee Name
William
Enter the Employee Age:
35
Enter the Employee Department:
Sales
Employee ID is : 1
```

```
Employee Name: William
Employee Age: 35
Employee Department: Sales
```

Listing 4: Output for Problem 2

7.3 Java Implementation of Problem C

```java
package assignment_4;

import java.util.Scanner;

// Validate the employee record with custom exception
// Create a class employee with attributes eid, name, age and department.
// Initialize values through parameterized constructor. If age of employee is not in between
// 25 and 60 then generate user-defined exception "AgeNotWithinRangeException". If
// name contains numbers or special symbols raise exception "NameNotValidException".
// Define the two exception classes.

class AgeNotWithinRangeException extends Exception {
    public AgeNotWithinRangeException(int s) {
        // super(s);
        System.out.println("Age not within range");
    }

    @Override
    public String getMessage() {
        return "Incorrect Age!";
    }
}

class NameNotValidException extends Exception {
    public NameNotValidException(String s) {
        // super(s);
    }
}

public class Employee {
    Scanner input = new Scanner(System.in);
    int e_id, age;
    String name, department;

    int accept() {
        String specialCharactersString = "1234567890!@#$%&*()'+,-./:;<=>?[]^_`{|}";
        try {
            System.out.println("Enter the Employee ID: ");
            e_id = input.nextInt();
            System.out.println("Enter the Employee Name");
            name = input.next();

            for (int i = 0; i < name.length(); i++) {
                char ch = name.charAt(i);
                if (specialCharactersString.contains(Character.toString(ch))) {
                    System.out.println(name + " contains special character");
                    throw new NameNotValidException(name);
                }
```

```
            }
            System.out.println("Enter the Employee Age: ");
            age = input.nextInt();
            if (age > 60 || age < 25) {
                throw new AgeNotWithinRangeException(age);
            }
            System.out.println("Enter the Employee Department: ");
            department = input.next();

        } catch (AgeNotWithinRangeException e) {
            System.out.println("Age is not within the correct range.");
            return -1;
        } catch (NameNotValidException e) {
            System.out.println("Name not Valid");
            return -1;
        }
        return 0;
    }

    void display() {
        System.out.println("Employee ID is : " + e_id);
        System.out.println("Employee Name: " + name);
        System.out.println("Employee Age: " + age);
        System.out.println("Employee Department: " + department);
    }

}
```

Listing 5: HillStation

7.3.1 Java Output

```

Enter the Employee ID:
1001
Enter the Employee Name
William
Enter the Employee Age:
1
Age not within range
Age is not within the correct range.


Enter the Employee ID:
1
Enter the Employee Name
William
Enter the Employee Age:
35
Enter the Employee Department:
Sales
Employee ID is : 1
Employee Name: William
Employee Age: 35
Employee Department: Sales
```

Listing 6: Output for Problem 3

7.4 Java Implementation of Problem D

```java
package assignment_4;

import java.util.*;

public class Bank {
    public int minimum_sal;
    public int withdrawal_amt;
    public int amount;
    public int c_id;
    Scanner input = new Scanner(System.in);

    public int accept() {
        System.out.println("Enter the customer id: ");
        try {
            c_id = input.nextInt();
            if (c_id > 20 || c_id < 0) {
                throw new Exception("wrong cust id");
            }
        } catch (Exception e) {
            System.out.println("Wrong customer id");
            return 0;
        }

        System.out.println("Enter the Amount in your Account: ");
        try {
            amount = input.nextInt();
            if (amount < 1000) {
                throw new Exception("Amount less than mimimum");
            }
        } catch (Exception e) {
            System.out.println("Mimimum amount cant be less than 1000. ");
            return 0;
        }

        System.out.println("Enter the Withdrawal Amount: ");
        try {
            withdrawal_amt = input.nextInt();
            if (withdrawal_amt > amount) {
                throw new Exception("Withdrawal amount more than amount. ");
            } else {
                amount -= withdrawal_amt;
            }
        } catch (Exception e) {
            System.out.println("Withdrawal Amount more than amount in bank. ");
            return 0;
        }

        return 1;
    }

    public void display() {
        System.out.println("\n\nThe Customer ID is: ");
        System.out.println(c_id);
        System.out.println("The Amount in the Bank Before Withdrawing was: ");
        System.out.println(amount + withdrawal_amt);
        System.out.println("The Withdrawal Amount is: ");
        System.out.println(withdrawal_amt);
        System.out.println("The Amount Remaining in Bank is: ");
        System.out.println(amount);
```

```
    }
}
```

Listing 7: HillStation

```
package assignment_4;

public class Main {

    public static void program_3() {
        Bank obj = new Bank();
        if (obj.accept() == 1) {
            System.out.println("Data entered Correctly!");
            obj.display();
        } else {
            System.out.println("Data entered Incorrectly!");
        }
    }

    public static void program_2() {
        Division d = new Division();
        d.accept();
        System.out.println("Dividing the inputs: ");
        System.out.println(d.divide());

    }

    public static void program_1() {

        Employee obj = new Employee();
        if (obj.accept() >= 0) {
            obj.display();
        }
    }

    public static void main(String args[]) {

        // program_1();
        program_2();
        program_3();

    }
}
```

Listing 8: HillStation

7.4.1 Java Output

```
Enter the customer id:
1001
Wrong customer id
Data entered Incorrectly!

Enter the customer id:
1
Enter the Amount in your Account:
5000
Enter the Withdrawal Amount:
3000
```

```
Data entered Correctly!


The Customer ID is:
1
The Amount in the Bank Before Withdrawing was:
5000
The Withdrawal Amount is:
3000
The Amount Remaining in Bank is:
2000
 krishnaraj@
```

Listing 9: Main.java

8 Conclusion

Thus, learned to use polymorphism and implemented solution of the given problem statement using C++ and Java.

9 FAQs

1. **Why do we use Exception Handling mechanism?**
2. **Is it possible to use multiple catch for single throw? Explain?**
3. **What is Exception Specification?**
4. **What is Re-throwing Exception?**
5. **Explain use of finally keyword in java.**