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 9, 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 Exception Handling

Exception handling is the process of responding to unwanted or unexpected events when a computer program runs. Exception handling deals with these events to avoid the program or system crashing, and without this process, exceptions would disrupt the normal operation of a program.

Exceptions occur for numerous reasons, including invalid user input, code errors, device failure, the loss of a network connection, insufficient memory to run an application, a memory conflict with another program, a program attempting to divide by zero or a user attempting to open files that are unavailable.

When an exception occurs, specialized programming language constructs, interrupt hardware mechanisms or operating system interprocess communication facilities handle the exception.

3.2 Try Throw Catch Block

Exception handling in C++ revolves around these three keywords:
throw- when a program encounters a problem, it throws an exception. The throw keyword helps the program perform the throw.

catch- a program uses an exception handler to catch an exception. It is added to the section of a program where you need to handle the problem. It's done using the catch keyword.

try- the try block identifies the code block for which certain exceptions will be activated. It should be followed by one/more catch blocks. Suppose a code block will raise an exception. The exception

will be caught by a method using try and catch keywords. The try/catch block should surround code that may throw an exception. Such code is known as protected code.

```
  try {
  // the protected code
  } catch( Exception_Name exception1 ) {
  // catch block
  } catch( Exception_Name exception2 ) {
  // catch block
  } catch( Exception_Name exceptionN ) {
  // catch block
  }
```

```
try {
  //  Block of code to try
}
catch(Exception e) {
  //  Block of code to handle errors
}
```

3.3 Catch All

Catch block is used to catch all types of exception. The keyword “catch” is used to catch exceptions. If used like this, it would catch all the exceptions in the try block.

```
  #include <iostream>
  using namespace std;

  void func(int a) {
  try {
    if(a==0) throw 23.33;
    if(a==1) throw 's';
  } catch(...) {
    cout << "Caught Exception!\n";
  }
  }
  int main() {
    func(0);
    func(1);
    return 0;
  }

```

In Java

```
  try {
    //  Block of code to try
  }
  catch(Exception e) {
    //  Block of code to handle errors
  }
```

3.4 Rethrowing Exceptions

If a catch block cannot handle the exception that it was designed to handle, then you can rethrow that exception from that catch block. It causes the original exception to be rethrown.

Because the exception has already been caught at the scope in which the rethrow expression occurs, it is rethrown out to the next dynamically enclosing try block. Therefore, it cannot be handled

by catch blocks at the scope in which the rethrow expression occurred. Any catch blocks for the dynamically enclosing try block have an opportunity to catch the exception.

```
void f() {
  try {
  cout << "In try block of f()" << endl;
  cout << "Throwing exception of type E1" << endl;
  E1 myException;
  throw myException;
  }
  catch (E2& e) {
  cout << "In handler of f(), catch (E2& e)" << endl;
  cout << "Exception: " << e.message << endl;
  throw;
  }
  catch (E1& e) {
  cout << "In handler of f(), catch (E1& e)" << endl;
  cout << "Exception: " << e.message << endl;
  throw;
  }
}
```

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

```
1
2  // Define a class Employee consisting following:
3  // Data members: a. Employee ID, b. Name of Employee, c. Age, d. Income, e. City,
       f. Vehicle
4  // Member Functions: a. To assign initial values, b. To display.
5
6  // Accept Employee ID, Name, Age, Income, City and Vehicle from the user. Create
7  // an exception to check the following conditions and throw an exception if the
8  // condition does not meet.
9
10 // Employee age between 18 and 55
11 // Employee income between Rs. 50,000 to Rs. 1,00,000 per month
12 // Employee staying in Pune/ Mumbai/ Bangalore / Chennai
13 // Employee having 4-wheeler
14
15 #include <iostream>
16 #include <string.h>
17 using namespace std;
18
19 class Employee
20 {
21 public:
22     int emp_id, age, income;
23     string name, city;
24     bool has_vehicle, data_entered_correctly;
25
26     Employee()
27     {
28         emp_id = 1;
29         age = 30;
30         income = 10000;
31         name = "William";
32         city = "Pune";
33         has_vehicle = true;
34         cout << "The default values are: " << endl;
35         data_entered_correctly = true;
36         display();
37     }
38
39     int accept()
40     {
41
42         cout << "Enter the information of the new Employee : " << endl;
43         cout << "Enter the Employee ID: " << endl;
44         cin >> emp_id;
```

```
45          cout << "Enter the Employee Name: " << endl;
46          cin >> name;
47      age:
48          cout << "Enter the Employee Age: " << endl;
49          try
50          {
51              cin >> age;
52              if (age <= 18 || age >= 55)
53              {
54                  throw age;
55              }
56          }
57          catch (int e)
58          {
59              cout << "Exception Caught!, Age is not in the valid limit. " << endl;
60              data_entered_correctly = false;
61              goto age;
62          }
63      salary:
64          cout << "Enter the Employee basic Salary: " << endl;
65          try
66          {
67              cin >> income;
68              if (income <= 50000 || income >= 100000)
69              {
70                  throw 'a';
71              }
72          }
73          catch (char e)
74          {
75              cout << "Exception Caught!, Income is not in the valid limit. " <<
    endl;
76              data_entered_correctly = false;
77              goto salary;
78          }
79      city:
80          cout << "Enter the Employee Address City: " << endl;
81          try
82          {
83              cin >> city;
84              if (city != "Mumbai" && city != "Pune" && city != "Bangalore")
85              {
86                  throw 1.1;
87              }
88          }
89          catch (double e)
90          {
91              cout << "Exception Caught!, City Entered Incorrectly " << endl;
92              data_entered_correctly = false;
93              goto city;
94          }
95      vehicle:
96          cout << "Does the Employee have a vehicle? (Y, N) " << endl;
97          try
98          {
99              char inp;
100             cin >> inp;
101             if (inp != 'Y')
102             {
```

```
                throw has_vehicle;
            }
        }
        catch (bool e)
        {
            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

```
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

```
// P1
// 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.

// P2
// 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.

// P3
// 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:
// Account should be created with minimum amount of 1000 rs..
// For withdrawal of amount, if wth_amt greater than amount.
// cid should be in the specific range of 1 to 20.
// Entered amount should be positive.

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?**
   Exception handling is the process of responding to unwanted or unexpected events when a computer program runs. Exception handling deals with these events to avoid the program or system crashing, and without this process, exceptions would disrupt the normal operation of a program.

   (a) It helps you avoid the program from crashing

   (b) It helps you set the program control in a more detailed manner

   (c) It helps you manually stop the program safely according to your wish.

```
// Psudo code for exception handling
int a;
try
{
cin>> a;
cout<<33/a; // if a is 0, Exception is thrown, program crashes.
}
catch DivisionByZeroException as e
{
cout<<"cant divide by zero;
}
```

2. **Is it possible to use multiple catch for single throw? Explain?**
   Yes, it is possible to use multiple catch statements for a single throw statement in both C++ and Java. C++ try and catch block works in a specific way, where the throw keyword throws an exception with an integer, or an exception. You can then write multiple catch statements just below the try block.
   In Java, you can throw instances of the Exception class, or throw any of the pre defined exceptions in java.

   Example

```
    int a;
    try
    {
      cin>> a;
      if(a == 0)
      {
        throw a; // int is being thrown.
      }
      else{
        cout<<33/a;
        throw 'a'; // character is being thrown.
      }
    }
    catch (int a)
    {
      cout<<"cant divide by zero;
    }
    catch (char a)
```

```
    {
      cout<<"A is not zero";
    }

```

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

```

3. **What is Exception Specification?**
   An exception specification is a contract between the function and the rest of the program. It is a guarantee that the function will not throw any exception not listed in its exception specification.

```
    void f1(void) throw(int) {
      printf_s("About to throw 1\n");
      if (1)
         throw 1;
     }

```

```
    void function_name() throw(Exception)
    {
      if (error)
      {
      throw Exception("Error");
      }
    }

```

4. **What is Re-throwing Exception?**
   If a catch block cannot handle the exception that it was designed to handle, then you can rethrow that exception from that catch block. It causes the original exception to be rethrown.

   Because the exception has already been caught at the scope in which the rethrow expression occurs, it is rethrown out to the next dynamically enclosing try block. Therefore, it cannot be handled by catch blocks at the scope in which the rethrow expression occurred. Any catch blocks for the dynamically enclosing try block have an opportunity to catch the exception.

```
    void f() {
      try {
        cout << "In try block of f()" << endl;
        cout << "Throwing exception of type E1" << endl;
```

```
        E1 myException;
        throw myException;
      }
      catch (E2& e) {
        cout << "In handler of f(), catch (E2& e)" << endl;
        cout << "Exception: " << e.message << endl;
        throw;
      }
      catch (E1& e) {
        cout << "In handler of f(), catch (E1& e)" << endl;
        cout << "Exception: " << e.message << endl;
        throw;
      }
    }


```

5. **Explain use of finally keyword in java.**
   Java finally block is a block used to execute important code such as closing the connection, etc.

   Java finally block is always executed whether an exception is handled or not. Therefore, it contains all the necessary statements that need to be printed regardless of the exception occurs or not.

   The finally block follows the try-catch block.

```
class TestFinallyBlock {
  public static void main(String args[]){
  try{
  //below code do not throw any exception
   int data=25/5;
   System.out.println(data);
  }
  //catch won't be executed
  catch(NullPointerException e){
  System.out.println(e);
  }
  //executed regardless of exception occurred or not
   finally {
  System.out.println("finally block is always executed");
  }
```