**Assignment number: 7**

**Subject: Object Oriented Programming**

**Exception Handling**

Name: ***RIA MITTAL***

Class: ***SECOND YEAR ENGINEERING***

Division: ***B***

Roll no: ***222008***

Batch: ***B1***

**Problem Statement:**

Create User defined exception to check the following conditions and throw the exception if the criterion does not meet. a. User has age between 18 and 55 b. User stays has income between Rs. 50,000 – Rs. 1,00,000 per month c.

User stays in Pune/ Mumbai/ Bangalore / Chennai d. User has 4-wheeler Accept age, Income, City, Vehicle from the user and check for the conditions mentioned above.

If any of the condition not met then throw the exception.

# Objective :

## **Assume appropriate data members and member function to accept required data & print the output.**

# CONCEPT:

## Exception Handling

# THEORY:

Exception handling is the process of responding to the occurrence, during computation, of exceptions – anomalous or exceptional conditions requiring special processing – often changing the normal flow of [program](https://en.wikipedia.org/wiki/Computer_program) [execution](https://en.wikipedia.org/wiki/Execution_(computing)).

It is provided by specialized [programming language](https://en.wikipedia.org/wiki/Programming_language) constructs or [computer hardware](https://en.wikipedia.org/wiki/Computer_hardware) mechanisms.

In general, an exception is handled (resolved) by saving the current state of execution in a predefined place and switching the execution to a specific [subroutine](https://en.wikipedia.org/wiki/Subroutine) known as an exception handler.

If exceptions are continuable, the handler may later resume the execution at the original location using the saved information.

For example, a [floating point](https://en.wikipedia.org/wiki/Floating_point) [divide by zero](https://en.wikipedia.org/wiki/Division_by_zero) exception will typically, by default, allow the program to be resumed, while an [out of memory](https://en.wikipedia.org/wiki/Out_of_memory) condition might not be resolvable transparently.

Alternative approaches to exception handling in software are error checking, which maintains normal program flow with later explicit checks for contingencies reported using special return values or some auxiliary global variable such as C's [**errno**](https://en.wikipedia.org/wiki/Errno) or floating point status flags or input validation to pre-emptively filter exceptional cases.

Some programmers write software with error reporting features that collect details that may be helpful in fixing the problem, and display those details on the screen, or store them to a file such as a [core dump](https://en.wikipedia.org/wiki/Core_dump), or in some cases an [automatic error reporting](https://en.wikipedia.org/wiki/Motorola_Q#Error_reporting) system such as [Windows Error Reporting](https://en.wikipedia.org/wiki/Windows_Error_Reporting) can automatically [phone home](https://en.wikipedia.org/wiki/Phone_home) and email those details to the programmers.

# ALGORITHM:

## 1] Start.

## 2]Declare a Character for name and the city.

3]make a function try.

4]List out the statements which can create exeptions

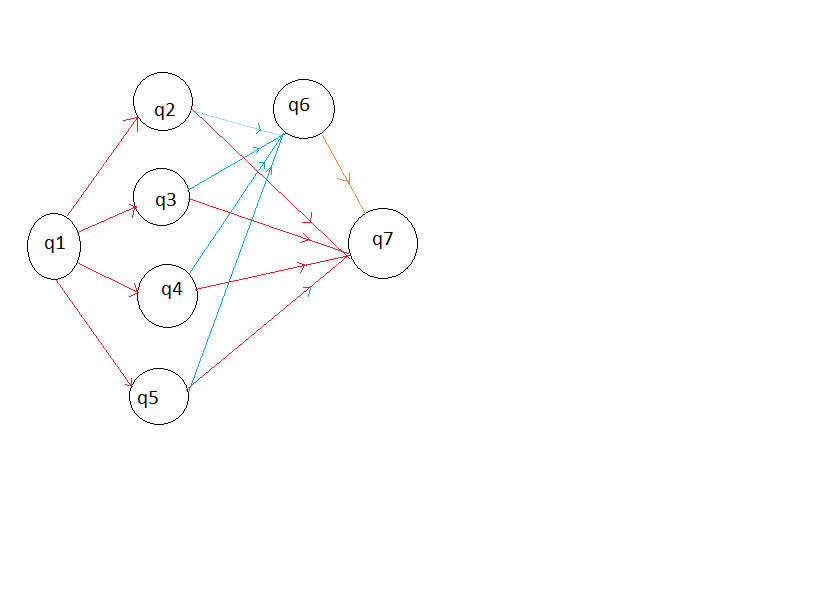
5]throw the exception causing variable

6]catch the exception causing variable using the catch block

7]print statements

## 8] Stop.

**STATE TRANSITION DIAG**



q0: Start program and initialize variables.

q1:call function age().

q2:call function salary().

q3:call function vehicle().

q4:call function city().

q5:Handle exception.

q6:End.

# PROGRAM:

#include<iostream>

using namespace std;

class except

{

public:

int age()

{

int age1;

cout<<"\nenter your age\n";

cin>>age1;

try

{

if(age1<18 || age1>55)

{

throw age1;

}

else

{

cout<<"\nyour age is :"<<age1<<endl;

}

}

catch(int)

{

cout<<"\nexception :age is: "<<age1<<endl;

}

}

double income1()

{

double income;

cout<<"\nenter your income:\n";

cin>>income;

try

{

if(income<50000 || income>100000)

throw income;

else

cout<<"\nyour income is :"<<income<<endl;

}

catch(double)

{

cout<<"\nexception: income is: "<<income<<endl;

}

}

void city1()

{

string city;

cout<<"\nenter the city:\n";

cin>>city;

try

{

if(city=="pune" || city=="chennai" || city=="bangalore" || city=="mumbai")

cout<<"\nyour city is : "<<city<<endl;

else

throw city;

}

catch(string)

{

cout<<"\nexception: your city is: "<<city<<endl;

}

}

int vehicle()

{

int num;

cout<<"\nenter the type of vehicle you have: 2 wheeler or 4 wheeler?\n";

cin>>num;

try

{

if(num==2)

throw num;

else if(num==4)

cout<<"\nyour vehicle is a "<<num<<" wheeler \n";

else

cout<<"\ninvalid choice....try again\n";

}

catch(int)

{

cout<<"\nexception : you have a 2 wheeler!!\n";

}

}

};

int main()

{

int ch;

char c;

except obj;

do

{

cout<<"\n1.age\n2.income\n3.city\n4.vehicle\n";

cin>>ch;

switch(ch)

{

case 1:obj.age();break;

case 2:obj.income1();break;

case 3:obj.city1();break;

case 4:obj.vehicle();break;

default:cout<<"\ninvalid choice...try again\n";

}

cout<<"\ndo you want to try again?(y/n)\n";

cin>>c;

}while(c=='y' || c=='Y');

return 0;

}

# OUTPUT:

1.age

2.income

3.city

4.vehicle

1

enter your age

19

your age is :19

do you want to try again?(y/n)

y

1.age

2.income

3.city

4.vehicle

1

enter your age

13

exception :age is: 13

do you want to try again?(y/n)

n

# CONCLUSION:

We’ve learnt how to implement exceptional handling .

We’ve also learnt about the throw string function and how they affect the data members during exception handling.