Devanshu Surana
PC-23, 1032210755
Batch - C1

# OOCCJ Lab Assignment 4A

Problem statement:
Develop an object-oriented program in C++ to create a class Hotel for hotel booking system. The class hotel has data members: string cust_name, int cust_id, int income, string city, and string room_type and has a default constructor. It has function accept() and display() to output data member values. The class Hotel has member functions as: getage(), getincome(), getcity(), get room_type().
Throw four exception as.
1. If the age is not between 18 and 55
2. If income is not between 50000 and 100000
3. If city is not Pune or Mumbai.
4. If room_type is not delux or super delux.
Use exception handling to check if above conditions are satisfied (display all customer information) else print error message. If the entered data is valid, store the data in the file. store such 5 records in the file.

Objectives:
1. To learn to create handling exceptions in C++.
2. To learn try-catch block in C++.
3. To learn throw exception in C++.
4. To learn file handling concepts.

Teacher's Sign.:

Theory:
Explain following concepts:
- Exception handling
- It indicate problems that occur during a progra
execution.
- It occurs in frequently
- It can resolve exceptions.
- Allow a program to continue executing or
- Notify the user of the problem and
- Terminate the program in a controlled ma
- It makes programs robust and fault tolerant.
- Remove error-handling code from the program ex
'main line'.

Try-Catch throw block.
→ try: keyword try followed by braces ({}) should
· statements that might cause exceptions.
· statements that should be skipped in case of
→.
→ catch: keyword catch
· Immediately follow a try block.
· One or more catch handlers for each try blo
· Exception parameter enclosed in parenthesis
→ throw: keyword throw followed by an operand
the type of exception.
· Can be of any type
· If throw operand is an object, then its called
object.
The throw operand initializes the exception pa
in the matching catch handler if one is

Teacher's Sign

- Standard Libr
- It is a base-
'what' for
- Exception class
1. bad-alloc
2. bad-cast
3. bad-type
4. bad-exce
- stream classe
- ofstream: stre
- ifstream: stre
- fstream: str
files.
- istream: Resp

Platform: 64

Input: Hotel

Output: Prop
invalid da

Conclusion:
Hence the
handling in

Standard Library Exceptions hierarchy.
- It is a base-class exception contains virtual function 'what' for storing error messages.
- Exception classes derived from exception
1. bad-alloc - thrown by new
2. bad-cast - thrown by dynamic-cast
3. bad-typeid - thrown by typeid
4. bad-exception - thrown by unexpected

- stream classes and functions.
- ofstream : stream class to write on files
- ifstream : Stream class to read from files.
- fstream : Stream class to both read and write from/to files.
- istream : Responsible for handling input stream.

Platform : 64-bit Open Source Linux

Input : Hotel-class object Data Members

Output : Properly displayed output or thrown exception if invalid data is Input.

Conclusion :
Hence the concepts of file handling and exception handling in C++ are studied successfully.

TP
27/4/23

Teacher's Sign :

FAQ's

1. What is exception handling?

→ Exception handling in c++ is a mechanism for dea
with errors that may occur during program execut
When an error occurs, the program can throw an e
which is a special type of object that contains in
-on about the error.

2. Explain Multiple Exceptions, re-throwing an exception.

→ To handle multiple exceptions, the catch block can
specify multiple exceptions types separated by the
operator (||) like this:

```
try {
    // some code that may throw exceptions
}
catch (const std::exception& ex) {
    // handle std::exception and its derived
}
catch (const std::string& str) {
    // handle std::string exceptions.
}
~~catch (const std::~~
catch (...) {
    // handle any other exceptions types.
}
```

This code will catch exceptions of type 'std
-eption' and its derived classes, executions of t
'std::string' and any other exception types tha
were caught by the previous catch blocks.

Teacher's Sign

Re-throwing ar
Sometimes it
so that it c
Example:

```
try {
    // some
}
catch (co
    // ha
    throw
}
```

3. Explain step
→ Steps in file
1. Include the
2. Declare file
3. Open the fi
4. Use the file
input/output
5. Close the

File Handlin
1. Creating a
2. Opening an
3. Reading d
4. Writing da
5. Moving dat
6. Closing th

Re-throwing an exception

Sometimes it is neccessary to re-throw an exception so that it can be handled further up the call stack.

Example:

```
try {
    // some code that may throw an exception
}
catch (const std::exception& ex) {
    // handle the exception
    throw; // re-throw the exception
}
```

Explain steps required in file handling.

Steps in file handling:

1. Include the header file fstream in the program.
2. Declare file stream object.
3. Open the file with the file stream object.
4. Use the file stream object with >>, << or other input/output functions.
5. Close the file.

File Handling:

- Creating a new file.
- Opening an existing file.
- Reading data from an existing file
- Writing data to a file
- Moving data to a specific location.
- Closing the file.

Page No.
Date

Algorithm:

Step 1: Define a class Hotel with private data me... string cust-name, int cust-id, int income, string city, ... string room-type. The class also has a default constru...

Step 2: Define member functions of class Hotel:
accept(): to accept the input values for data ...
display(): to display the values of data members.

Step 3: Define the following member functions of class Hotel: getage(), getincome(), getcity(), get room-type.

Step 4: Define 4 exception as follows:
1) Age not valid
2) Income not valid
3) City Not valid
4) Room type not vali...

Step 5: Use exception handling to check if above con... are satisfied or not. If ~~stat~~ satisfied then s... data in file.

Step 6: Store 5 records in the file.

Teacher's Sign...

Devanshu Sur...
PC-23, 10322...
Batch C1

OOCJ Lab

Aim: Write ... Exception ...

Part - 1 (f...
A) Read th... 5 subjects [... - Reader and... Writer and...

Objectives:
- To study ...
- To study ...

Theory:
Abstracti... and showing... shows only ... -al detai... for ex: ... the messa... it can b...

Java I/O ... output. Th... for input ...

**Name:** Devanshu Surana

**Roll No.:** 23

**Panel:** C

**Batch:** C1

## OOCCJ Lab Assignment 4A

**CODE:**

**FILE HANDLING:**

```
#include <iostream>
#include <fstream>
using namespace std;
class student{ private:
int rollno; string name; int id;
public:
student();
student(int rollno,string name,int id);
void display();
void get();
void clear();
};
student::student(){
rollno=0000;
name="UN_named";
id=9999;
}
student::student(int x,string y,int z){ rollno=x;
```

```
name=y;
id=z;
}
void student::display(){
cout<<"Roll Number is: "<<rollno<<endl; cout<<"Name is:
"<<name<<endl; cout<<"ID Is: "<<id<<endl;
}
void student::get(){
int a,b;
string c;
cout<<"Your Roll Number is: "<<endl; cin>>a;
rollno=a;

cout<<"Your Name Is: "<<endl; cin>>c;
name=c;
cout<<"Your ID Is:"<<endl; cin>>b;
id=b;
}
void student :: clear(){
rollno=-1;
name="NULL";
id=9999;
}
int main() {
ifstream fin;
ofstream fout;
```

student s1; fout.open("text.txt",ios::out); s1.get();

fout.write((char *)&s1,sizeof(s1)); fout.close(); fin.open("text.txt",ios::in); s1.clear();

s1.display();

fin.read((char *)&s1,sizeof(s1)); s1.display();

return 0;

}

**OUTPUT:-**

```
Your Roll Number is:
23
Your Name Is:
Devanshu Surana
Your ID Is:
Roll Number is: -1
Name is: NULL
ID Is: 9999
Roll Number is: -1
Name is: NULL
ID Is: 9999
```

**EXCEPTION HANLDING:**

#include<iostream>

#include<fstream>

using namespace std;

class hotel

```
{
    string cust_name;
    int cust_id;
    int income;
    string city;
    string room_type;
    int age;

    public:
        hotel();
        void accept();
        void display();
        void getage();
        void getincome();
        void getcity();
        void getroom_type();
};

hotel::hotel()
{
    cust_name="";
    cust_id=0;
    income=0;
    city="";
    room_type="";
    age=0;
```

```
}

void hotel::accept()
{
    cout <<"\n Enter customer name: ";
    cin >> cust_name;
    cout <<"\n Enter customer id: ";
    cin >> cust_id;
    getage();
    getincome();
    getcity();
    getroom_type();
}

void hotel::getage()
{
    try
    {
        int a;
        cout<<"Enter your age: ";
        cin>>a;
        if (a<18 || a>55)
        throw(a);
        age=a;
    }
    catch(int a1)
```

```
    {
        cout<<"Age should be between 18 and 55!!"<<endl;
    }
}

void hotel::getincome()
{
    try
    {
        int b;
        cout<<"Enter your income: ";
        cin>>b;
        if (b<50000 || b>100000)
        throw(b);
        income=b;
    }
    catch (int b1)
    {
        cout<<"Income should be between 50k and 1L!!"<<endl;
    }
}

void hotel::getcity()
{
    try
    {
```

```
        string c;
        cout<<"Enter city: ";
        cin>>c;
        if (c!="pune" && c!="mumbai")
        throw(c);
        city=c;
    }
    catch(string c1)
    {
        cout<<"City must be pune or mumbai!!"<<endl;
    }
}

void hotel::getroom_type()
{
    try
    {
        string d;
        cout<<"Enter room type: ";
        cin>>d;
        if (d!="delux" && d!="super delux")
        throw(d);
        room_type=d;
    }
    catch(string d1)
    {
```

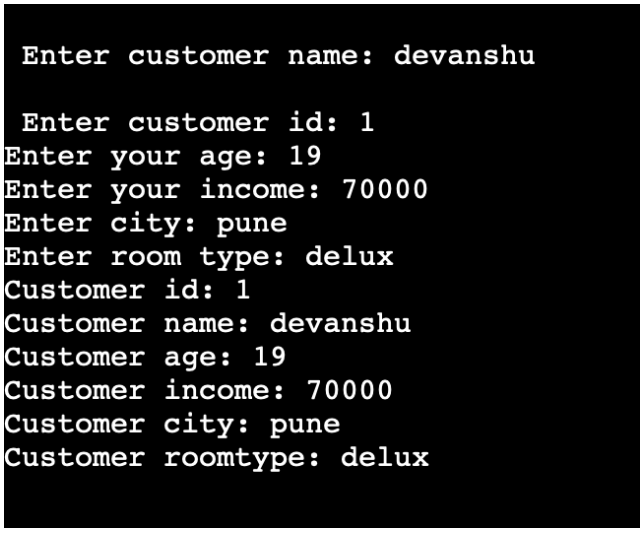
```
        cout<<"Room must be delux or super delux!!"<<endl;
    }
}

void hotel::display()
{
    cout<<"Customer id: "<<cust_id<<endl;
    cout<<"Customer name: "<<cust_name<<endl;
    cout<<"Customer age: "<<age<<endl;
    cout<<"Customer income: "<<income<<endl;
    cout<<"Customer city: "<<city<<endl;
    cout<<"Customer roomtype: "<<room_type<<endl;
}

int main()
{
    hotel h1,h2;
    h1.accept();
    ofstream out("hotel.txt",ios::out);
    out.write((char*)&h1,sizeof(h1));
    out.close();
    ifstream in("hotel.txt",ios::in);
    in.read((char*)&h2,sizeof(h2));
    in.close();
    h2.display();
return 0;
```

}

**OUTPUT:**


Enter customer name: devanshu
Enter customer id: 1
Enter your age: 19
Enter your income: 70000
Enter city: pune
Enter room type: delux
Customer id: 1
Customer name: devanshu
Customer age: 19
Customer income: 70000
Customer city: pune
Customer roomtype: delux