ith private data
nt income, string city
has a default
of class Hotel:
nput values for
es of data
mber functions of
getcity(), get room
llows:
city Not Valid
Room type not
check if above
sted satisfied
file.

Devanshu Surana
PC-23, 1032210755
Batch C1

OOCCJ Lab Assignment 4B

Aim: Write a Java program to showcase the use of File I/o Exception Handling in Java.

Part - 1 (file I/o)

A) Read the data from console (Student Rno, name, marks of 5 subjects [calculate Grade] by InputStreamReader and BufferReader and write the same to another file using InputStreamWriter and BufferedWriter.

Objectives:
- To study I/o stream classes
- To study Java abstract class.

Theory:

Abstraction is a process of hiding the implementation details and showing only functionality to the user. Another way, it shows only essential things to the user and hides the internal details.

for ex: sending SMS where you type the text and send the message.

It can be achieved with either abstract class or interfaces.

Java I/O is used to process the input and produce the output. The java.io package contains all classes required for input and output operation.

Stream is a sequence of data. It is composed of

Input stream:
Java application uses an input stream to read data a source, it may be a file, an array, peripheral socket.

Output Stream:
Java application uses an output stream to write to a destination, it may be a file, an array, peri-eral device or socket.

Algorithm:
step 1: start
step 2: Create object of fileWriter
step 3: Pass this object as a parameter to the Buffe class.
step 4: Declare variables
step 5: Take Input
step 6: Write the input value onto the file.
Step 7: close the file.
step 8: open the txt file to ensure everything is properly.
Step 9: End.

Input: Data of students
Output: file displays all records.

Conclusion: Thus, we have successfully implemented I/o stream classes.

Teacher's Sign

FAQ's

1) Abstract class ca
while concrete cl
Abstract class m
concrete class
Abstract class ca
Class can be de

2)

Array	chara charA char
String	string Strin
pipe	pip pi
file	fi fi

is composed of by

am to read data fr
ay, peripheral devi

am to write dat
, an array, periph-

to the Buffered

ile.

verything is adde

plemented usage

acher's Sign.

Page No.
Date

EPS's

Abstract class cannot be instantiated using new keyword while concrete class can be instantiated using new keyword.
Abstract class may or may not have abstract methods but concrete class can not have abstract methods.
Abstract class can not declared as final class while concrete class can be declared final.

	Character stream	Byte stream
Array	CharArrayReader	Byte Array Input Stream
	Char Array Writer	Byte Array Output stream
String	String Reader	String Buffer Input stream
	String Writer	
pipe	piped Reader	piped Input stream
	piped Write	piped Output stream
file	file Reader	file Input Stream
	file Writer	file output stream

Teacher's Sign.

Part - 2 (Exception)

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

1. Amount should be created with minimum amount of 1000rs.
2. For withdrawl of amount, if with_amt > amount
3. cid should be in specific range of 1 to 20.
4. Entered amount should be positive.

Objectives:

Understand the different types of exceptions.
Exception Handling using various Exception classes.

Theory:

The exception handling in Java is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.

In Java, an exception is an event that disturbs the normal flow of program. It is an object which is thrown at runtime. It is a mechanism to handle runtime errors.

Java Exception Keywords:

try, catch, finally, throw, throws.

for eg:

```
public class Java {
    public static void main () {
        try {
            int data = 10010;
        }
        catch (Exception e)
            { S.O.P ("Error"); }
    }
}
```

Types:
1) Checked Exception:
classes that directly inherit the throwable class
Runtime Exception & Error are known as Checked Exc
2) Unchecked Exception:
classes that inherit the Runtime Exception are k
as unchecked Exceptions are checked during runtim
3) Error: Errors are irrecoverable.

Pseudo code:

```
class Employee extends Exception {
        Employee (string s)
          { super (s); }
    }
class sample Employee {
        check id<0 or >999
        throw error;
    }
main () {
try { empcheck (1111);
    };
catch (e) {
    print error;
    }
}
```

Input: EmpId 1111 is passed
Output: Throw Error.

Conclusion
Thus, studied the exception handling concepts in Java

Teacher's Sign

FAQ's

1) The err
the sc
issues
In Java
can be

2) chec

The a
unchec

3) i) tr
ii) ca
iii) f
iv) t
v) th

4) "th
ke

FAQ's

The error indicates trouble that primarily occurs due to the scarcity of system resources. The exceptions are the issues that can occur at runtime and compile time. In Java all the errors are unchecked while exceptions can be both checked and unchecked.

checked

```
Class main {
    public static void main() {
        filereader fr = newfileReader("Path");
        BufferedReader br = new Bufferedreader (fr)
        System.out.println (fr Input readline ());
            fr close();
    }
}
```

The above error can be avoided by using try catch block.

unchecked

```
int x = 1010;
```

It throws arithmetic Exception during runtime.

3) i) try: Used to specify the exception block.
ii) catch: used to specify the solution
iii) finally: The 'finally' keyword has a mandatorily executable code.
iv) throw: Throws an exception
v) throws: used to declare an exception

4) "throw" keyword throws an exception while the "throws" keyword is used to declare an exception.

Teacher's Sign.:

Name: Devanshu Surana

Roll No.: 23

Panel: C

Batch: C1

OOCCJ Lab Assignment 4B

CODE:

```
import java.io.*;
import java.util.Scanner;

class filejava
{
   public static void main(String[] args)throws IOException
   {
      FileWriter fw=new FileWriter("stud.txt");
      BufferedWriter bw=new BufferedWriter(fw);

      String name;
      int rollno;
      int marks[]=new int[5];

      Scanner sc=new Scanner(System.in);
      System.out.println("Enter name:");
      name=sc.next();
      System.out.println("Enter roll no.:");
      rollno=sc.nextInt();
```

```
        for(int i=0;i<marks.length;i++)
        {
           System.out.println("Enter marks:");
           marks[i]=sc.nextInt();
        }
        bw.write(name);
        bw.write(""+rollno);

        for(int i=0;i<marks.length;i++)
        {
           bw.write(""+marks[i]);
        }
        bw.close();
        fw.close();
    }
}
```

OUTPUT:

```
Enter name:
Devanshu
Enter roll no.:
23
Enter marks:
98
Enter marks:
87
Enter marks:
75
Enter marks:
68
Enter marks:
81
```