B. Sc. (H) Computer Science Semester II
BHCS03 – Programming in JAVA

Topic:

- **Java I/O**

Basic I/O

Java I/O (Input and Output) is used to process the input and produce the output.

Java uses the concept of a stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.

We can perform file handling in Java by Java I/O API.

Streams

A stream is a sequence of data. In Java, a stream is composed of bytes. It's called a stream because it is like a stream of water that continues to flow.

In Java, 3 streams are created for use automatically. All these streams are attached with the console.

1) System.out: standard output stream

2) System.in: standard input stream

3) System.err: standard error stream

Code to print output and an error message to the console.

```
System.out.println("simple message");

System.err.println("error message");
```

Code to get input from console.

```
int i=System.in.read();//returns ASCII code of 1st character

System.out.println((char)i);//will print the character
```

Output stream vs. Input stream

Java application uses an output stream to write data to a destination; it may be a file, an array, peripheral device or socket. Java application uses an input stream to read data from a source; it may be a file, an array, peripheral device or socket.

Input Stream Classes

Java's input stream classes are used to read 8-bit bytes from the stream. The InputStream class is the superclass for all byte-oriented input stream classes. All the methods of this class throw an IOException. Being an abstract class, the InputStream class cannot be instantiated hence, its subclasses are used. Some of these are listed in the following Table

Class	Description
BufferedInputStream	contains methods to read bytes from the buffer (memory area)
ByteArrayInputStream	contains methods to read bytes from a byte array
DataInputStream	contains methods to read Java primitive data types
FileInputStream	contains methods to read bytes from a file
FilterInputStream	contains methods to read bytes from other input streams which it uses as its basic source of data
ObjectInputStream	contains methods to read objects
PipedInputStream	contains methods to read from a piped output stream. A piped input stream must be connected to a piped output stream
SequenceInputStream	contains methods to concatenate multiple input streams and then read from the combined stream

The Input Stream class defines various methods to perform reading operations on data of an input stream. Some of these methods along with their description are listed in the following Table

Method	Description
int read()	returns the integral representation of the next available byte of input. It returns -1 when end of file is encountered
int read (byte buffer [])	attempts to read buffer. length bytes into the buffer and returns the total number of bytes successfully read. It returns -1 when end of file is encountered
int read (byte buffer [], int loc, int nBytes)	attempts to read 'nBytes' bytes into the buffer starting at buffer [loc] and returns the total number of bytes successfully read. It returns -1 when end of file is encountered
void close ()	closes the input source. If an attempt is made to read even after closing the stream then it generates IOException

Output Stream Classes

Java's output stream classes are used to write 8-bit bytes to a stream. The OutputStream class is the superclass for all byte-oriented output stream classes. All the methods of this class throw an IOException. Being an abstract class, the Output Stream class cannot be instantiated hence, its subclasses are used. Some of these are listed in the following Table

Class	Description
BufferedOutputStream	Contains methods to write bytes into the buffer
ByteArrayOutputStream	Contains methods to write bytes into a byte array
DataOutputStream	Contains methods to write Java primitive data types
FileOutputStream	Contains methods to write bytes to a file
FilterOutputStream	Contains methods to write to other output streams
ObjectOutputStream	Contains methods to write objects
PipedOutputStream	Contains methods to write to a piped output stream
PrintStream	Contains methods to print Java primitive data types

The OutputStream class defines methods to perform writing operations. These methods are discussed in the following Table

Method	**Description**
void write (int i)	writes a single byte to the output stream
void write (byte buffer [])	writes an array of bytes to the output stream
Void write(bytes buffer[],int loc, int nBytes)	writes 'nBytes' bytes to the output stream from the buffer b starting at buffer [loc]
void close ()	closes the output stream. If an attempt is made to write even after closing the stream then it generates IOException

Please solve/refer examples discussed on pages 305-308 (for reading console input, writing console output) of reference book [*].

*Schildt, H. Java: The Complete Reference. 9th edition. McGraw-Hill Education.

Reading and Writing Files

Java FileOutputStream is an output stream used for writing data to a file.

If you have to write primitive values into a file, use FileOutputStream class. You can write byte-oriented as well as character-oriented data through FileOutputStream class. But, for character-oriented data, it is preferred to use FileWriter than FileOutputStream.

Java File Output Stream

Example 1: write byte

```
import java.io.FileOutputStream;
public class FileOutputStreamExample {
  public static void main(String args[]){
       try{
         FileOutputStream fout=new FileOutputStream("D:\\testout.txt");
         fout.write(65);
         fout.close();
```

```
 }catch(Exception e){System.out.println(e);}
    }
}
```

Output:

```
Success...
```

The content of a text file testout.txt is set with the data A.

testout.txt

```
A
```

Example 2: write string

```
import java.io.FileOutputStream;
public class FileOutputStreamExample {
   public static void main(String args[]){
          try{
            FileOutputStream fout=new FileOutputStream("D:\\testout.txt");
            String s="Welcome to javaTpoint.";
            byte b[]=s.getBytes();//converting string into byte array
            fout.write(b);
            fout.close();
            System.out.println("success...");
           }catch(Exception e){System.out.println(e);}
     }
}
```

Output:

```
Success...
```

Java FileInputStream Class

Java FileInputStream class obtains input bytes from a file. It is used for reading byte-oriented data (streams of raw bytes) such as image data, audio, video etc. You can also read character-stream data. But, for reading streams of characters, it is recommended to use FileReader class.

Example 1: read single character

```
import java.io.FileInputStream;
public class DataStreamExample {
    public static void main(String args[]){
```

```
try{
           FileInputStream fin=new FileInputStream("D:\\testout.txt");
           int i=fin.read();
           System.out.print((char)i);

           fin.close();
         }catch(Exception e){System.out.println(e);}
        }
       }
```

Note: Before running the code, a text file named as "testout.txt" is required to be created. In this file, we are having following content:

Welcome to java.

After executing the above program, you will get a single character from the file which is 87 (in byte form). To see the text, you need to convert it into character.

Output:

```
W
```

Example 2: read all characters

```
import java.io.FileInputStream;
public class DataStreamExample {
    public static void main(String args[]){
         try{
           FileInputStream fin=new FileInputStream("D:\\testout.txt");
           int i=0;
           while((i=fin.read())!=-1){
            System.out.print((char)i);
           }
           fin.close();
         }catch(Exception e){System.out.println(e);}
        }
       }
```

Output:

```
Welcome to java
```

Please solve/refer examples discussed on pages 309-315 of reference book [*].

*Schildt, H. Java: The Complete Reference. 9th edition. McGraw-Hill Education.

NOTE: The following resources are used to prepare the content written above. Please go through the reference book also for the above topic and feel free to mail your doubts or discuss anything.

References/Resources:

1. Balaguruswamy, E. (2014). Programming with JAVA: A Primer. 5th edition. India: McGraw Hill Education
2. Horstmann, C. S. (2017). Core Java - Vol. I – Fundamentals (Vol. 10). Pearson Education
3. https://www.tutorialspoint.com
4. https:// www.javatpoint.com
5. https://beginnersbook.com
6. http://ecomputernotes.com
7. https://www.includehelp.com
8. Schildt, H. (2018). Java: The Complete Reference. 10th edition. McGraw-Hill Education.

Assignment

Q1. What will be the output (write explanation also) of the below program?

```
import java.io.*;
class Chararrayinput
{
   public static void main(String[] args)
   {
         String obj  = "abcdef";
      int length = obj.length();
      char c[] = new char[length];
      obj.getChars(0, length, c, 0);
      CharArrayReader input1 = new CharArrayReader(c);
      CharArrayReader input2 = new CharArrayReader(c, 0, 3);
      int i;
      try
      {
               while((i = input2.read()) != -1)
         {
            System.out.print((char)i);
         }
          }
catch (IOException e)
        {
           e.printStackTrace();
            }
         }
   }
```

Q2. Write a program that copies content of one file to another. Pass the names of the files through command-line arguments. (Try on machine also, if possible)

Q3. Write a program to read a file and display only those lines that have the first two characters as '//' (Use try with resources). (Try on machine also, if possible)

Q4. How do you handle console output using PrintWriter class?

Q5. How do you read 1) characters 2) a string , using BufferedReader class?

Q6. Write name and meaning of all stream classes discussed in reference book.