Stream and IO

Stream is a logical entity that either produces or consumes information. Streams implement *sequential* access of data. There are two kinds of streams:

i. Input stream: An input stream is an object from which we can read a sequence of bytes.
ii. Output stream: An output stream is an object from which we can write a sequence of bytes.

The input stream acts as a source of data whereas the output stream acts a a destination of the data. Some entities can act as both input and output streams:

an array of bytes or characters
a file
a pipe
a network connection

The `java.io` package contains a collection of stream classes that support these algorithms. To use these classes, a program needs to import the `java.io` package.

The stream classes are divided into two class hierarchies, based on the data type they operate on:

- Characters - 16 bit Unicode : to read and write text data only.

- Bytes – read/write data in bytes (8 bits) : to work with images, characters, videos, audios, etc.

Example #1:

The following program demonstrates the use of the `BufferedReader` class . This program is designed to read integer numbers from the user interactively.

```
 2  import java.io.*;
 3  public class InputStreamDemo {
 4    public static void main(String[] args) {
 5      String rawData = null;
 6      int intVal = 0;
 7      System.out.print("Please input an integer: ");
 8
 9       try {
10       // create an object of BufferedReader
11          BufferedReader br = new BufferedReader(new  InputStreamReader(System.in));
12       //   now read user data
13          rawData = br.readLine();
14          intVal = Integer.parseInt(rawData);
15      } catch (NumberFormatException ex) {
16        System.err.println("Not a valid number: " + rawData);
17      }catch (IOException e) {
18        System.err.println("Unexpected IO ERROR: " + e);
19      }
20      System.out.println("Evaluated Integer value: " + intVal);
21    }
22  }
23
```

Example #2:

The following program opens a file reader on input.txt and opens a file writer on outagain.txt. The program reads characters from the reader as long as there's more input in the input file and writes those characters to the writer. When the input runs out, the program closes both the reader and the writer.

```
3   import java.io.*;
4   public class FileCopyDemo {
5
6       public static void main(String[] args) throws IOException {
7
8           File inputFile = new File("inputFile.txt");
9           File outputFile = new File("outputFile.txt");
10
11          FileReader in = new FileReader(inputFile);
12
13        FileWriter out = new FileWriter(outputFile);
14          int c;
15
16          while ((c = in.read()) != -1) {
17              out.write(c);
18          }
19          in.close();
20          out.close();
21      }
22  }
23
```

The data in inputFile.txt is copied to a new file named outputFile.txt in the same directory.

Example#3:

Program to compress files ("Student.ser", "inputFile.txt") to “outfile.zip"

```
3  import java.io.*;
4  import java.util.*;
5  import java.util.zip.*;
6  public class TestZip{
7      public static void main(String args[]){
8          // These are the files to include in the ZIP file
9          String[] filenames = new String[]{"Student.ser", "inputFile.txt"};
10
11         // Create a buffer for reading the files
12         byte[] buf = new byte[1024];
13
14         try {
15             // Create the ZIP file
16             String outFilename = "outfile.zip";
17             ZipOutputStream out = new ZipOutputStream(new FileOutputStream(outFilename));
18
19             // Compress the files
20             for (int i=0; i<filenames.length; i++) {
21                 FileInputStream in = new FileInputStream(filenames[i]);
22
23                 // Add ZIP entry to output stream.
24                 out.putNextEntry(new ZipEntry(filenames[i]));
25
26                 // Transfer bytes from the file to the ZIP file
27                 int len;
28                 while ((len = in.read(buf)) > 0) {
29                     out.write(buf, 0, len);
30                 }
31                 // Complete the entry
32                 out.closeEntry();
33                 in.close();
34             }
35
36             // Complete the ZIP file
37             out.close();
38         } catch (IOException e) {
39
40         }
41     }
42 }
43
```

Example#4:

Program to list the files in “outfile.zip"

```
3   import java.io.*;
4   import java.util.*;
5   import java.util.zip.*;
6
7   public class ZipContents{
8
9       public static void main(String args[])  {
10          try {
11              // Open the ZIP file
12              ZipFile zf = new ZipFile("outfile.zip");
13
14              // Enumerate each entry
15              for (Enumeration entries = zf.entries(); entries.hasMoreElements();) {
16                  // Get the entry name
17                  String zipEntryName = ((ZipEntry)entries.nextElement()).getName();
18                  System.out.println("name: "+ zipEntryName);
19              }
20          } catch (IOException e) {
21          }
22      }
23  }
24
```

Example#5: Program to extract the files in “outfile.zip"

```
1  import java.io.*;
2  import java.util.*;
3  import java.util.zip.*;
4
5  public class ZipRetrive{
6
7      public static void main(String args[]){
8
9          try {
10             // Open the ZIP file
11             String inFilename = "outfile.zip";
12             ZipInputStream in = new ZipInputStream(new FileInputStream(inFilename));
13             ZipFile zf= new ZipFile(inFilename);
14             int a=0;
15             OutputStream out=null;
16             for (Enumeration em = zf.entries(); em.hasMoreElements();) {
17
18                 String outFilename = em.nextElement().toString();
19                 // Get the  entry
20                 ZipEntry entry = in.getNextEntry();
21
22                // Open the output file
23
24                 out = new FileOutputStream(outFilename);
25
26              // Transfer bytes from the ZIP file to the output file
27                 byte[] buf = new byte[1024];
28                 int len;
29                 while ((len = in.read(buf)) > 0) {
30                     out.write(buf, 0, len);
31                 }
32                 a=a+1;
33             }
34
35             if(a>0)
36                 System.out.println("Files Unzipped");
37
38             // Close the streams
39             out.close();
40             in.close();
41         } catch (IOException e) {
42             System.out.println("Error!");
43         }
44     }
45
46 }
47
```

Example #6: Object Stream

Object Serialization: The object of class Student is serialized and kept into Student.ser file. Next, the file is read and the deserialization is done to obtain the original object. Observe the behaviour of the transient variable used in the program.

```
// Student.java
public class Student implements java.io.Serializable
{
   public String name;
   public String address;
   public transient int regNum;
   public int number;
   public void mailResult()
   {
      System.out.println("Result of " + name  + " " );
   }
}


//SerializeDemo.java
import java.io.*;
public class SerializeDemo
{
   public static void main(String [] args)
   {
      Student e = new Student();
      e.name = "Tom Swayer";
      e.address = "Pokhara, Nepal";
      e.regNum = 123456;
      e.number = 101;
      try
      {
         FileOutputStream fileOut =
         new FileOutputStream("Student.ser");
         ObjectOutputStream out = new ObjectOutputStream(fileOut);
         out.writeObject(e);
         out.close();
         fileOut.close();
         System.out.printf("Serialized data is saved as Student.ser \n");
      }catch(IOException i)
      {
          i.printStackTrace();
      }
   }
}
```

```java
//DeserializeDemo.java
import java.io.*;
public class DeserializeDemo
{
   public static void main(String [] args)
   {
      Student e = null;
      try
      {
         FileInputStream fileIn = new FileInputStream("Student.ser");
         ObjectInputStream in = new ObjectInputStream(fileIn);
         e = (Student) in.readObject();
         in.close();
         fileIn.close();
      }catch(IOException i)
      {
         i.printStackTrace();
         return;
      }catch(ClassNotFoundException c)
      {
         System.out.println("Student class not found");
         c.printStackTrace();
         return;
      }
      System.out.println("Deserialized Student...");
      System.out.println("Name: " + e.name);
      System.out.println("Address: " + e.address);
      System.out.println("Reg Num: " + e.regNum);
      System.out.println("Number: " + e.number);
    }
}
```