I/O Operation in Java

Presented by

Dr. Rubeena Doomun

I/O operation

- The java.io package contains classes required to perform input and output (I/O) in Java.
- A stream can be defined as a sequence of data.
- The **InputStream** is used to read data from a source and the **OutputStream** is used for writing data to a destination.

Standard Streams

- **Standard Input** – This is used to feed the data to user's program and usually a keyboard is used as standard input stream and represented as **System.in**.
- **Standard Output** – This is used to output the data produced by the user's program and usually a computer screen is used for standard output stream and represented as **System.out**.
- **Standard Error** – This is used to output the error data produced by the user's program and usually a computer screen is used for standard error stream and represented as **System.err**.

Buffered Reader

- In Java, there are multiple ways to obtain user input from the keyboard.
- One of the objects that could get the job done is a **Buffered Reader** using the **readline method,** along with an **Input Stream Reader.**
- A buffered reader's role is to buffer input from the underlying "Reader" object, for more efficient use later.
- If we didn't use a buffered reader, we would have had to use an input stream reader on its own, and read the line of text character by character, storing them in a character array, and then converting them to a string, which is not very efficient.

Example of how to use an input stream reader and a buffered reader to obtain a line of text from the user

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;


public class BufferedReaderExample {

    public static void main(String[]args)throws IOException{

        InputStreamReader ISR = new InputStreamReader(System.in);
        BufferedReader BR = new BufferedReader(ISR);

        System.out.println("What's your name?");
        String userInput = BR.readLine(); //program waits here until the user inserts a line of text

        System.out.println("Your name is : "+userInput);

        BR.close();
        ISR.close();
    }

}
```

Reading and Writing Files

- **FileInputStream** and **FileOutputStream**

```
import java.io.*;
public class CopyFile {

   public static void main(String args[]) throws IOException {
      FileInputStream in = null;
      FileOutputStream out = null;

      try {
         in = new FileInputStream("input.txt");
         out = new FileOutputStream("output.txt");

         int c;
         while ((c = in.read()) != -1) {
            out.write(c);
         }
      }finally {
         if (in != null) {
            in.close();
         }
         if (out != null) {
            out.close();
         }
      }
   }
}
```

Database connectivity

- **Java Database Connectivity** (**JDBC**) is an application programming interface (API) for the programming language Java, which defines how a client may access a database.
- JDBC connections support creating and executing statements. These may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT.

JDBC connection

- When a Java application needs a database connection, one of the DriverManager.getConnection() methods is used to create a JDBC connection.

```
try (Connection conn = DriverManager.getConnection(
     "jdbc:somejdbcvendor:other data needed by some jdbc vendor",
     "myLogin",
     "myPassword" ) ) {
     /* you use the connection here */
}  // the VM will take care of closing the connection
```

SQL Statement

- Once a connection is established, a statement can be created.

```
try (Statement stmt = conn.createStatement()) {
    stmt.executeUpdate( "INSERT INTO MyTable( name ) VALUES ( 'my name' ) " );
}
```

References

- http://voidexception.weebly.com/getting-input-from-the-keyboard---using-java-buffered-readers-and-input-stream-readers.html
- https://www.javatpoint.com/java-io
- https://www.javatpoint.com/example-to-connect-to-the-mysql-database
- https://en.wikipedia.org/wiki/Java_Database_Connectivity