Create, Read and Write File

Create a File in Java

- We can create a file in Java using createNewFile() method. This method creates an empty file, if the file doesn't exist at the specified location and returns true. If the file is already present then this method returns false.
- It throws:
  - ❖IOException – If an Input/Output error occurs during file creation.
  - ❖SecurityException – If a security manager exists and its SecurityManager.checkWrite(java.lang.String) method denies write access to the file.

The below code would create a txt file named “newfile.txt” in C drive. You can change the path in the below code in order to create the file in different directory or in different drive.

```
import java.io.File;
import java.io.IOException;

public class CreateFileDemo
{
  public static void main( String[] args )
  {
    try {
                File file = new File("C:\\Users\\TOSHIBA\\eclipse-workspace\\newfile.txt");
 /*If file gets created then the createNewFile() method would return true or if the file is  already present it would
return false    */
          boolean fvar = file.createNewFile();
                if (fvar){
                    System.out.println("File has been created successfully");
                }
                else{
                    System.out.println("File already present at the specified location");
                }
            } catch (IOException e) {
                        System.out.println("Exception Occurred:");
                         e.printStackTrace();
             }
  }
}
```

Read file in Java – BufferedInputStream

We can read a file in Java using FileInputStream and BufferedInputStream.

Example of Demo Steps:

1) Created a File instance by providing the full path of the file(which we will read) during File Object creation.
2) Passed the file instance to the FileInputStream which opens a connection to the actual file, the file named by the File object file in the file system.
3) Passed the FileInputStream instance to BufferedInputStream which creates a BufferedInputStream and saves its argument, the input stream, for later use. An internal buffer array is created and stored in buf using which the read operation gives good performance as the content is readily available in the buffer.
4) Used while loop to read the file. Example, method available() is used for checking the end of the file as it returns 0 when the pointer reaches to the end of the file. Read the file content using read() method of FileInputStream.

```
import java.io.*;
public class ReadFileDemo {
  public static void main(String[] args) {
    //Specify the path of the file here
    File file = new File("C://myfile.txt");
    BufferedInputStream bis = null;
    FileInputStream  fis= null;

    try
    {
       //FileInputStream to read the file
       fis = new FileInputStream(file);

       /*Passed the FileInputStream to BufferedInputStream For Fast read using the buffer array.*/
       bis = new BufferedInputStream(fis);

       /*available() method of BufferedInputStream returns 0 when there are no more bytes
        * present in the file to be read*/
       while( bis.available() > 0 ){
          System.out.print((char)bis.read());
       }
     }
```

```
    catch(FileNotFoundException fnfe)
     {
        System.out.println("The specified file not found" + fnfe);
     }
     catch(IOException ioe)
     {
        System.out.println("I/O Exception: " + ioe);
     }
     finally
     {
        try{
          if(bis != null && fis!=null)
          {
                    fis.close();
                   bis.close();
          }
         } catch(IOException ioe)
          {
             System.out.println("Error in InputStream close(): " + ioe);
          }
     }
  }
}
```

Read file in Java using BufferedReader

- There are two ways to read a file using BufferedReader.

➢ Method 1: Using readLine() method of BufferedReader class.

```
public String readLine() throws IOException
```

It reads a line of text.

➢ Method 2: Using read() method

```
public int read() throws IOException
```

It reads a character of text. Since it returns an integer value, it needs to be explicitly cast as char for reading the content of file.

Here I have two txt files myfile.txt and myfile2.txt. In order to demonstrate both the ways to read file. I'm reading first file using readLine() method while the second file is being read using read() method.

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

public class ReadFileDemo {
  public static void main(String[] args) {

    BufferedReader br = null;
    BufferedReader br2 = null;
    try{
      br = new BufferedReader(new FileReader("B:\\myfile.txt"));

      //One way of reading the file
          System.out.println("Reading the file using readLine() method:");
          String contentLine = br.readLine();
          while (contentLine != null) {
            System.out.println(contentLine);
            contentLine = br.readLine();
          }
```

```
br2 = new BufferedReader(new
FileReader("B:\\myfile2.txt"));
//Second way of reading the file
System.out.println("Reading the file using read()
method:");
                int num=0;
                char ch;
                while((num=br2.read()) != -1)
                {
                 ch=(char)num;
                   System.out.print(ch);
                }

      }
      catch (IOException ioe)
      {
                ioe.printStackTrace();
      }
```

```
finally
      {
                try {
                  if (br != null)
                            br.close();
                  if (br2 != null)
                            br2.close();
                }
                catch (IOException ioe)
                 {
                  System.out.println("Error in closing the
BufferedReader");
                }
              }
   }
}
```

Write to a file in java using FileOutputStream

- We can use FileOutputStream to write to a file in java. We would be using write() method of FileOutputStream to write the content to the specified file.

Here is the signature of write() method.

```
public void write(byte[] b) throws IOException
```

- It writes b.length bytes from the specified byte array to this file output stream. As you can see this method needs array of bytes in order to write them into a file. Hence we would need to convert our content into array of bytes before writing it into the file.

In the below example we are writing a String to a file. To convert the String into an array of bytes, we are using getBytes() method of String class.

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

public class WriteFileDemo {
   public static void main(String[] args) {
      FileOutputStream fos = null;
      File file;
      String mycontent = "This is my Data which need to be written into the file";
      try {
         //Specify the file path here
                file = new File("C:/myfile.txt");
                fos = new FileOutputStream(file);
  /* This logic will check whether the file exists or not. If the file is not found at the specified location it would
create  a new file*/
                if (!file.exists()) {
                   file.createNewFile();
                }
```

```
/*String content cannot be directly written into a file. It needs to be converted into bytes   */
                byte[] bytesArray = mycontent.getBytes();
                fos.write(bytesArray);
                fos.flush();
                System.out.println("File Written Successfully");
      }
      catch (IOException ioe) {
                ioe.printStackTrace();
      }
      finally {
                try {
                  if (fos != null)
                  {
                            fos.close();
                  }
        }
                catch (IOException ioe) {
                  System.out.println("Error in closing the Stream");
                }
      }
   }
}
```

Write to a file in java using PrintWriter

```
import java.io.*;
public class WriteData {
public static void main(String[] args) throws IOException {
File file = new File("C:\\Users\\TOSHIBA\\eclipse-workspace\\Welcome2.txt");
    if (file.exists()) {
      System.out.println("File already exists");
      }

    // Create a file
    PrintWriter output = new PrintWriter(file);

    // Write formatted output to the file
    output.print("John T Smith ");
    output.println(90);
    output.print("Eric K Jones ");
    output.println(85);

    // Close the file
    output.close();
  }
}
```