**- INPUT/OUTPUT (IO) –**

|  |
| --- |
| **Input:** Inserting information in the program.   1. InputStream (for photos an videos – reads 1 byte at a time) 2. Reader (for String values – reads 2 bytes at a time)   **Output:** Printing something from the program   1. OutputStream (for photos an videos – reads 1 byte at a time) 2. Writer (for String values – reads 2 bytes at a time) |

1. **Input**

|  |  |  |
| --- | --- | --- |
| **Input Stream** | | |
| **Object that reads** | **InputStream is = null;** | InputStream: An abstract superclass of all classes representing an input stream of bytes. |
| **Reading file** | **1 BYTE AT A TIME:**  **is = new FileInputStream("txtFile/inTest.txt");**  **while (true) {**  **i:** returned as byte  **(char) i:** returned as a character  **int i = is.read();**  **if (i == -1)**  **break;**  **System.out.print((char) i + "(" + i + ")");** | The file reads the txt in the specified file (“TxtFile/inTest.txt”)  Read(): The method reads the next byte of data from the input stream and returns it as an int. |
| **10 BYTES AT A TIME:**  **is = new FileInputStream("txtFile/inTest.txt");**  **byte[] bs = new byte[10];**  **while (true) {**  **int readByteCount = is.read(bs);**  **if (readByteCount == -1)**  **break;**  **for (int i = 0; i < readByteCount; i++) {**  **System.out.print((char) bs[i]);** | Constructs an array (bs) that is 10 bytes long so that file reads 10 bytes at a time  bs [1]: 10 bytes  bs [2]: 10 bytes  .. |
| **Exceptions** | **catch (FileNotFoundException e) {**  **System.out.println(e.getMessage());**  **} catch (IOException e) {**  **System.out.println(e.getMessage());** | **FileNotFoundException:** Exception when the file specified cannot be found  **IOException:** Exception when the file specified cannot be read |
| **Closing files** | **try {**  **if (is != null)**  **is.close();**  **} catch (IOException e) {** | When ‘is’ is empty (it has finished reading the file), we must close the ovject.  **IOException:** Exception when file cannot close |

|  |  |
| --- | --- |
| **Reader** | |
| **Object that reads** | **Reader reader = null;** | |
| **Reading file** | **reader = new FileReader("txtFile/inTest.txt"); //**  **while (true) {**  **int i = reader.read();**  **if (i == -1)**  **break;**  **System.out.print((char) i);** | |
| **Exceptions** | **} catch (IOException e) {**  **System.out.println(e.getMessage());** | |
| **Closing files** | **try {**  **if (reader != null)**  **reader.close();**  **} catch (IOException e) {** | |

1. **Output**

|  |  |  |
| --- | --- | --- |
| **Output Stream** | | |
| **Object that writes** | **OutputStream is = null;** | OutputStream: This abstract class is the superclass of all classes representing an output stream of bytes. An output stream accepts output bytes and sends them to some sink. |
| **Writing**  **file** | **WRITING IN BYTES**  **os = new FileOutputStream("txtFIle/outTest.txt", true);**  **byte[] bs = { 'H', 'e', 'l', 'l', 'o' };**  **os.write(bs);**  **System.out.println("Successfully printed file");**  Prints “Successfully printed file” in console, and Hello in the txtFIle/outTest.txt | The file chooses the destination of where it will print to: (“TxtFile/outTest.txt”)  Write():Writes bytes from the specified byte array to this output stream. |
| **WRITING STRINGS**  **os = new FileOutputStream("txtFile/outTest.txt", true);**  **String str = "Unie's\npractice String?";**  **byte[] bs = str.getBytes();**  **os.write(bs);**  **System.out.println("Successfully printed file”);** | It converts the string in to bytes and then prints in to: (“TxtFile/outTest.txt”) |
| **Exceptions** | **catch (FileNotFoundException e) {**  **System.out.println(e.getMessage());**  **} catch (IOException e) {**  **System.out.println(e.getMessage());** | **FileNotFoundException:** Exception when the file specified cannot be found  **IOException:** Exception when the file specified cannot be read |
| **Closing files** | **try {**  **if (os != null)**  **is.close();**  **} catch (IOException e) {** | When ‘is’ is empty (it has finished reading the file), we must close the ovject.  **IOException:** Exception when file cannot close |
| **Writer**  FileReader: Reads characters from a file  BufferedReader reads characters from another Reader. (Reads faster)  PrintWriter: | | |
| **Object that reads** | **Writer wrier = null;**  **BufferedReader br = null;** | | |
| **Reading file** | **READER**  **writer = new FileWriter("txtFile/outTest.txt", true);**  **String msg = "\n\nAdded text file. Hello";**  **writer.write(msg);**  **System.out.println("Successfully printed");** | | |
| **BUFFERED READER**  **reader = new FileReader("txtFile/inTest.txt”);**  **br = new BufferedReader(reader**  **int cnt = 0;**  **while (true) {**  **String linedata = br.readLine();**  **if (linedata == null)**  **break;**  **System.out.println(++cnt + linedata);** | | |
| **Exceptions** | **catch (FileNotFoundException e) {**  **System.out.println(e.getMessage());**  **} catch (IOException e) {**  **System.out.println(e.getMessage());** | | |
| **Closing files** | **try {**  **if (br != null)**  **br.close();**  **if (reader != null)**  **reader.close();**  **} catch (Exception e2) {**  **}** | | |