**1.Scanner:**

The Java Scanner class is used to collect user input. Scanner is part of the java. util package, so it can be imported without downloading any external libraries. Scanner reads text from standard input and returns it to a program.

Example:

import java.util.Scanner;

class Main {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter your name: ");

String name = input.nextLine();

System.out.println("My name is " + name);

input.close();

}

}

Output:

Enter your name: Priyanka

My name is Priyanka

**2. Flush and close:**

**Flush:**

flush() writes the content of the buffer to the destination and makes the buffer empty for further data to store but it does not closes the stream permanently. That means you can still write some more data to the stream.

The flush() method of PrintWriter Class in Java is used to flush the stream. By flushing the stream, it means to clear the stream of any element that may be or maybe not inside the stream

**Close():**

The close() method of Reader Class in Java is used to close the stream and release the resources that were busy in the stream.

This method has following results:

1.If the stream is open, it closes the stream releasing the resources.

2. If the stream is already closed, it will have no effect.

3.But close() closes the stream permanently.

**3.**. **use case scenario: if I want to read huge data file in java which class is preferred?**

If you want to read a large file with the Files class, you can use the newBufferedReader() method to obtain an instance of BufferedReader class and read the file line by line using a BufferedReader

**4. how to create excel file and pdf file in java?**

**To create excel file in java:**

**To create an excel file in Java ,** We have to rely on the third-party library that is [Apache POI.](https://www.javatpoint.com/apache-poi-tutorial)This is because Java does not provide direct API to read or write Microsoft Excel or Word documents. Apache POI (Poor Obfuscation Implementation) is a Java API for reading and writing Microsoft Documents. It contains classes and interfaces. The Apache POI library provides two implementations for reading or writing excel files:

* (HSSF (Horrible SpreadSheet Format) Implementation
* XSSF (XML SpreadSheet Format) Implementation}
* **Steps to create Excel file:**
* **Step 1:** Create a Java project in eclipse.create a Java project with the name CreateExcelEile.
* **Step 2:**  Create a class with the name CreateExcelFileExample1 and write the code that we have written in CreateExcelFileExample1.java file.
* **Step 3:**  Download the Apache POI library
* **Step 4:**  Add the Apache POI to the project. Right-click on the project -> Build Path -> Configure Build Path. It opens the Properties window for the current project.
* **Step 5:**  In the Properties window, click on the Add External JARs button.
* **Step 6:**  Go to the path where the **poi-3.17.jar**file is located. Select the JAR file and click on the **Open** button. It adds the JAR file to the project. After that, click on the **Apply and Close** button to apply the changes..
* import java.io.\*;
* import org.apache.poi.hssf.usermodel.HSSFWorkbook;
* import org.apache.poi.ss.usermodel.Workbook;
* public class CreateExcelFileExample1
* {
* public static void main(String[] args) throws FileNotFoundException, IOException
* {
* //creating an instance of Workbook class
* Workbook wb = new HSSFWorkbook();
* //creates an excel file at the specified location
* OutputStream fileOut = new FileOutputStream
* System.out.println("Excel File has been created successfully.");
* wb.write(fileOut);
* }
* }

**Output:**

Excel file has been created successfully

**To create pdf file in java:**

There are various libraries that can be used to create a PDF file using a [Java](https://www.javatpoint.com/java-tutorial) program. Some of them are:

1.Apache PDFBox

2.iText

3.PDF Clown

**Apache PDFBox:**

Here, we are going for apache pdfbox because it is an open-source library that can be used while dealing with PDF documents. It includes several command-line utilities. It allows us to create PDFs, read, and manipulate PDFs

First, we have to install the library. In order to install the library, follow the steps given below:

**Step 1:** Go to the [PDFBox](https://pdfbox.apache.org/download.html" \t "_blank) official site and download the PDFBox library. It will land you on the home page.

**Step 2:** Under the Current releases section, download all the JAR files for PDFBox 3.0.0-RC1 release.

**Step 3:** Once the required JARs are download, open eclipse IDE and create a Java project.

**Step 4:** Add all the downloaded JRAs in the Path. Right-click on the project -> Build Path -> Configure Build Path -> Add External JARs -> Select the all the downloaded JARs -> Open -> Apply -> Apply and Close.

**Step 5:** Create a class file with the name **CreatePdf** and write the following code into it.

**CreatePdf.java**

1. import java.io.IOException;
2. import org.apache.pdfbox.pdmodel.PDDocument;
3. import org.apache.pdfbox.pdmodel.PDPage;
4. public class CreatePdf
5. {
6. public static void main(String args[]) throws IOException
7. {
8. PDDocument pdfdoc= **new** PDDocument();
9. pdfdoc.addPage(**new** PDPage());
10. //path where the PDF file will be store
11. pdfdoc.save
12. //prints the message if the PDF is created successfully
13. System.out.println("PDF created");
14. //closes the document
15. pdfdoc.close();
16. }
17. }

**Output:**

PDF created