POLITEKNIK STATISTIKA STIS

For Better Official Statistics

Pemrograman Berorientasi Objek (PBO) – Pertemuan 10 (Teori)

oleh: Nano Yulian P.

Jakarta, 11-04-2022

Bahasan materi

- *Pengenalan I/O Streams (java.io)*
- *Byte Stream dan Character Stream*
- *Hirarki Class pada Package Java.io*

Penggunaan Byte based Stream dan Character Based Stream

- *- FileInputStream & FileOutputStream*
- *-ByteArrayInputStream & ByteArrayOutputStream*
- *-ObjectInputStream & ObjectOutputStream*

- *-FileReader & FileWriter*
- *-BufferedReader & BufferedWriter*

Input/Output (I/O) Streams

Streams = Aliran Data

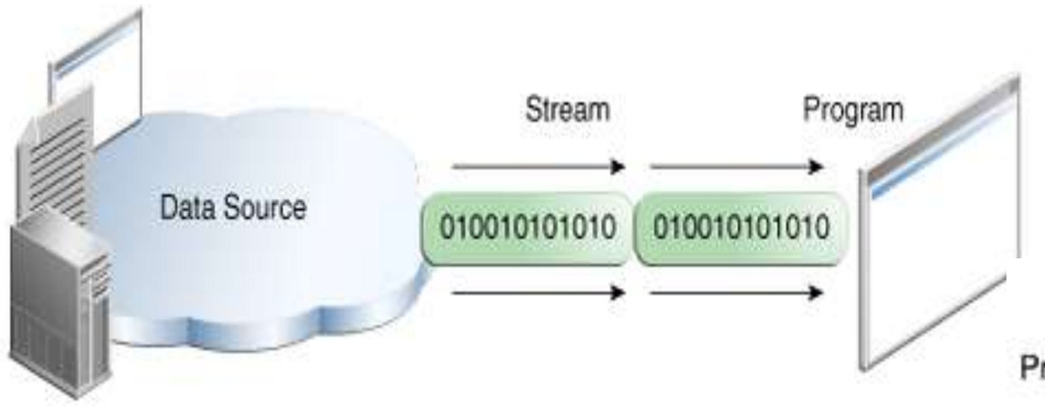
Input/Output Stream menggambarkan aliran urutan data (sekuensial) dari (***source***) ke ***(destination).***

Contoh :

- *Baca dan/atau Tulis data dari dan kedalam file di disk,*
- *Baca dan/atau Tulis data pada String ke console.*
- *Baca dan/atau tulis data dari dan melalui network, program lain, dan memory arrays.*

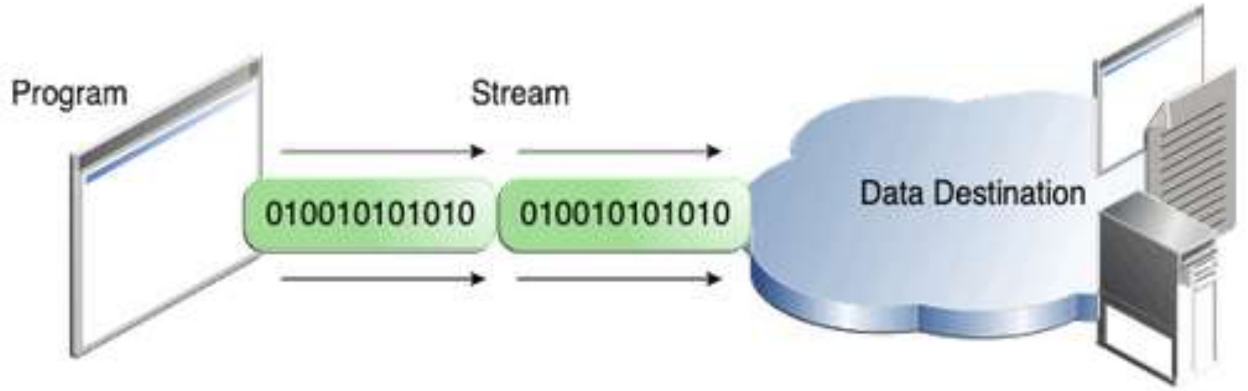
Program di Java melakukan operasi I/O menggunakan streams.

I/O Streams



Reading information into a program.

Pass/ Manipulate/Transform



Writing information from a program.

Sumber : https://docs.oracle.com/javase/tutorial/essential/io/streams.html

Program Java menggunakan ***input stream*** untuk **membaca** data dari *source*.
Program Java menggunakan ***output stream*** untuk **menulis** data ke *destination*. ***(one Length at a time)***

Jenis data : *simple bytes, primitive data types, localized characters / unicode, and objects.*

Jenis Java Stream

Byte stream (8 bit / 1 Byte)

- Cocok digunakan untuk operasi I/O yang menggunakan data biner (byte).
- 1 **Byte** at a time
- "Machine Oriented"

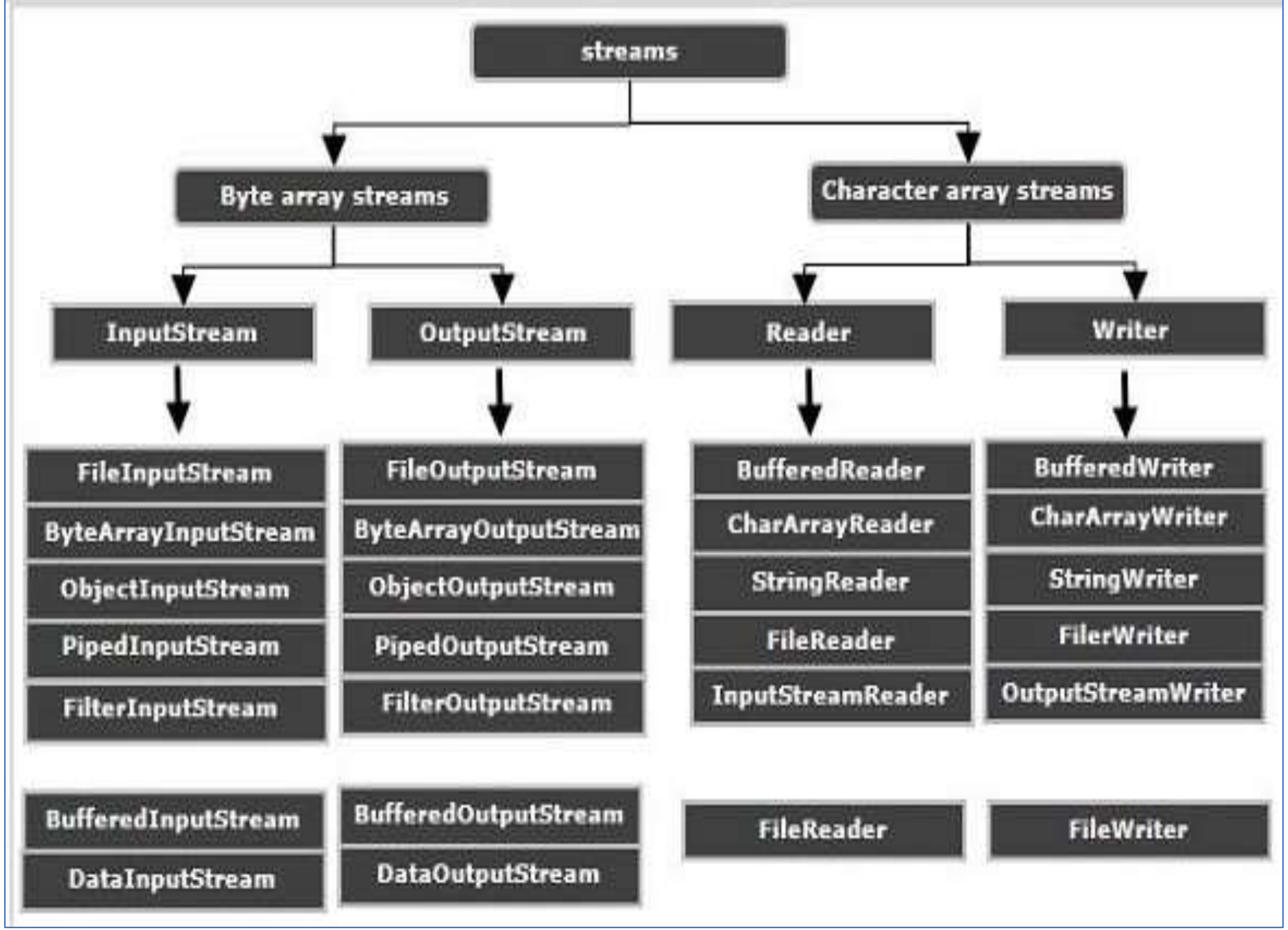
Misal : non-Unicode characters, video, image, audio, dsj..

Character stream (16 bit / 2 Bytes)

- Digunakan untuk operasi I/O yang menggunakan karakter.
- 1 **Character** at a time.
- "Human Oriented"

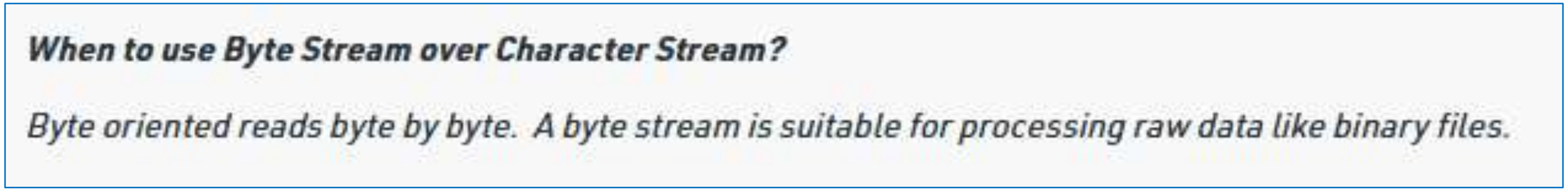
Karakter di Java menggunakan Unicode, sehingga bisa menangani karakter-karakter internasional (karakter pengembangan dari kode ASCII standar).
Misal: text files dengan character Unicode, icon, dsj...

Hirarki Class di java.IO

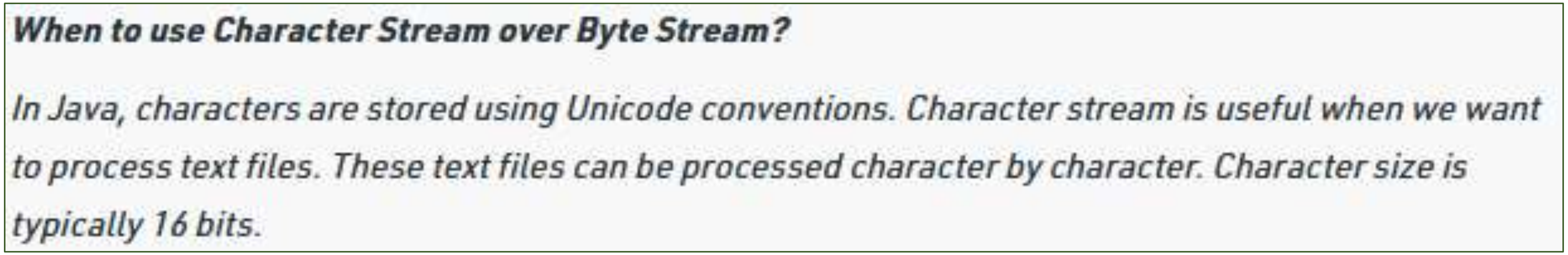


Java IO berisi banyak *subclass* dari **Abstract class**
InputStream, OutputStream,
Reader dan Writer.

Tipe Java Stream dan penggunaannya


When to use Byte Stream over Character Stream?

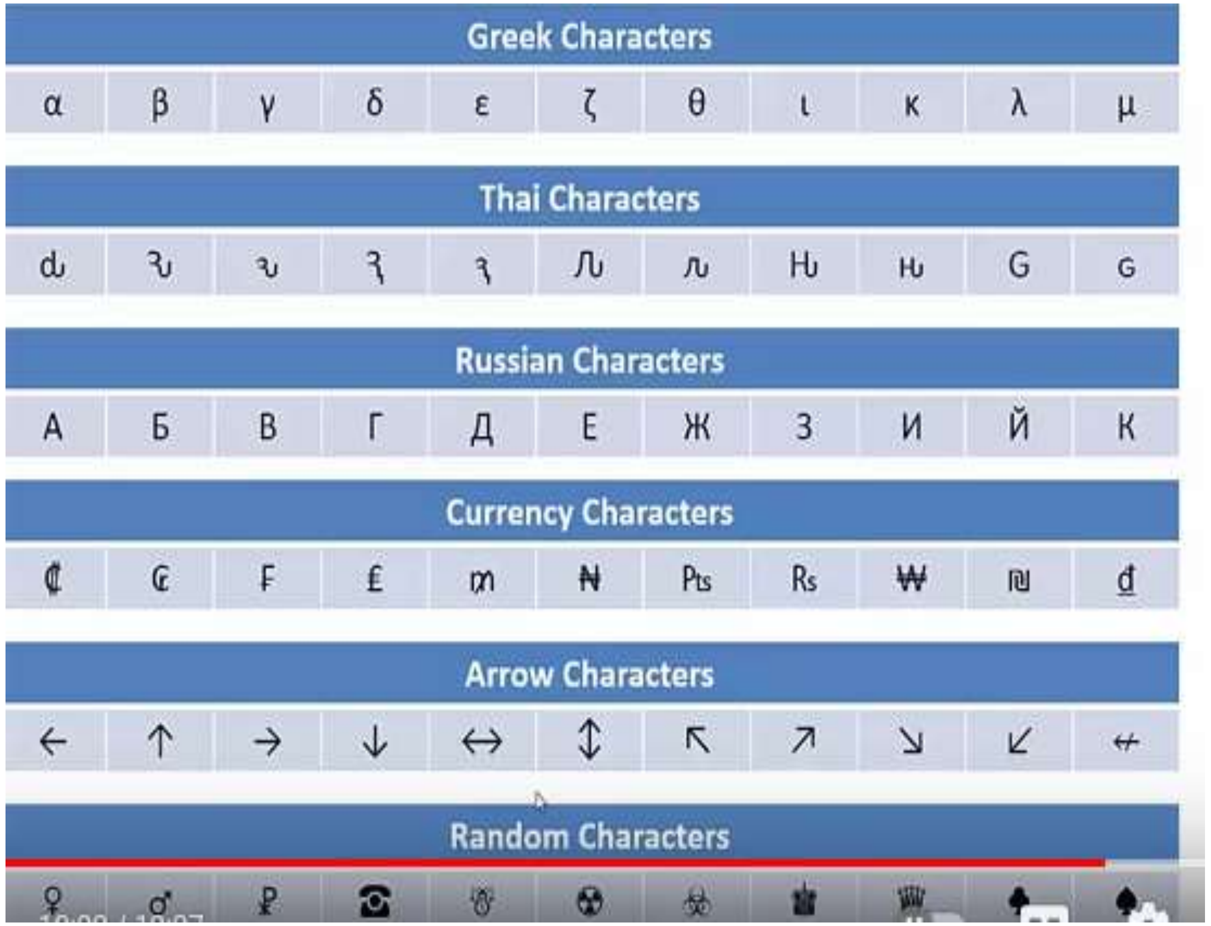
Byte oriented reads byte by byte. A byte stream is suitable for processing raw data like binary files.


When to use Character Stream over Byte Stream?

In Java, characters are stored using Unicode conventions. Character stream is useful when we want to process text files. These text files can be processed character by character. Character size is typically 16 bits.

https://www.geeksforgeeks.org/character-stream-vs-byte-stream-java/

Unicode



Greek Characters										
α	β	γ	δ	ε	ζ	θ	ι	κ	λ	μ

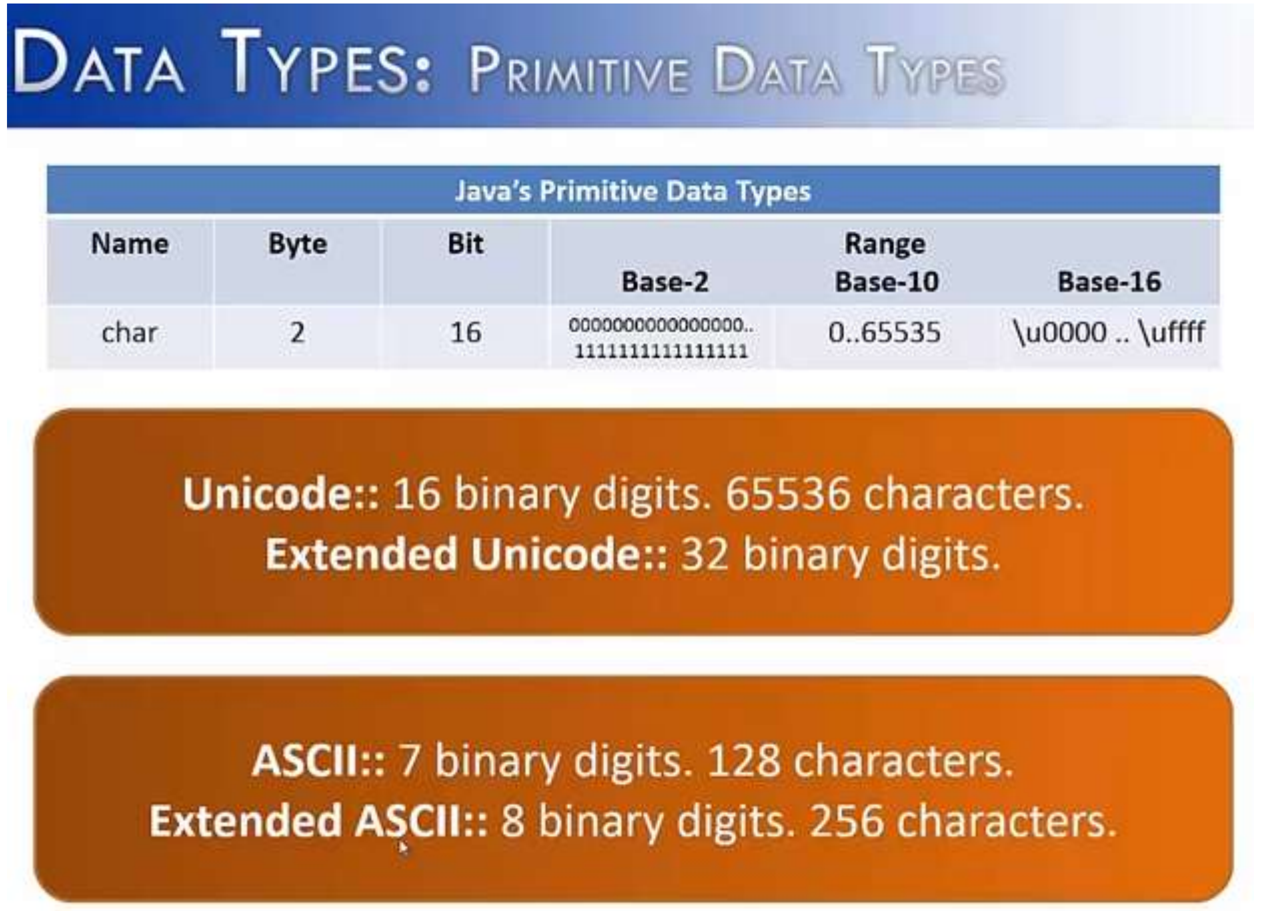
Thai Characters										
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	G	G

Russian Characters										
А	Б	В	Г	Д	Е	Ж	З	И	Й	К

Currency Characters										
₡	₢	₣	₤	₥	₦	₧	₨	₩	₪	₫

Arrow Characters										
←	↑	→	↓	↔	↕	↖	↗	↘	↙	↚

Random Characters										
♀	♂	☧	☎	☃	☢	☣	♚	♛	♣	♠



Data Types: Primitive Data Types

Java's Primitive Data Types					
Name	Byte	Bit	Range		
			Base-2	Base-10	Base-16
char	2	16	0000000000000000.. 1111111111111111	0..65535	\u0000 .. \uffff

Unicode:: 16 binary digits. 65536 characters.
Extended Unicode:: 32 binary digits.

ASCII:: 7 binary digits. 128 characters.
Extended ASCII:: 8 binary digits. 256 characters.

https://onlineutf8tools.com/convert-utf8-to-binary

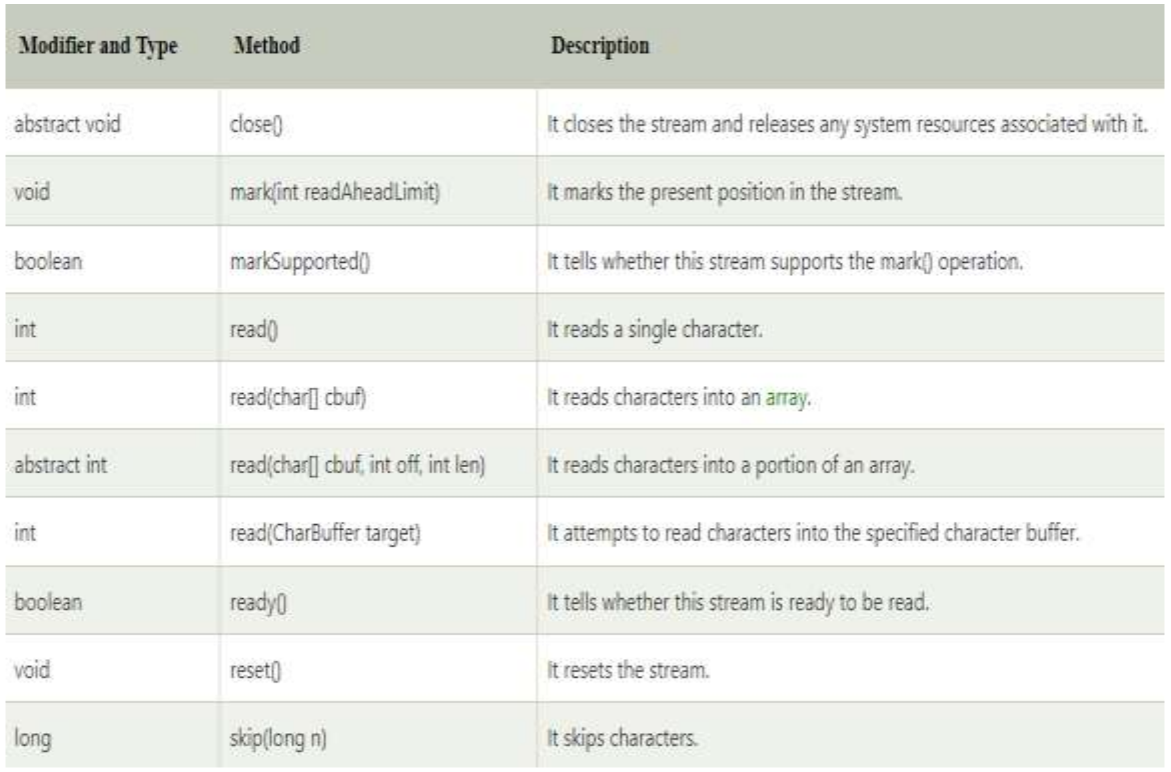
https://www.youtube.com/watch?v=61Bs7-ycL64

Methods Input/Output Stream dan Reader/Writer

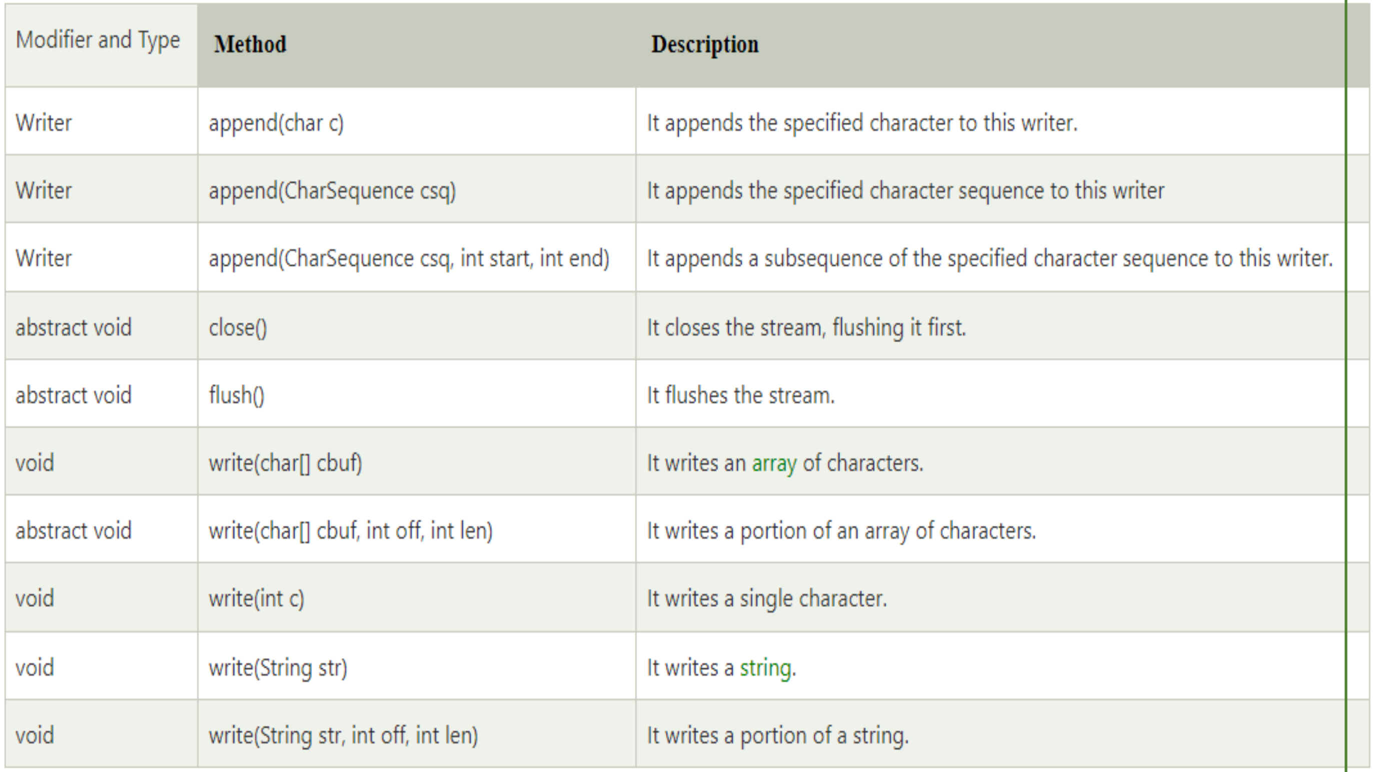
For Better Official Statistics

Useful Method of InputStream	Description
1) public abstract int read() throws IOException	**reads the next byte of data** from the input stream. It returns -1 at the end of the file.
2) public int available() throws IOException	returns an **estimate of the number of bytes** that can be read from the current input stream.
3) public void close() throws IOException	is used to close the current input stream.

Useful Method of OutputStream	Description
1) public void write(int) throws IOException	is used to **write a byte** to the current output stream.
2) public void write(byte[]) throws IOException	is used to **write an array of byte** to the current output stream.
3) public void flush() throws IOException	flushes the current output stream.
4) public void close() throws IOException	is used to close the current output stream.



Modifier and Type	Method	Description
abstract void	close()	It closes the stream and releases any system resources associated with it.
void	mark(int readAheadLimit)	It marks the present position in the stream.
boolean	markSupported()	It tells whether this stream supports the mark() operation.
int	read()	It reads a single character.
int	read(char[] cbuf)	It reads characters into an array.
abstract int	read(char[] cbuf, int off, int len)	It reads characters into a portion of an array.
int	read(CharBuffer target)	It attempts to read characters into the specified character buffer.
boolean	ready()	It tells whether this stream is ready to be read.
void	reset()	It resets the stream.
long	skip(long n)	It skips characters.



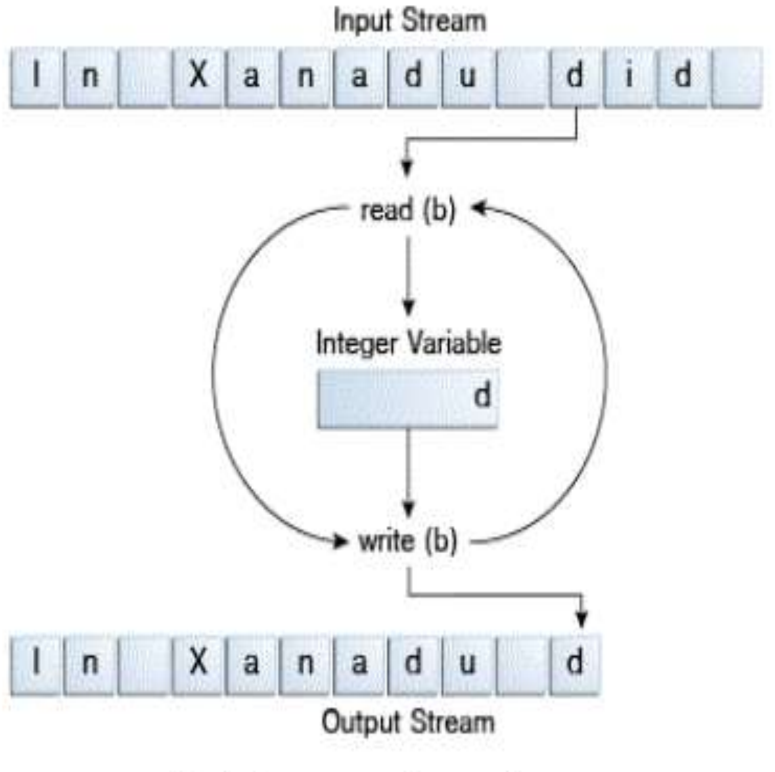
Modifier and Type	Method	Description
Writer	append(char c)	It appends the specified character to this writer.
Writer	append(CharSequence csq)	It appends the specified character sequence to this writer
Writer	append(CharSequence csq, int start, int end)	It appends a subsequence of the specified character sequence to this writer.
abstract void	close()	It closes the stream, flushing it first.
abstract void	flush()	It flushes the stream.
void	write(char[] cbuf)	It writes an array of characters.
abstract void	write(char[] cbuf, int off, int len)	It writes a portion of an array of characters.
void	write(int c)	It writes a single character.
void	write(String str)	It writes a string.
void	write(String str, int off, int len)	It writes a portion of a string.

Byte Based (FileInputStream & FileOutputStream)

Java FileInputStream class obtains input bytes from a file.
Java FileOutputStream is an output stream used for writing data to a file.

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

   public static void main(String args[]) throws IOException {
      InputStream in = null;
      OutputStream 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();
         }
      }
   }
}
```

Pastikan selalu close() stream saat selesai digunakan !!!



Simple byte stream input and output.

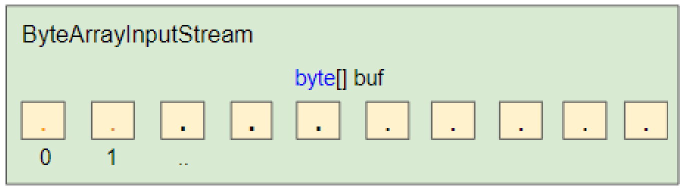
Byte Based (ByteArrayInputStream & ByteArrayOutputStream)

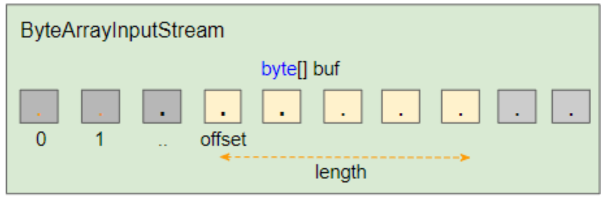
ByteArrayInputStream converting input stream to a byte array.

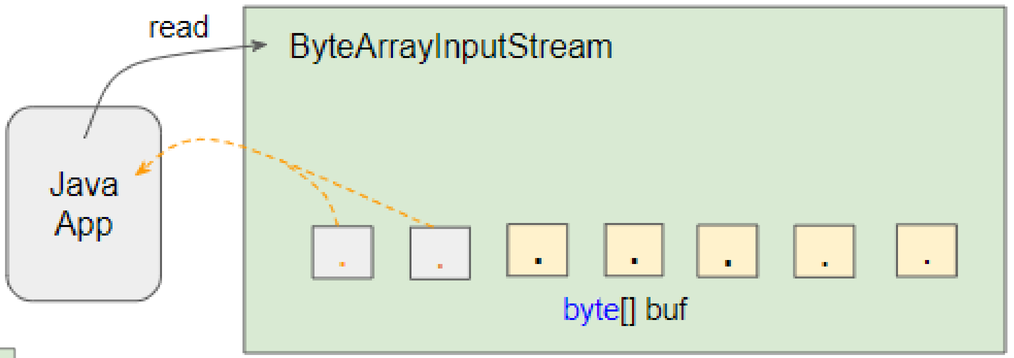
ByteArrayInputStream constructors

ByteArrayInputStream(byte[] buf)

ByteArrayInputStream(byte[] buf, int offset, int length)





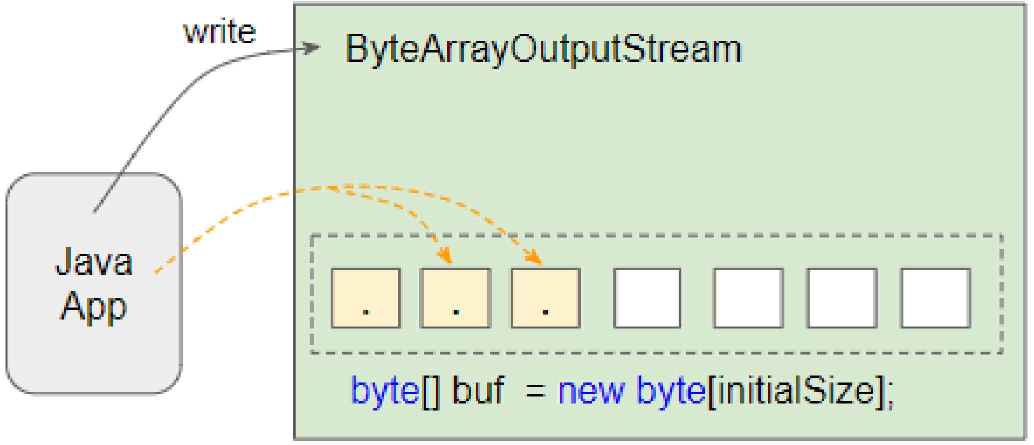


ByteArrayOutputStream converting output stream to a byte array.

Constructors

public ByteArrayOutputStream()

public ByteArrayOutputStream(int initialSize)



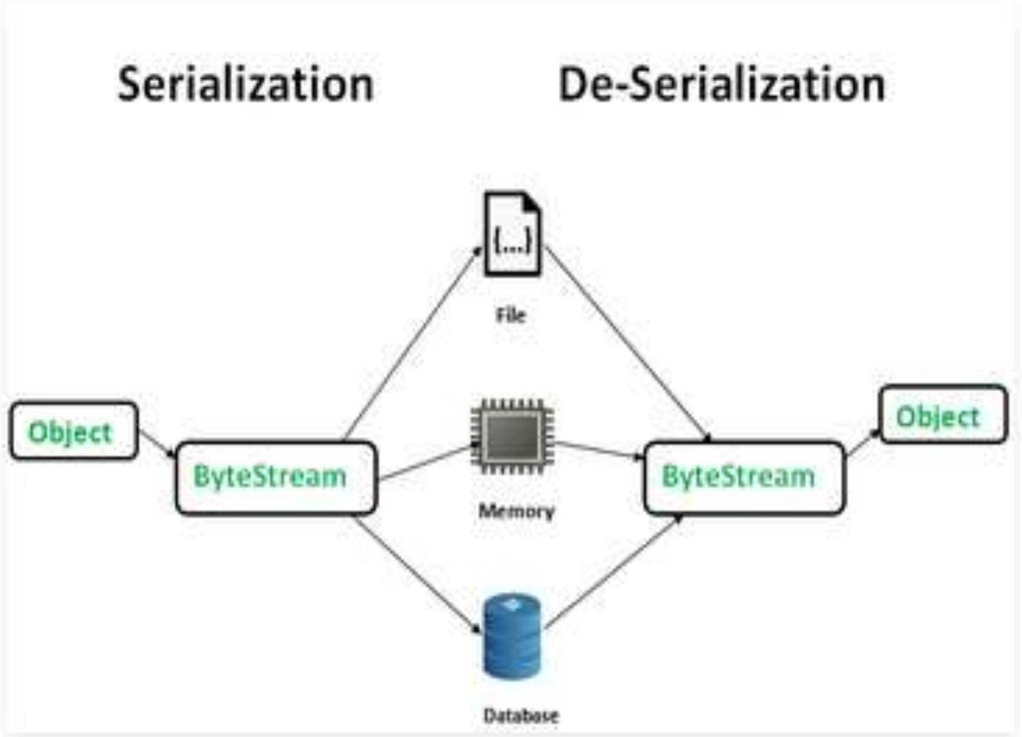
- ByteArrayOutputStream(int) constructor creates a ByteArrayOutputStream object with a byte array that has a specified initial size.
- ByteArrayOutputStream() constructor creates a ByteArrayOutputStream object with a byte array of that has default size (32).

Source : https://o7planning.org/13531/java-bytearrayinputstream

Byte Based (ObjectInputStream & ObjectOutputStream)

For Better Official Statistics

Serialization is **the conversion of an object to a series of bytes,**
so that the object can be easily saved to persistent storage or streamed across a communication link.



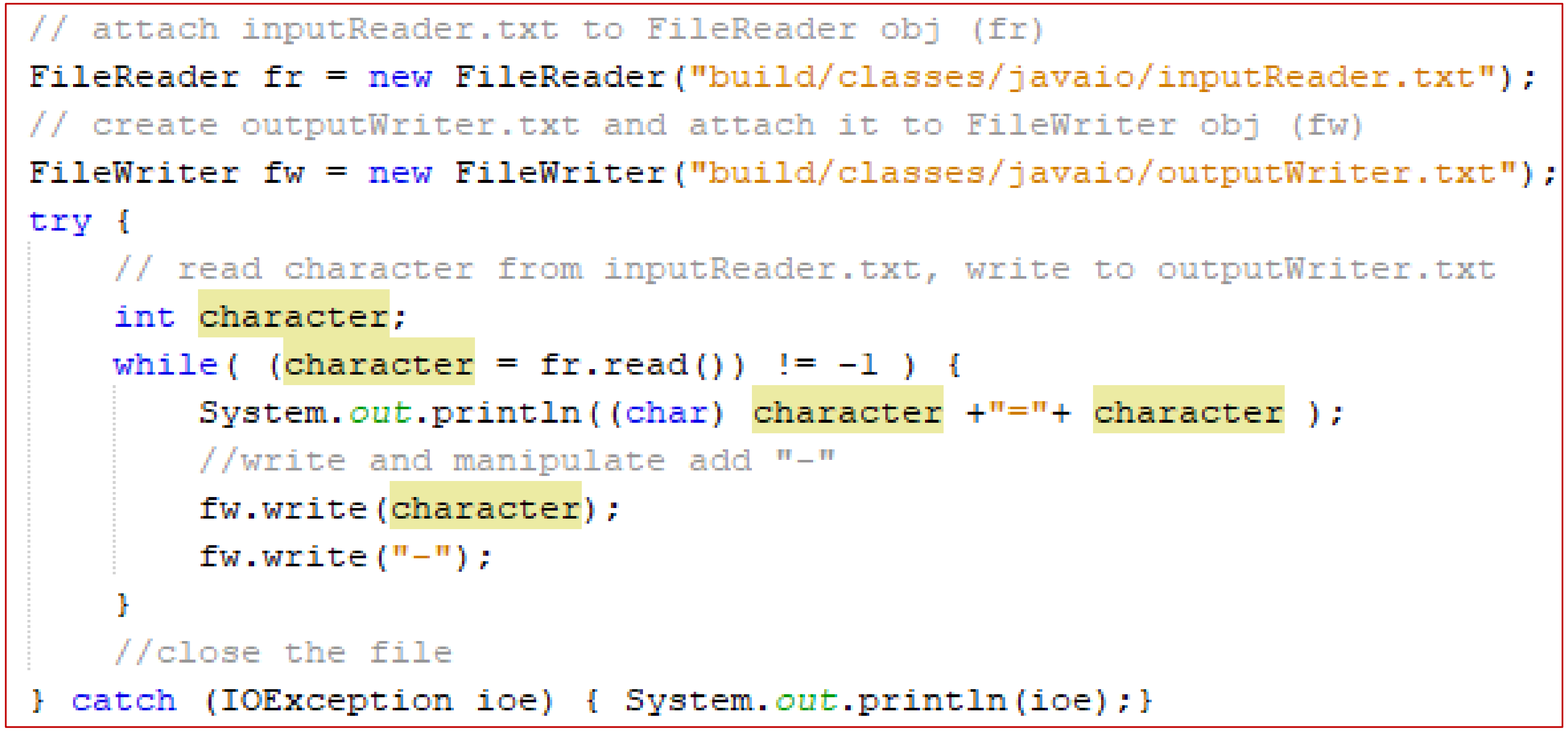
```java
// Helper class implementing Serializable interface
class Student implements Serializable {
    // Private class member variables
    private static final long serialVersionUID = -1438960132000208485L;
    private String name;
    private int age;

    public Student(String name, int age){
        this.name = name;
        this.age = age;
    }
    // Getters and Setter for class
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public int getAge() { return age; }
    public void setAge(int age) { this.age = age; }
    // Override toString method
    @Override public String toString(){
        // Simply return the name and age
        return "Student [name=" + name + ", age=" + age + "]";
    }
}
```

Source : https://www.geeksforgeeks.org/serializable-interface-in-java/

https://stackoverflow.com/questions/36719555/what-happens-if-we-dont-serialize-object-in-java

Character Based (FileReader & FileWriter)

Java FileWriter and FileReader classes are used to write and read data from text files (they are Character Stream classes). It is recommended **not** to use the FileInputStream and FileOutputStream classes if you have to read and write any textual information as these are Byte stream classes.

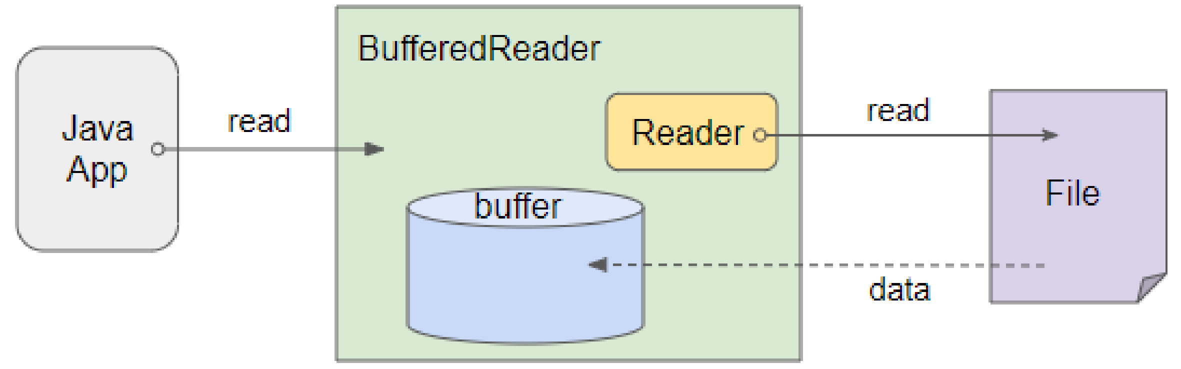


```java
// attach inputReader.txt to FileReader obj (fr)
FileReader fr = new FileReader("build/classes/javaio/inputReader.txt");
// create outputWriter.txt and attach it to FileWriter obj (fw)
FileWriter fw = new FileWriter("build/classes/javaio/outputWriter.txt");
try {
    // read character from inputReader.txt, write to outputWriter.txt
    int character;
    while( (character = fr.read()) != -1 ) {
        System.out.println((char) character +"="+ character );
        //write and manipulate add "-"
        fw.write(character);
        fw.write("-");
    }
    //close the file
} catch (IOException ioe) { System.out.println(ioe);}
```

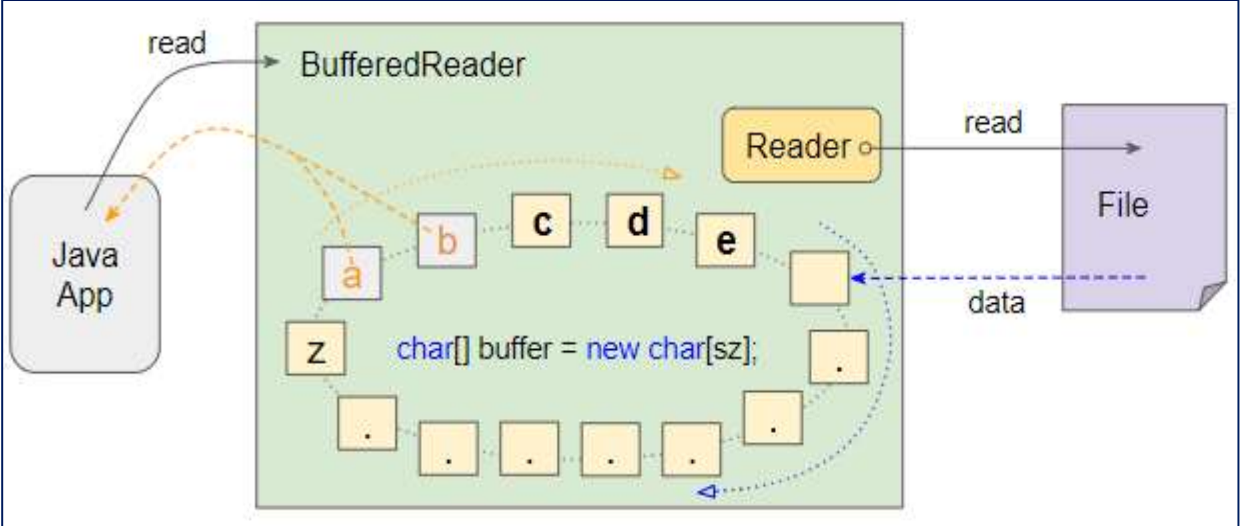
Character Based (BufferedReader)

BufferedReader is a subclass of **Reader**, which is used to simplify reading text from character input streams, and improve program performance.

BufferedReader operation principle looks like the following illustration:



Source : https://o7planning.org/13361/java-bufferedreader



```
// Create Reader to read a file.
Reader reader = new FileReader("/Volumes/Data/test/test.txt");

// Create a BufferedReader with default buffer array size of 8192 (16384 bytes = 16 KB).
BufferedReader br = new BufferedReader(reader);
```

BufferedReader wraps inside it a **Reader** object, which automatically reads data from the origin (such as file) and stores it in **BufferedReader**'s **buffer**.
BufferedReader also provides a **readLine()** method to read a line of text from **buffer.**

```
public class BufferedReaderEx1 {

  public static void main(String[] args) throws IOException {
    // Create Reader to read a file.
    Reader reader = new FileReader("/Volumes/Data/test/test.txt");

    // Create a BufferedReader with buffer array size of 16384 (32786 bytes = 32 KB).
    BufferedReader br = new BufferedReader(reader, 16384);

    String line = null;

    while((line = br.readLine())!= null) {
      System.out.println(line);
    }
    br.close();
  }
}
```

Java Tutorials:

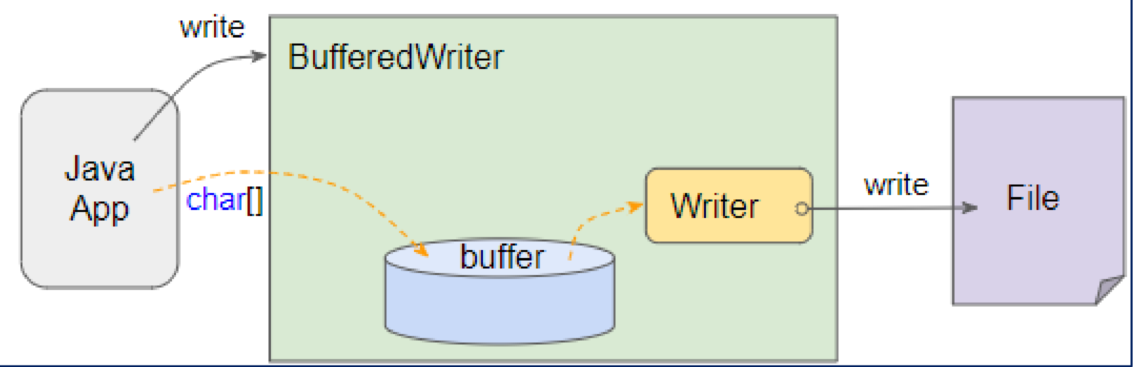
Java Reader Tutorial
Java Writer Tutorial

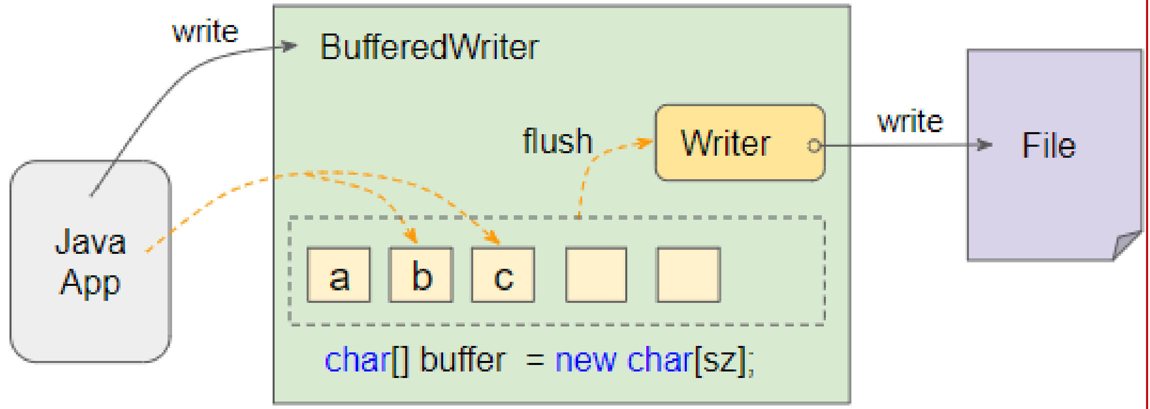
Character Based (BufferedWriter)

For Better Official Statistics

BufferedWriter is a subclass of **Writer**, which is used to simplify writing text to a character output stream, and improve program performance.

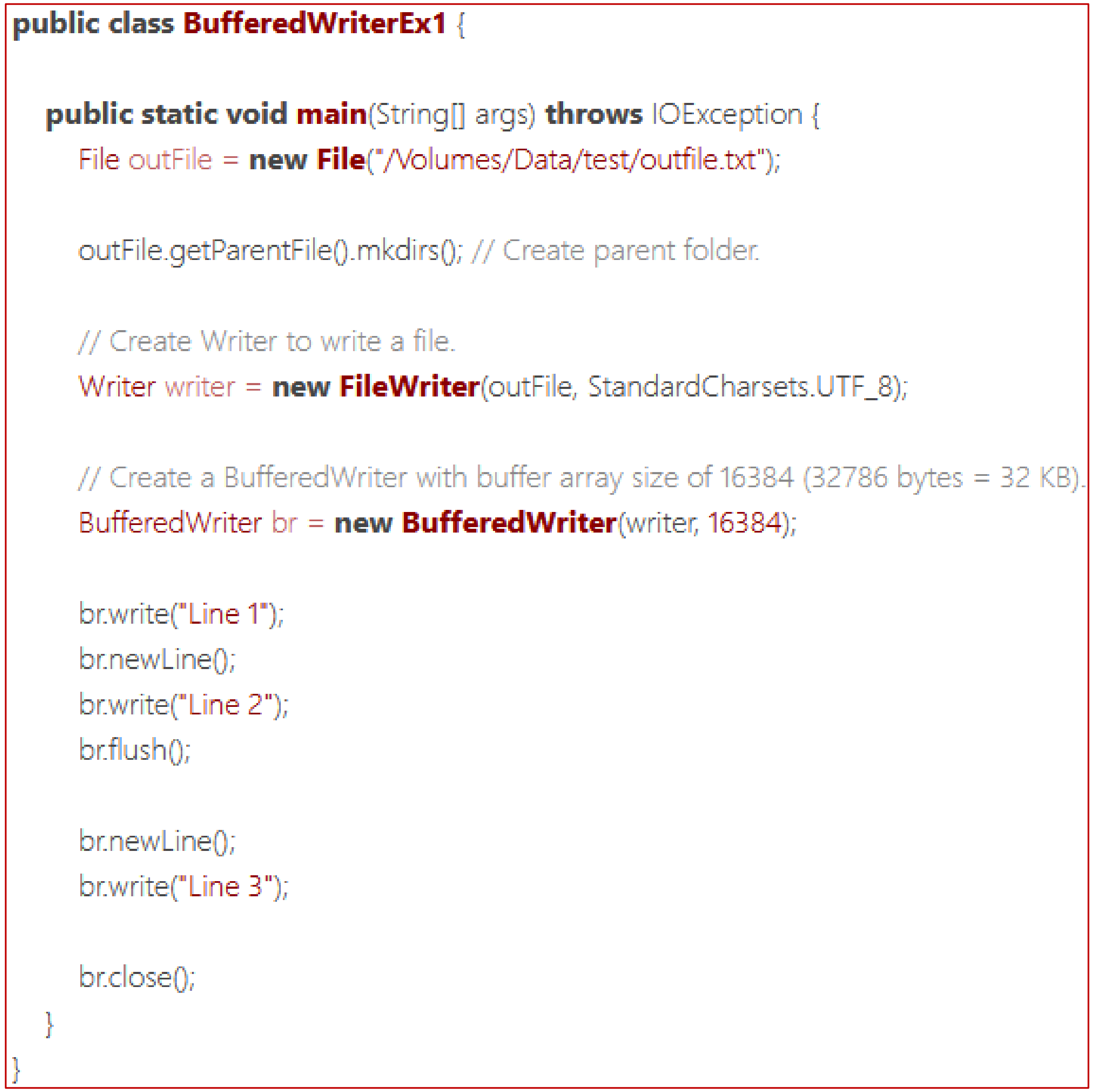
Operating principle of BufferedWriter looks like the following illustration:





```
File outFile = new File("/Volumes/Data/test/outfile.txt");
outFile.getParentFile().mkdirs(); // Create parent folder.

// Create Writer to write a file.
Writer writer = new FileWriter(outFile, StandardCharsets.UTF_8);

// Create a BufferedWriter with default buffer array size of 8192 (16384 bytes = 16 KB).
BufferedWriter br = new BufferedWriter(writer);
```

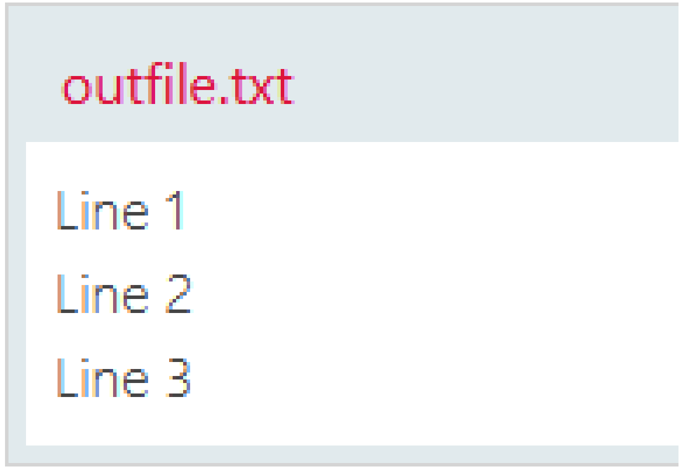
BufferedWriter wraps inside it a **Writer** object, which is responsible for writing data to the target (such as file). **BufferedReader** also provides a **write(String str)** and **newline()** methods.



```
public class BufferedWriterEx1 {

    public static void main(String[] args) throws IOException {
        File outFile = new File("/Volumes/Data/test/outfile.txt");

        outFile.getParentFile().mkdirs(); // Create parent folder.

        // Create Writer to write a file.
        Writer writer = new FileWriter(outFile, StandardCharsets.UTF_8);

        // Create a BufferedWriter with buffer array size of 16384 (32786 bytes = 32 KB).
        BufferedWriter br = new BufferedWriter(writer, 16384);

        br.write("Line 1");
        br.newLine();
        br.write("Line 2");
        br.flush();

        br.newLine();
        br.write("Line 3");

        br.close();
    }
}
```

https://o7planning.org/13363/java-bufferedwriter

Output:



outfile.txt

```
Line 1
Line 2
Line 3
```

Java I/O

- ✓ Terdapat dua tipe Java Stream I/O yaitu **Byte Stream (1 Byte/8 bits)** dan **Character Stream (2 Bytes/16 bits)**
- ✓ Terdapat banyak class pada package java.io sesuai kebutuhan penggunaannya :
  - ✓ Byte Stream Classes, beroperasi dengan mengalirkan Byte per Byte tiap waktu dan baik untuk memproses raw data seperti file biner, image, audio, objek, dsj.
  - ✓ Character Stream Classes, beroperasi dengan mengalirkan Character tiap waktu dan digunakan untuk memproses file teks terutama jika terdapat karakter Unicode 16 bit.
- ✓ Pastikan menutup close() stream setiap kali sudah tidak digunakan.
- ✓ Untuk melakukan operasi I/O pada objek pastikan class dari objek tersebut telah meng-implements marked interface Serializable.
- ✓ Untuk melakukan operasi I/O pada file gunakan class FileInputStream/FileOutputStream (byte based) dan FileReader/FileWriter (character based).
- ✓ Untuk meningkatkan performa baik saat melakukan proses read atau write bisa menggunakan Buffered Classes seperti BufferedInputStream/BufferedOutputStream untuk *byte based*, dan/atau BufferedReader/BufferedWriter untuk *character based*.
- ✓ Pastikan selalu menggunakan Exception Handling yang sesuai saat melakukan operasi stream I/O

Terima Kasih

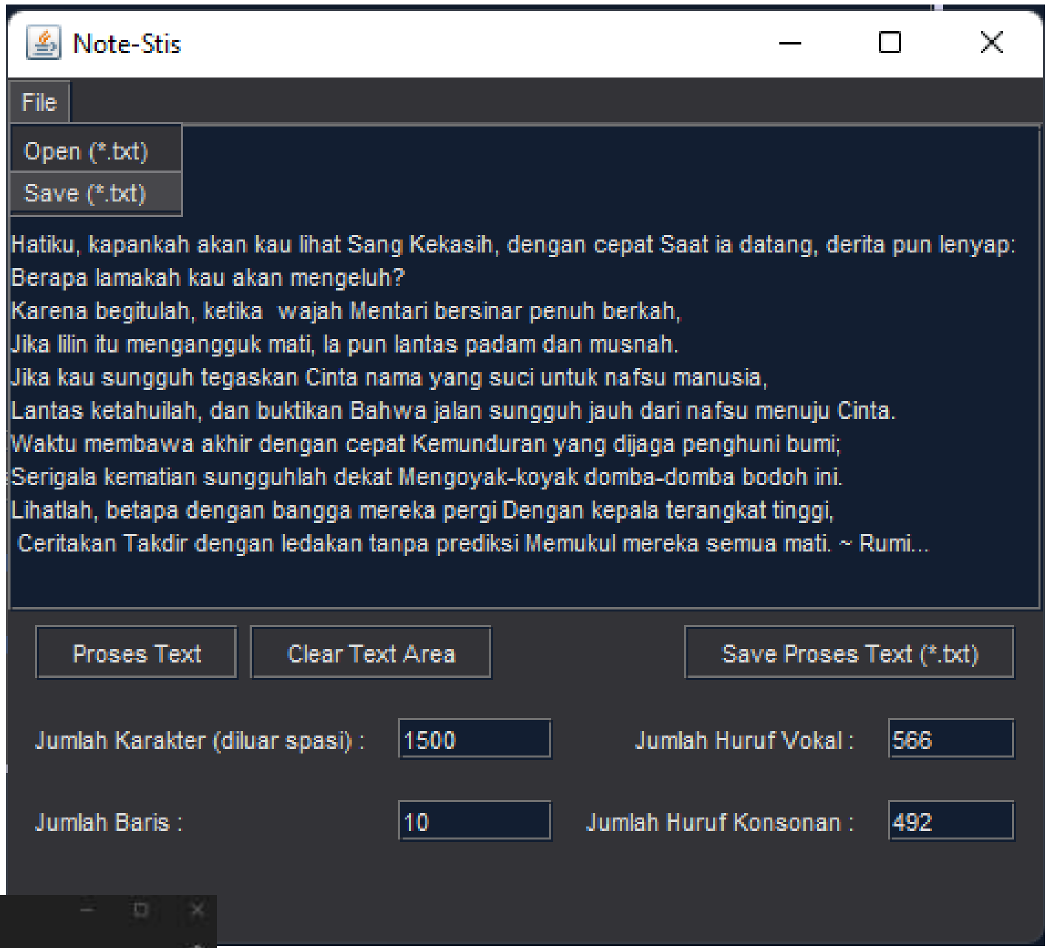
Tugas Pertemuan 10

Buatlah aplikasi GUI pengolah kata dengan fitur sebagai Berikut :

1. Aplikasi memiliki menubar (JMenuBar) dengan menu item open dan save file dari dan ke penyimpanan lokal di computer. Keduanya bisa digunakan untuk file berektensi *.txt. Konten Hasil file yang disave harus sama formatnya seperti yang dituliskan pada editor (spasi, baris, dsb).

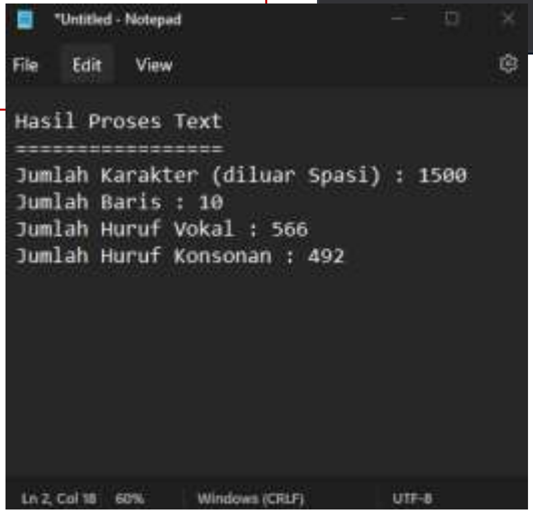
2. Terdapat sebuah editor JTextArea untuk mengetikkan karakter text (dgn scroll vertical aktif tetapi tidak dengan scroll horizontal)

3. Terdapat tombol "Proses Text" untuk melakukan perhitungan jumlah karakter (diluar spasi), jumlah baris, jumlah huruf vocal, dan jumlah huruf konsonan dari teks yang dituliskan pada editor. Tombol "Clear Text Area" untuk menghapus semua teks pada editor, dan tombol "Save Proses Text (*.txt)" yang digunakan untuk menyimpan hasil proses text ke dalam sebuah file berekstensi *.txt.



```
Hatiku, kapankah akan kau lihat Sang Kekasih, dengan cepat Saat ia datang, derita pun lenyap:
Berapa lamakah kau akan mengeluh?
Karena begitulah, ketika  wajah Mentari bersinar penuh berkah,
Jika lilin itu mengangguk mati, Ia pun lantas padam dan musnah.
Jika kau sungguh tegaskan Cinta nama yang suci untuk nafsu manusia,
Lantas ketahuilah, dan buktikan Bahwa jalan sungguh jauh dari nafsu menuju Cinta.
Waktu membawa akhir dengan cepat Kemunduran yang dijaga penghuni bumi;
Serigala kematian sungguhlah dekat Mengoyak-koyak domba-domba bodoh ini.
Lihatlah, betapa dengan bangga mereka pergi Dengan kepala terangkat tinggi,
 Ceritakan Takdir dengan ledakan tanpa prediksi Memukul mereka semua mati. ~ Rumi...
```

Jumlah Karakter (diluar spasi) : 1500
Jumlah Huruf Vokal : 566
Jumlah Baris : 10
Jumlah Huruf Konsonan : 492



```
Hasil Proses Text
=================
Jumlah Karakter (diluar Spasi) : 1500
Jumlah Baris : 10
Jumlah Huruf Vokal : 566
Jumlah Huruf Konsonan : 492
```