Java Orientado a Objetos
Lendo e Escrevendo Arquivos

Console I/O

```
*************** CUNIT CONSOLE - MAIN MENU ***********************
(R)un all, (S)elect suite, (L)ist suites, Show (F)ailures, (Q)uit
Enter Command : r

Running Suite : Suite_success
        Running test : successful_test_1
        Running test : successful_test_2
        Running test : successful_test_3
WARNING - Suite initialization failed for Suite_init_failure.
Running Suite : Suite_clean_failure
        Running test : successful_test_4
        Running test : failed_test_2
        Running test : successful_test_1
WARNING - Suite cleanup failed for Suite_clean_failure.
Running Suite : Suite_mixed
        Running test : successful_test_2
        Running test : failed_test_4
        Running test : failed_test_2
        Running test : successful_test_4

--Run Summary: Type      Total     Ran  Passed  Failed
               suites        4       3     n/a       2
               tests        13      10       7       3
               asserts      10      10       7       3

*************** CUNIT CONSOLE - MAIN MENU ***********************
(R)un all, (S)elect suite, (L)ist suites, Show (F)ailures, (Q)uit
Enter Command :
```

```java
import java.io.Console;

public class Teste {

    Console con = System.console();
}
```

System.in

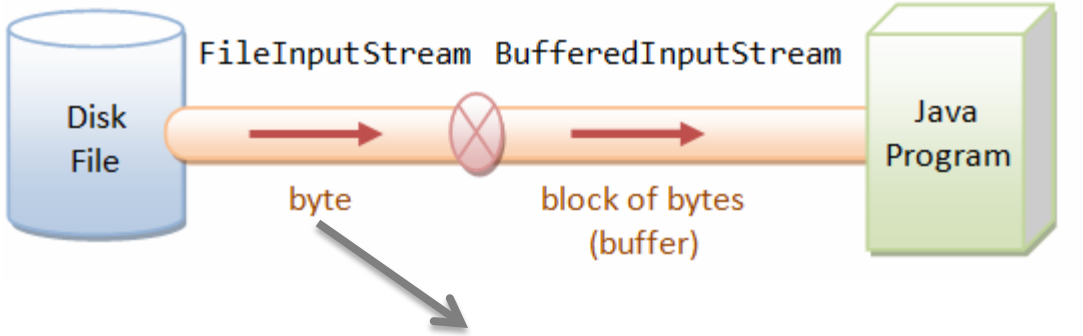
System.out

Usando a classe Scanner

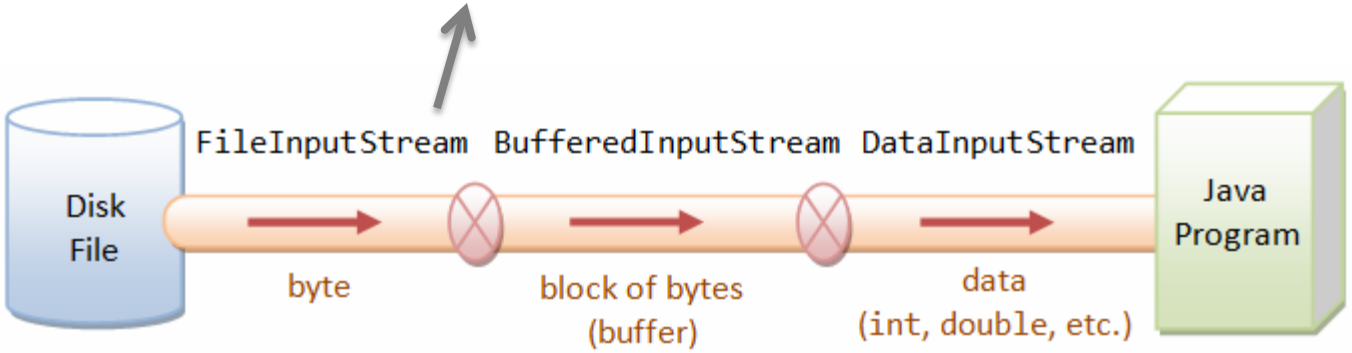
Scanner engloba diversos métodos para facilitar a entrada de dados

```
public static void main(String[] args) {

    Scanner scan = new Scanner(System.in);
    String teste = scan.nextLine();
    System.out.println("palavra digitada: " + teste);
}
```

Encadeando I/O Stream




```
InputStream bis = new BufferedInputStream(new FileInputStream(new File("file.zip")));

InputStream dis = new DataInputStream(new BufferedInputStream(new FileInputStream(new File("file.dat"))));
```



FileWriter e BufferedWriter

```
public static void main(String[] args) {

    try {

        File arquivo = new File("C:\\teste.txt");

        FileWriter fw = new FileWriter(arquivo);

        BufferedWriter bw = new BufferedWriter(fw);

        bw.write("Texto a ser escrito no txt");
        bw.newLine();
        bw.write("Quebra de linha");

        bw.close();

        fw.close();

    } catch (IOException e) {
        System.out.println("Arquivo não existe!");
    }

}
```

Um arquivo, caminho absoluto

Encadeando para FileWriter

Encadeando para BufferedWriter

FileReader e BufferedReader

```
public static void main(String[] args) {

    try {

        File arquivo = new File("C:\\teste.txt");

        FileReader fr = new FileReader(arquivo);

        BufferedReader br = new BufferedReader(fr);

        while (br.ready()) {
            String linha = br.readLine();
            System.out.println(linha);
        }

        br.close();
        fr.close();

    } catch (FileNotFoundException e) {
        System.out.println("Arquivo não existe!");
    } catch (IOException e) {
        System.out.println("Erro ao ler arquivo!");
    }

}
```

Um arquivo, caminho absoluto

Encadeando para FileReader

Encadeando para BufferedReader

NIO.2 - Path

Path pode ser um arquivo no diretório atual, caminho relativo ao programa

Path pode ser um arquivo, caminho absoluto, no windows use caractere de escape '\'

```
Path p1 = Paths.get("in.txt");

Path p2 = Paths.get("c:\\projetos\\java\\Hello.java");

Path p3 = Paths.get("/use/local");
```

Path pode ser um diretório

JAVA 7 NIO.2, mais simples, Buffered, sem bloqueio de IO. A classe **java.nio.Path**, especifica a localização de um arquivo, ou diretório, ou link simbólico. Substitui **java.io.File**

NIO.2 – I/O Streams

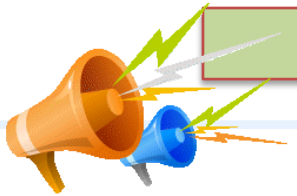
```
String fileStr = "small_file.txt";
Path path = Paths.get(fileStr);
List<String> lines = new ArrayList<String>();
lines.add("Oi,您好! Olá,吃饱了没有?");

try {
   Files.write(path, lines, Charset.forName("UTF-8"));
} catch (IOException ex) {
   ex.printStackTrace();
}

byte[] bytes;
try {
   bytes = Files.readAllBytes(path);
   for (byte aByte: bytes) {
      System.out.printf("%02X ", aByte);
   }
   System.out.printf("%n%n");
} catch (IOException ex) { }

List<String> inLines;
try {
   inLines = Files.readAllLines(path, Charset.forName("UTF-8"));
   for (String aLine: inLines) {
      for (int i = 0; i < aLine.length(); ++i) {
         char charOut = aLine.charAt(i);
         System.out.printf("[%d]'%c'(%04X) ", (i+1), charOut, (int)charOut);
      }
      System.out.println();
   }
} catch (IOException ex) {}
```

Um arquivo no diretório atual, caminho relativo ao programa

Escreve dados para arquivo texto

Lê dados do arquivo como bytes

Lê dados do arquivo como caracteres UTF-8



Use com arquivos pequenos

NIO.2 – I/O Streams

```
InputStream in = Files.newInputStream(path);

OutputStream out = Files.newOutputStream(path);

Reader reader = Files.newBufferedReader(path);

Writer writer = Files.newBufferedWriter(path);
```

Compatibilidade com Java I/O Básico

```
try (OutputStream out = new BufferedOutputStream(
  Files.newOutputStream(p,StandardOpenOption.CREATE,
          StandardOpenOption.APPEND))) {
  out.write(data, 0, data.length);
} catch (IOException x) {
  System.err.println(x);
}
```

Leitura ou escrita orientada por streams, um caractere por vez.

Use com arquivos Grandes, para usar streams compativeis com java IO

NIO.2 – I/O Channel

```
private void leia(Path path) {

    try (SeekableByteChannel sbc = Files.newByteChannel(path)) {

        ByteBuffer buf = ByteBuffer.allocate(64);

        while (sbc.read(buf) > 0) {
            buf.rewind();
            System.out.print(Charset.forName("UTF-8").decode(buf));
            buf.flip();
        }

    } catch (IOException e) {
        log.warning(e.toString());
    }
}
```

I/O de arquivos conectados a um channel, dados são lidos para o buffer

O método rewind() muda o ponteiro para inicio do buffer e o deixa pronto para leitura

O método flip() muda o ponteiro e permite ler dados a partir do buffer, use antes de escrever para arquivo

Use com arquivos Grandes. Channel lê um buffer por vez