howest
hogeschool

Object Oriented Architectures and Secure Development

(File) Input and Output (IO)

Matthias Blomme
Mattias De Wael
Frédéric Vlummens

Standard IO

```
Scanner in = new Scanner(System.in);
System.out.println("What is your name?");
String name = in.nextLine();
System.out.println("Hello " + name);
```

```
What is your name?
>
```

Standard IO

```
Scanner in = new Scanner(System.in);
System.out.println(“What is your name?”);
String name = in.nextLine();
System.out.println(“Hello ” + name);
```

```
What is your name?
> Alice
```


howest hogeschool

Standard IO

```
Scanner in = new Scanner(System.in);
System.out.println("What is your name?");
String name = in.nextLine();
System.out.println("Hello " + name);
```

```
What is your name?
> Alice
Hello Alice
```


howest hogeschool

Standard IO

```
Scanner in = new Scanner(System.in);
System.out.println("What is your name?");
String name = in.nextLine();
System.out.println("Hello " + name);
```

What are `System.in` and `System.out`?

howest hogeschool

PrintStream

```
System.out.println(“Hello World”);
```

- This is plain Java.
- Classes, objects, and methods.

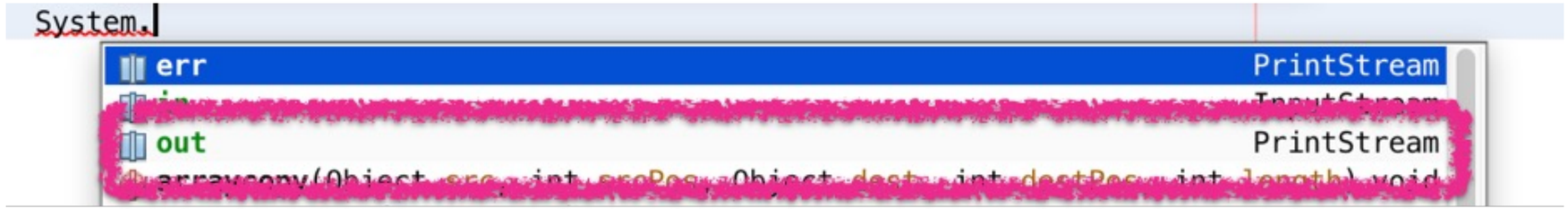
howest hogeschool

PrintStream

```
System.out.println("Hello World");
```

class field (object) method argument (String)

All objects have a type




```
PrintStream ps = System.out;
ps.println("Hello World");
```

PrintStream

```
public void foo( PrintStream ps ) {
    ps.println("Hello World");
}

public void bar() {
    foo( System.out );
}
```

PrintStream to Standard Output

```
private void run() {
    Product p1 = new Product("smartphone", 599);
    printProduct(System.out, p1);
}

private void printProduct(PrintStream ps, Product p)
{
    ps.printf("Name:\t%s%n", p.getName());
    ps.printf("Price:\t%.2f%n", p.getPrice());
}
```

PrintStream to Error

```
private void run() {
    Product p1 = new Product("smartphone", 599);
    printProduct(System.err, p1);
}

private void printProduct(PrintStream ps, Product p)
{
    ps.printf("Name:\t%s%n", p.getName());
    ps.printf("Price:\t%.2f%n", p.getPrice());
}
```

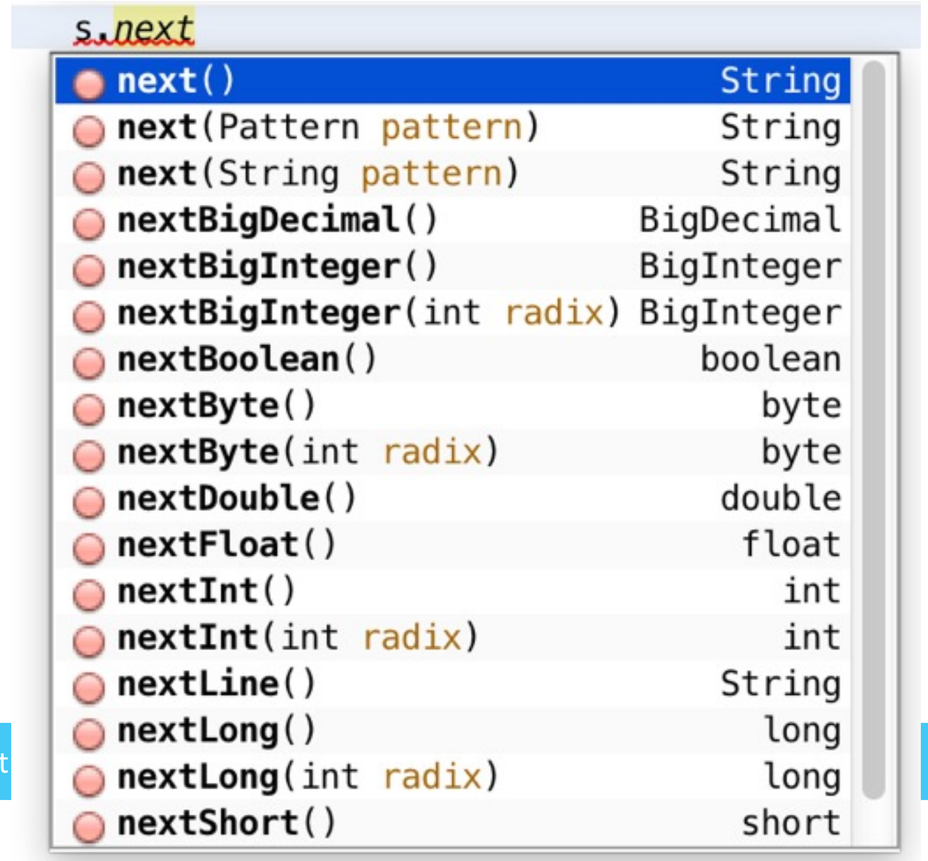
Printf and String.format *(need more control? Search the internet!)*

```
System.err.printf("Name:\t%s%nPrice:\t%.2f%n",
      p.getName(),
      p.getPrice()
);
```

%s	any	String, or implicitly object.toString()
%d	Simple number	[byte, short, int, long]
%f	Decimal Number	[float, double]
%4d	Simple Number	Occupies 4 positions
%.2f	Decimal Number	Always 2 numbers after the decimal point
%n	none	OS independent newline

Scanner

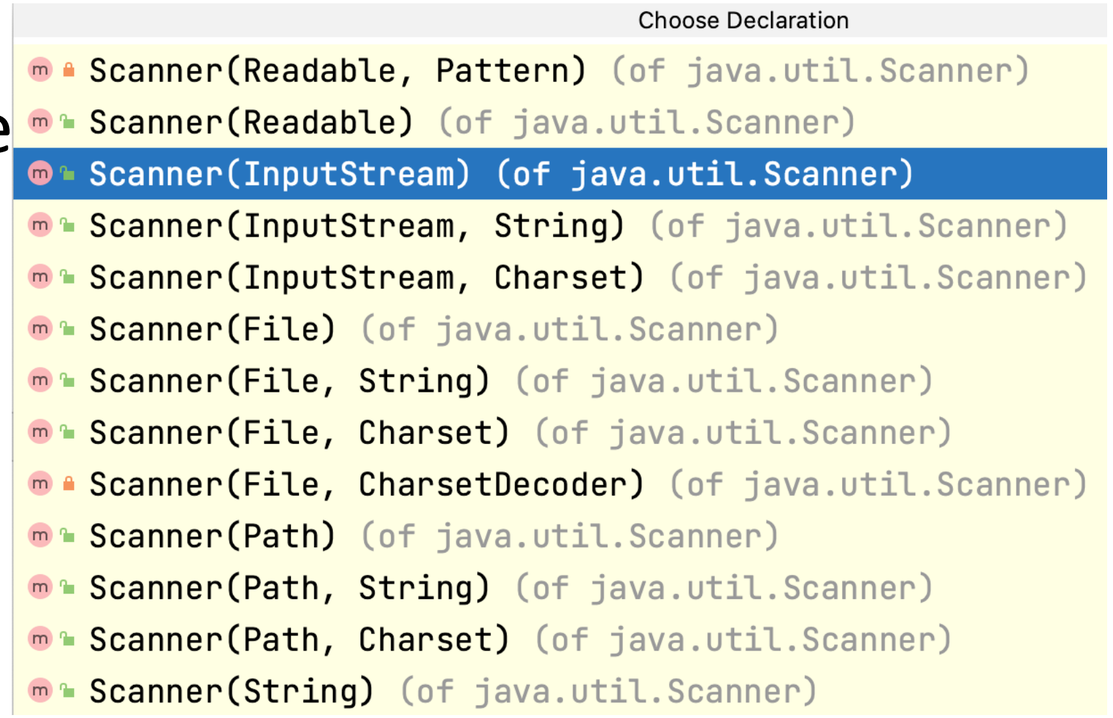
```
Scanner in = new Scanner( System.in );
while( in.hasNextLine() ) {
    System.out.println( in.nextLine() );
}
```



howest hogeschool

Scanner

```
Scanner in = new Scanner( System.in );
while( in.hasNextLine() ) {
    System.out.println( in.ne
}
```



Standard IO

Both `System.in` and `System.out` are IO-streams.
(not to be confused with the stream-API of map filter and reduce!)

They represent the command-line from the System. If we can link them to the data of a file we can replace Standard IO with file IO.

PrintStream to File

```
private void run() {
    Product p1 = new Product("smartphone", 599);
    printProduct(System.err, p1);

    printProduct(new PrintStream("product.txt"), p1);
    printProduct(new PrintStream("/product.txt"), p1);
    printProduct(new PrintStream("./product.txt"), p1);

}

private void printProduct(PrintStream ps, Product p) {
    ps.printf("Name:\t%s%n", p.getName());
    ps.printf("Price:\t%.2f%n", p.getPrice());
}
```

Figure out where Java stores the file on your HD.

howest hogeschool

PrintStream to File

```
String fileName = "test.txt";
PrintStream ps = new PrintStream("test.txt");
```

```
public PrintStream(String fileName) throws FileNotFoundException {
    this(false, new FileOutputStream(fileName));
}
```

```
public FileOutputStream(String name) throws FileNotFoundException {
    this(name != null ? new File(name) : null, false);
}
```

PrintStream to File

```
PrintStream ps = new PrintStream(
    "test.txt"
);
```

Just the file name …

howest hogeschool

PrintStream to File

```
PrintStream ps = new PrintStream(
    new FileOutputStream(
        "test.txt"
    )
);
```

New file outputstream (with filename)

howest hogeschool

PrintStream to File

```
PrintStream ps = new PrintStream(
    new FileOutputStream(
        new File("test.txt")
    )
);
```

Pass a file to the file outputstream

PrintStream to File

```
PrintStream ps = new PrintStream(
    new FileOutputStream(
        new File("test.txt")
    )
);
```

Different versions, to give the developer more or less control (default vs configuration).

PrintStream to File

```
PrintStream ps = new PrintStream(
    new FileOutputStream(
        new File("test.txt")
    )
);
```

Create: add data to file, if it exists, clear the file first.

PrintStream to File

```
PrintStream ps = new PrintStream(
    new FileOutputStream(
        "test.txt"
    )
);
```

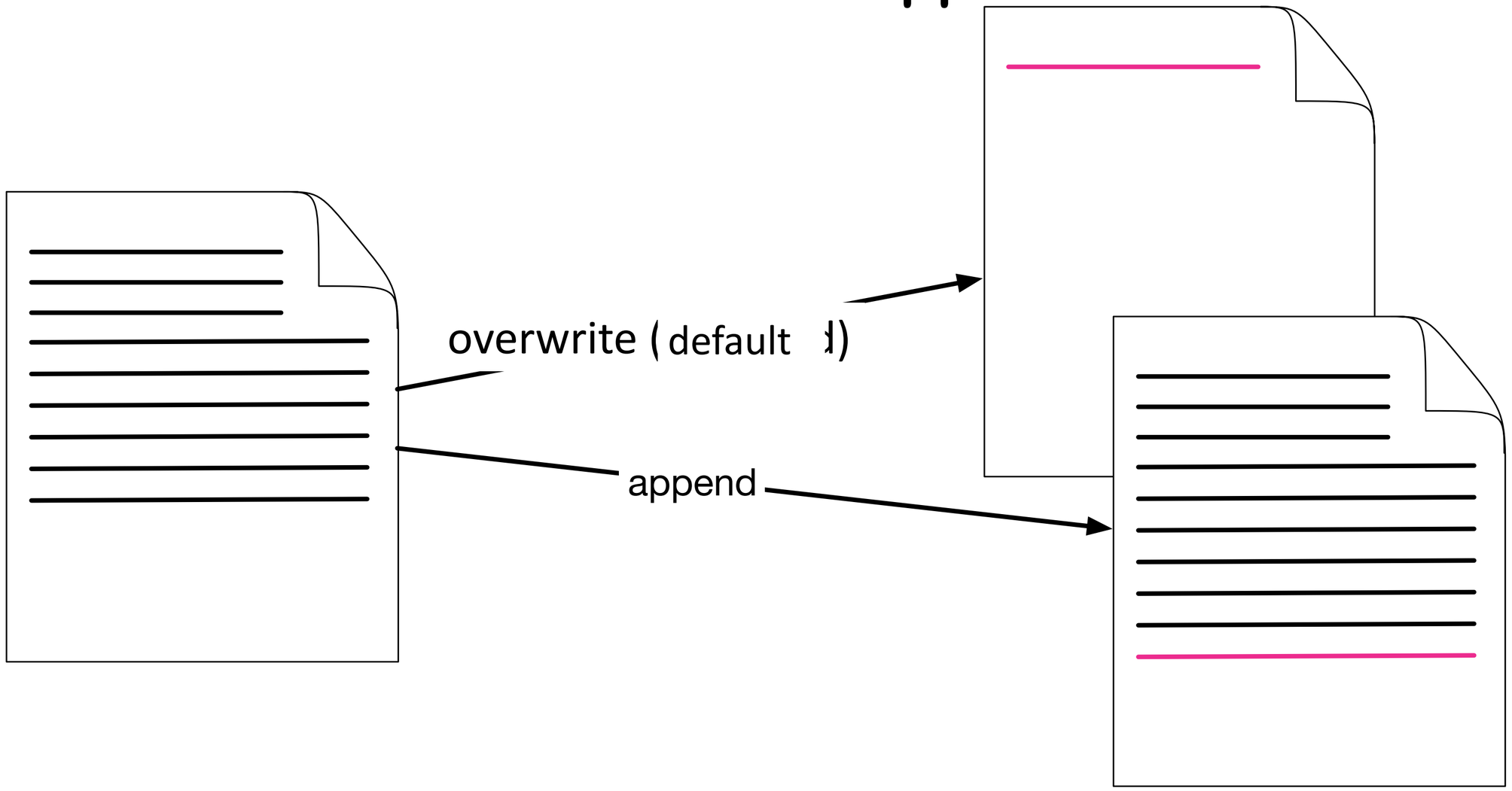
Create: add data to file, if it exists, clear the file first.

PrintStream to File: append

```
PrintStream ps = new PrintStream(
    new FileOutputStream(
        "test.txt", true
    )
);
```

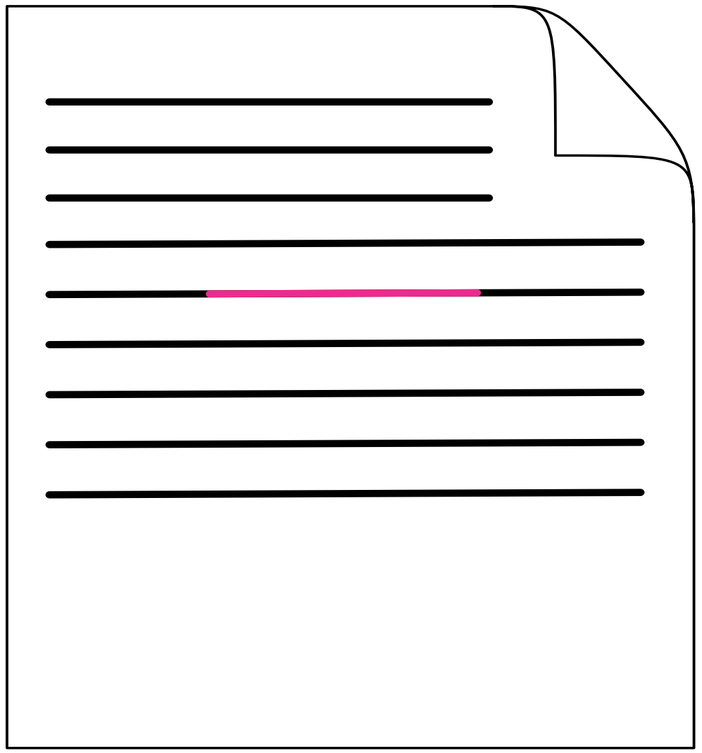
Append: default is false, when true data is added at the end of the file instead of “clearing” the file first.

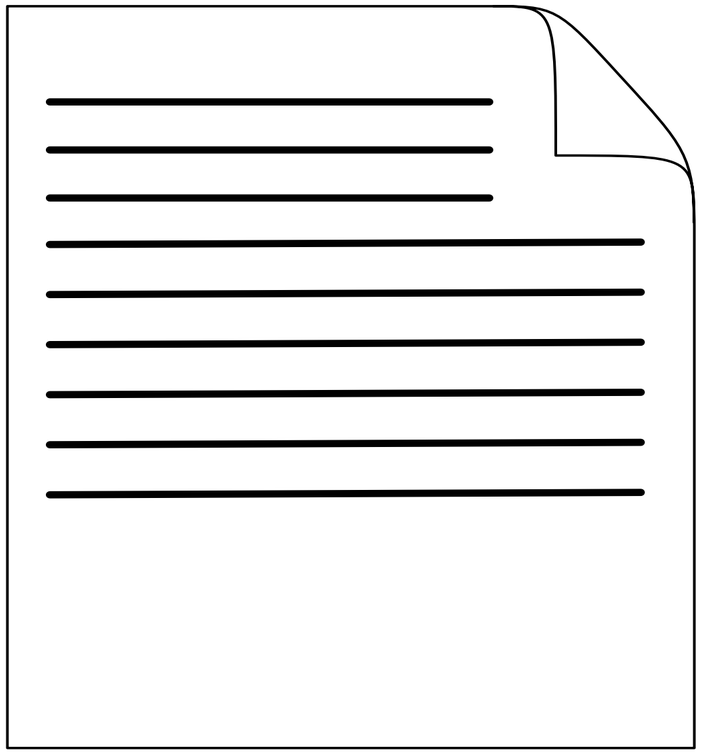
PrintStream to File: create versus append

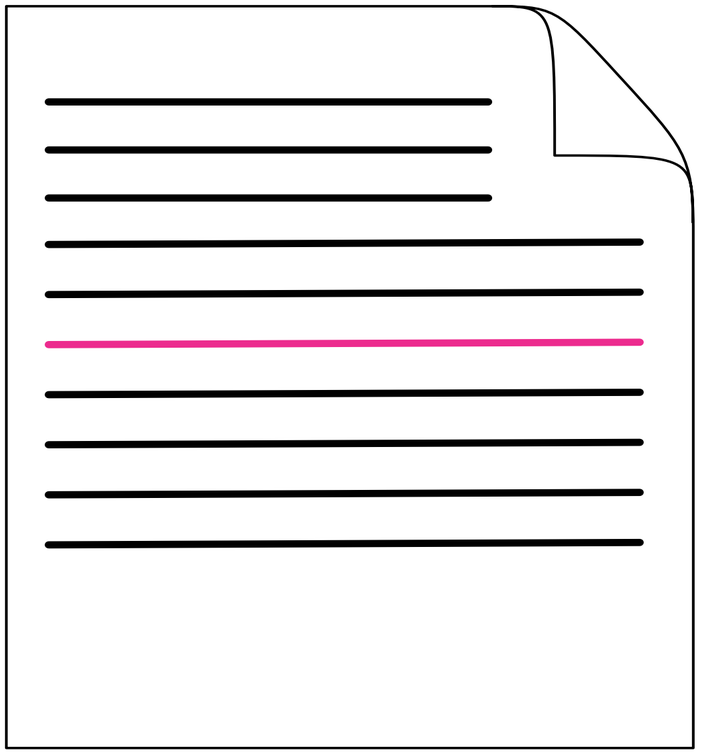


howest hogeschool

Random access







We do not study random access in files

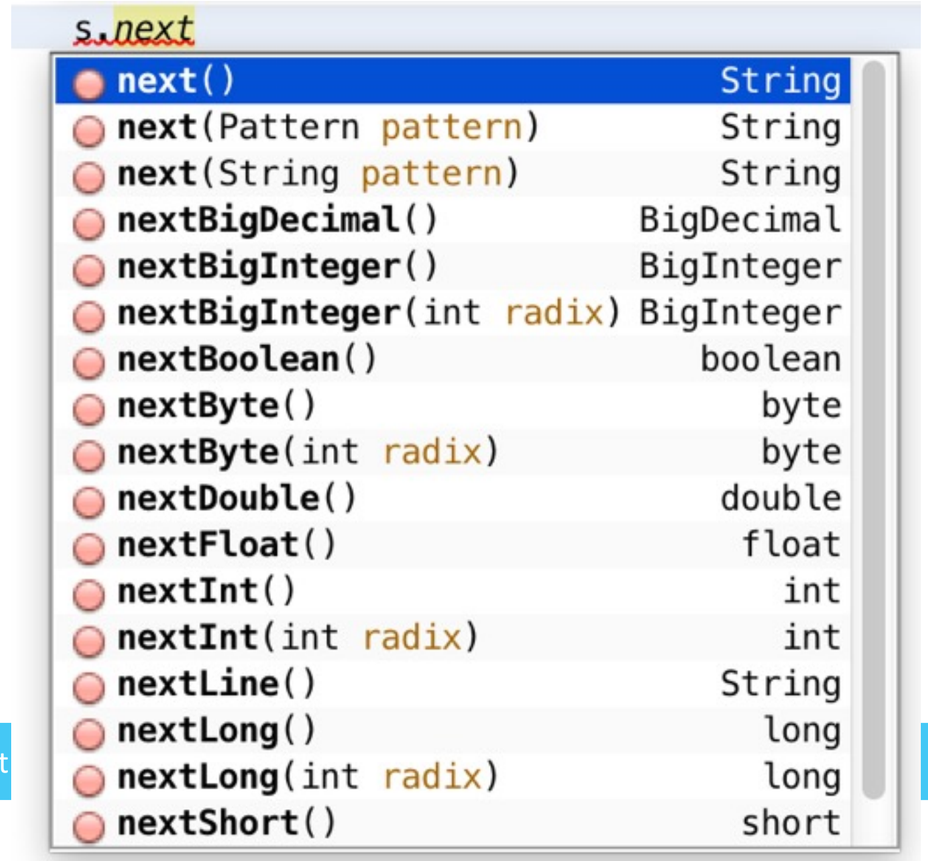
https://lmgtfy.com/?q=java+file+random+access

howest hogeschool

Scanner from File

```
Scanner s = new Scanner( new File("test.txt") );
while( s.hasNext() ) {
    System.out.println( s.next() );
}
```

```
s.hasNextLine()
s.nextLine()
```



howest hogeschool

IO operations tend to go wrong sometimes …

```
try {
    Scanner s = new Scanner( new File("test.txt") );
    while( s.hasNext() ) {
        System.out.println(s.next());
    }
} catch (FileNotFoundException e) {
    // handle it !
}
```

IO operations tend to go wrong sometimes …

```
try {
    PrintStream ps = new PrintStream(new FileOutputStream(
                new File("test.txt")
    ));
    ps.println("I have so much content!");
} catch (FileNotFoundException e) {
    // handle it !
}
```


howest hogeschool

IO operations tend to allocate resources (close!!!) …

```
try {
    Scanner s = new Scanner( new File("test.txt") );
    while( s.hasNext() ) {
        System.out.println(s.next());
    }
    s.close(); // NOT SAFE (enough) !!!
} catch (FileNotFoundException e) {
    // handle it !
}
```

IO operations tend to allocate resources (close!!!) …

```
try {
    PrintStream ps = new PrintStream(new FileOutputStream(
                new File("test.txt")
    ));
    ps.println("I have so much content!");
    ps.close(); // NOT SAFE (enough) !!!
} catch (FileNotFoundException e) {
    // handle it !
}
```


howest hogeschool

IO operations tend to allocate resources (close!!!) …

```
try (Scanner s = new Scanner( new File("test.txt") )) {
    while( s.hasNext() ) {
        System.out.println(s.next());
    }
} catch (FileNotFoundException e) {
    // handle it !
}
```

IO operations tend to allocate resources (close!!!) …

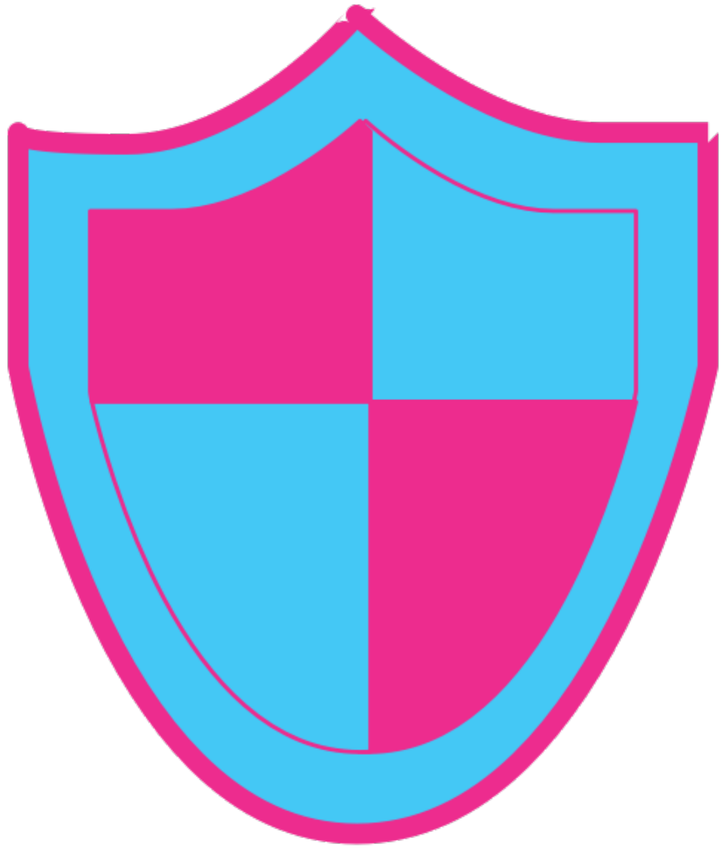
```
try (
    PrintStream ps = new PrintStream(new FileOutputStream(
            new File("test.txt")
        )
)) {
    ps.println("I have so much content!");
} catch (FileNotFoundException e) {
    // handle it !
}
```

Guideline 1-2: Release resources in all cases

https://www.oracle.com/technetwork/java/seccodeguide-139067.html#1-2

```
try (  /* put your resources here */  ) {
    /* put your computations here */
} catch (FileNotFoundException e) {
   // handle it !
}
```

Some objects, such as open files, locks and manually allocated memory, behave as resources which require every acquire operation to be paired with a definite release. It is easy to overlook the vast possibilities for executions paths when exceptions are thrown. Resources should always be released promptly no matter what.



howest hogeschool

The File class

https://docs.oracle.com/javase/7/docs/api/java/io/File.html

The File-class represents a file or directory on your hard-disk.

The class contains a lot of handy methods.

howest hogeschool