Building Blocks

Commands

Calling java on file with java extension

java directly on .java file, no need to call javac.

```
$ java HelloWorld.java
```

And no *HelloWorld.class* being created!

packages

From project root folder

```
$ javac  -d target/classes  ./src/main/java/org/enricogiurin/ocp17/

$ java -cp target/classes org.enricogiurin.ocp17.ch1.HelloWorld
```

We have to specify all the single packages if present.

Alternative cp

```
$ java -cp target/classes org.enricogiurin.ocp17.ch1.HelloWorld
$ java -classpath target/classes org.enricogiurin.ocp17.ch1.HelloWo
$ java --class-path target/classes org.enricogiurin.ocp17.ch1.Hello
```

Variables

- Local variables **need** to be initialized before to be used;
- Instance variables **do not need** to be initialized before to be used.

Variable Names

```
    int $ = 5;
    int € = 66;
    int €$_ = 55;
    int £€$ = 55;
```

Possible names for variables

Variables scope

- **Local variables**: In scope from declaration to the end of the block
- **Method parameters**: In scope for the duration of the method
- **Instance variables**: In scope from declaration until the object is eligible for garbage collection
- **Class variables**: In scope from declaration until the program ends

var

use of var

var can be only used with local variables, not as an instance variable.

```
class WrongVar {
    var x = "hello";  //does NOT compile!
    void local() {
        var y = "ciao"; //does compile!
    }

//does not compile: var can only be used as local variable
   int varAsArgument(var x){
        return 0;  //does NOT compile!
    }
}
```

var not a reserved key

```
  public void var() {
    var var = "var";
  }
```

var initialization

```
var x;  //does not compile!
x = 5;
```

This compiles as **var** is not a reserved key in java

var compound declaration

```
var x=5, y=6; //DOES NOT COMPILE
```

Usage of var

Garbage Collection

```
System.gc();
```

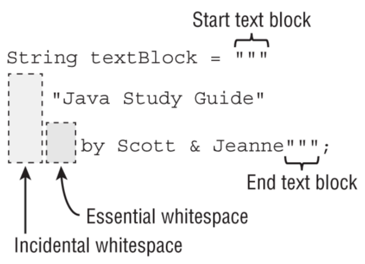
In Java, there are no guarantees about when garbage collection will run. The JVM is free to ignore calls to System.gc()

(apparent) conflict of class names

```
import org.enricogiurin.ocp17.ch1.fruits.Apple;
import org.enricogiurin.ocp17.ch1.phones.*;
```

In this case will be used class from the package *fruits* as:
importing by class name takes precedence over wildcards!

Text Blocks


```
String textBlock = """
  "Java Study Guide"
    by Scott & Jeanne""";
```

Incidental whitespace just happens to be there to make the code easier to read. You can reformat your code and change the amount of incidental whitespace without any impact on your String value.

Text Blocks

```
String tb = """
    Hello
    World""";
```

the code within the `"""` and `"""` is just text.
text blocks require a break between beginning and the end.

Imagine a vertical line drawn on the leftmost non-whitespace character in your text block. Everything to the left of it is **incidental whitespace**, and everything to the right is **essential whitespace**.

Example

```
String s = """aaa"""; //does not compile
```

Trailing whitespace: *spazi bianchi finali* (IT)

```
jshell> var text = """
   ...> John is a good guy\
   ...>  and he's my friend""";
text ==> "John is a good guy and he's my friend"
```

Remember that a backslash (****) means to skip the line break.

```
    String s = """
        Hello \
        World
        """;
    System.out.println(s);  //Hello World
```

Escape sequences

There are two special escape sequences for Text Blocks. These allow fine-grained control of the processing of line breaks and whitespaces: `\` (followed by a line break) and `\s`.