

# Fussy things of syntax that you will have to know about

## REQUIREMENTS

Anything in between `/*` and `*/` is taken as a comment and not viewed to be java source code.

Anything following `//` and up to the end of the line is also taken as a comment.

Java "statements" are of two types.

Simple Java statements end with a semicolon, as in the simple statement

```
celsius = (fahrenheit - 32.0) * 5.0 / 9.0;
```

Compound Java statements look like

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

and have text followed by an open and close brace that defines the limits of the compound statement, just as parentheses work in algebra or in English text.

In general, Java consists of "symbols" separated by "white space". Viewed this way, any amounts of white space are equivalent to a single space. Thus the line

```
celsius = (fahrenheit - 32.0) * 5.0 / 9.0;
```

could also be written

```
celsius=(fahrenheit-32.0)*5.0/9.0;
```

or also could be written

```
celsius =                                (fahrenheit - 32.0)
* 5.0 /                                     9.0;
```

and Java would not care. (We who grade your papers do care, however; see below on style.)

However, the line

```
celsius = (fahr enheit - 32.0) * 5.0 / 9.0;
```

would have Java looking at TWO symbols "**fahr**" and "**enheit**" and not one symbol "**fahrenheit**" and this would be an error.

Java is case sensitive: the symbol "**celsius**" is entirely different from the symbol "**Celsius**".

There are a couple of dozen "reserved words" in Java that cannot be used as symbols except in the precise meaning in which they are defined by Java. These reserved words include the symbols "**public**", "**class**", "**static**", "**void**", "**main**".

Each yadayada.java file should have one Java "class" in it, as in

```
public class TempConversion1
{
}
```

that declares the name of the class and then encloses the Java for that class inside braces.

Note that in the second program you have to "**import**" (also a reserved word) the **Scanner** class.

This is something you will be doing a lot in many of your programs.

## CONVENTIONS AND "REQUIREMENTS" OF GOOD STYLE

Your programs should always have names and dates in the initial comments.

Use enough white space to make the text readable.

Use symbols that are meaningful.

You will see some Java written as

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

and some written as

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

I use the former because I find that easier to read, but there is no clear standard on this. Choose something that you yourself are comfortable with, and stick with it at least inside a single program.

Indentation is good and is mandatory as part of good style. Eclipse will indent with a tab character. I personally prefer two spaces, as in

```
public class TempConversion1
{
    public static void main(String[] args)
    {
    }
}
```

Either is ok. I find that with tabs it is too easy to have so much indentation there's not enough space on the line for a full line, and it's annoying to have to go set the editor to use fewer spaces.

**VERY IMPORTANT:** Eclipse will "help" you with indentation, spell checking, semicolon checking, brace closings, etc. This is good, but it is not something that you should rely on to the extent that you don't actually know what is correct, because then you will have a harder time on the exams.

I also tend to put the top line of a compound statement at the end of the statement as a comment, as in

```
public class TempConversion1
{
    public static void main(String[] args)
    {
    }
} // public class TempConversion1
```

This for you is optional. Losing track of open and close braces is a major constant irritating mistake that we all make, and I find that writing code the way I do helps to minimize the number of such errors I make. If the compound statement gets long enough that it might scroll off one screenful of code, then having the identifier on the closing brace helps me to know where I am in the code.