Compiler

Your Java program, in a bunch of `.java` files like `Animal.java` and `A2.java`, cannot be executed or run in that form. A program is needed to put it in a form that can be executed. That program is called a *compiler*.

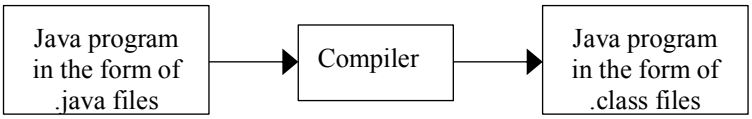
The Java compiler has two tasks:

1. Check that the program is syntactically correct —it follows the grammar rules of the Java Language Specification. For example, the assignment statement to the right is syntactically incorrect because a **`float`** value may not be stored in an **`int`** variable, and the expression under it is syntactically incorrect because it is missing a right parenthesis.

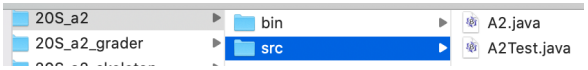
```
int k= 2.3;
3* (2 + 4
```

2. If the program is syntactically correct, translate into a form that can be executed: Translate each `.java` file, say a file `C.java`, into the machine language[1] and store it in a corresponding file, `C.class`. These `.class` files are what can be executed, or run.

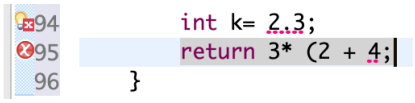
The diagram to the right depicts the process. Of course, the compiler produces a `.class` file only if the `.java` file is syntactically correct.



Where are the `.class` files placed? Below to the left, we show the contents of the hard drive on our computer for an Eclipse project named `20S_a2`. Directory `20S_a2` contains two subdirectories. Subdirectory `src` has been selected, and it contains two `.java`. files. To the right, we show the same directory except that subdirectory `bin` has been selected; in it are the corresponding `.class` files. Spend some time looking at the hard drive on your own computer to see the `.java` and `.class` files.



The process of compiling the program happens so fast that you don't notice it. In fact, as you type lines into a `.java` file in Eclipse, the compiler is continually checking for correctness. And, Eclipse tells you if there is a syntax error. For example, to the right, we show part of a `.java` file that is being edited in the Eclipse window. The two red X's —actually white X's in a red circle— tell you that those lines have syntax errors in them. Hover your mouse over a red X and a small window opens, telling you what the syntax error is.



Thus, if you see a red X, you know the `.java` file that is being edited can't be compiled, or translated, into the machine language. But if there are no red X's, then the compiler that Eclipse uses has already compiled the class into a `.class` file. Oooh! It's fast.

---

[1] The situation is a bit more complicated. Yes, some compilers translate directly into the machine language. But Java translates the classes into its own language, called the *Java Virtual Machine Language*. It's *virtual* because no real computer or machine exists with that machine language. Java created and described that virtual machine language so that, not matter what computer a Java program was compiled on, the result would always be in the same language. This provided a sense of universality that was unheard of at the time (early 1990's). C compilers, for example, translated to whatever machine the compiler was on, and a C program could possibly produce different results depending on what computer it was run on. Not so with a Java program. Because of this, there is another program that *interprets*, or executes the program written in the Java Virtual Machine Language.