main()

Running a program in Java takes two steps. The javac program takes a class file and compiles it (and any other classes it refers to). The java program then runs this compiled code through an interpreter. If you do this yourself at the command line it looks like this:

```
javac Student.java
java Student
```

The system expects the class you are running to contain a method called main( ).

The main( ) method must be declared both *public* and *static*. It should be public because you are running it outside its class. When a variable or method is declared static, that means there is only one copy for the class and it can be accessed without creating an object of the class. If you had a class *Person* and wanted to have a variable *population* for the class that was incremented every time you construct a new object of the class, you would want that variable to be declared static. You could refer to that variable as Person.population; you don't need an object of the class for this.

Similarly, if you tell java to run class Person, the system will call

Person.main( )

The full signature of the main( ) method is

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

The parameter *args* means that main can be given an array of String arguments. For example, if you want program PrintFile to work with file foobar.txt you might run the program with

```
java PrintFile foobar.txt
```

The string "foobar.txt" would then be args[0] in main( ).

We will primarily run programs inside Eclipse. The command-line arguments are specified in the Run/RunConfigurations window.

The fact that main( ) must be a static method sometimes gives Java beginners fits. Static means it can be used outside the class without an object of the class. main( ) can only directly call methods of the class that are themselves declared static.

So if we have a class Primes that has a main( ) method that prints out a list of primes, main( ) could call a method

```
static boolean isPrime(int x)
```

to determine if number x is prime. This method can be declared static because you don't need any properties of the Primes class to decide if x is prime. main() can't call isPrime() if it isn't declared static.

On the other hand, if you have a class Student that has properties such as name, age, id, gpa, classList, etc for a student, a method such as setAge( ) could not be declared static. This method would be setting the age property of a specific student object.

main( ) in this case, would work by creating Student objects. It sometimes surprises beginners to Java that methods of a class such as Student can themselves construct Student objects.