Commands and your first Program

1. Commands

A *program* is a set (list) of commands. First, the first command is executed, then the second, the third, and so on. When all the commands have been executed, the program ends.

The specific commands that are allowed in the list depend on *who is performing the command*, i.e. what commands does the *performer* know or understand. You can command a dog to "Sit" or "Speak", a cat to "Shoo", a human to "Stop! Or I'll shoot!", and a robot to "Work! Work, you robot scum!"

Programs written in the Java language are executed by the *Java virtual machine (JVM)*. The *JVM* is a special program that can execute programs written in the Java language.

The list of commands it knows is quite extensive.
For example, the following *command* tells the JVM to display `Robots are friends to humans`:

```
System.out.println("Robots are friends to humans");
```

The simplest command

But we won't start with commands. Instead, let's begin with a couple of simple principles. *Knowledge of a few principles replaces knowledge of many facts.*

Principle 1: in Java, it is customary to write each command on a new line. A *semicolon* goes at the end of each command.

Let's say we want to *display* the phrase `Robots are friends to humans` 3 times. This is how the code would look:

```
System.out.println("Robots are friends to humans");
System.out.println("Robots are friends to humans");
System.out.println("Robots are friends to humans");
```

A program consisting of three commands

Principle 2: a program cannot consist of only commands. Java commands must be inside functions, and functions must be inside classes.

Imagine a sofa. A sofa cannot exist on its own. It exists in a room somewhere. And a room also cannot exist on its own. A room is located in some house. Or, you could say that the house is divided into rooms, and those rooms contain things.

So, commands are like furniture. In the Java programming language, a command cannot be by itself: it is part of a function (in Java, functions are also called methods). A *method* (function) is part of a *class*. In other words, a *class* is divided into *methods*,
and *methods* contain *commands*.

Java programs are made up of classes, classes contain methods, and methods contain commands.

---

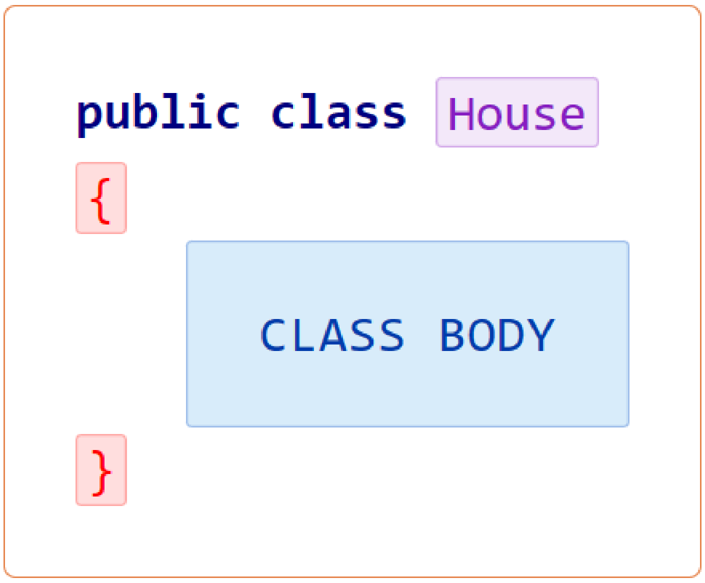
2. Structure of a typical program

Java programs are composed of *classes*. There might be tens of thousands of classes. The smallest program has a single class. For each class, we create a separate file whose name coincides with the name of the class.

Let's say you decide to create a class to represent a house. Then you need to create a `House` class contained in a `House.java` file.

Now suppose you want to represent a cat in your program. You need to create a `Cat.java` file and define the `Cat` class in it, and so on.

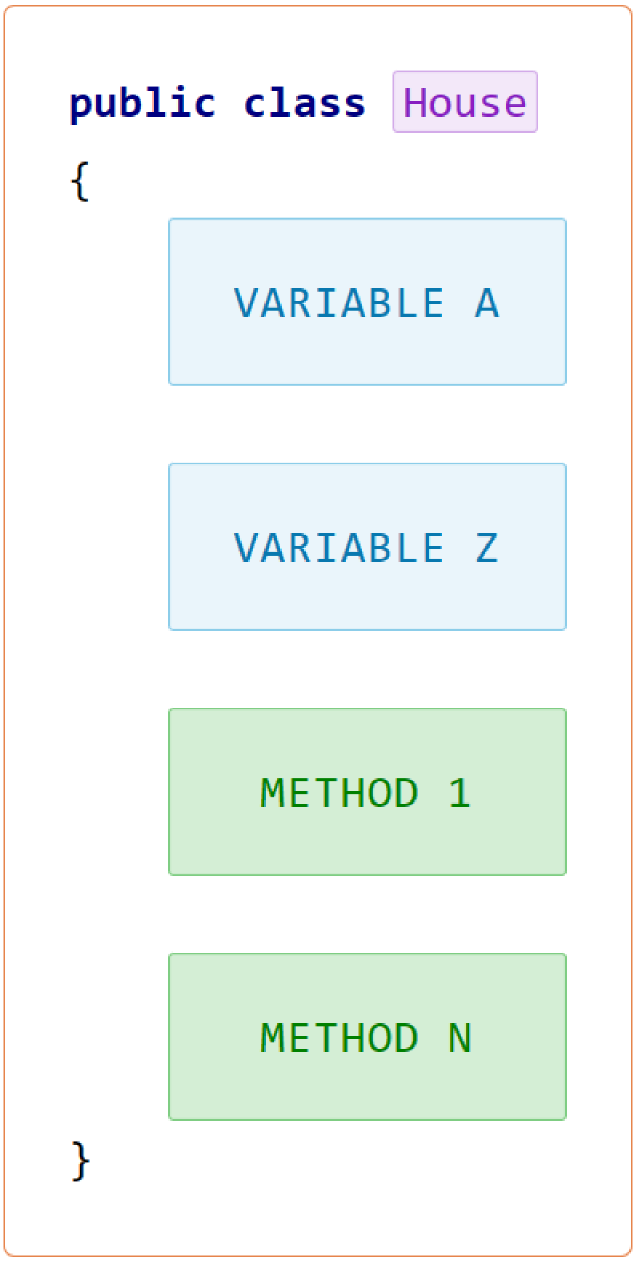
The files contain text — code written in the **Java programming language**. The code of a class usually consists of a *class name* and a *class body*. The *body of a class* is enclosed in *curly braces*. This is what the `House` class might look like:



```
public class House
{
        CLASS BODY
}
```

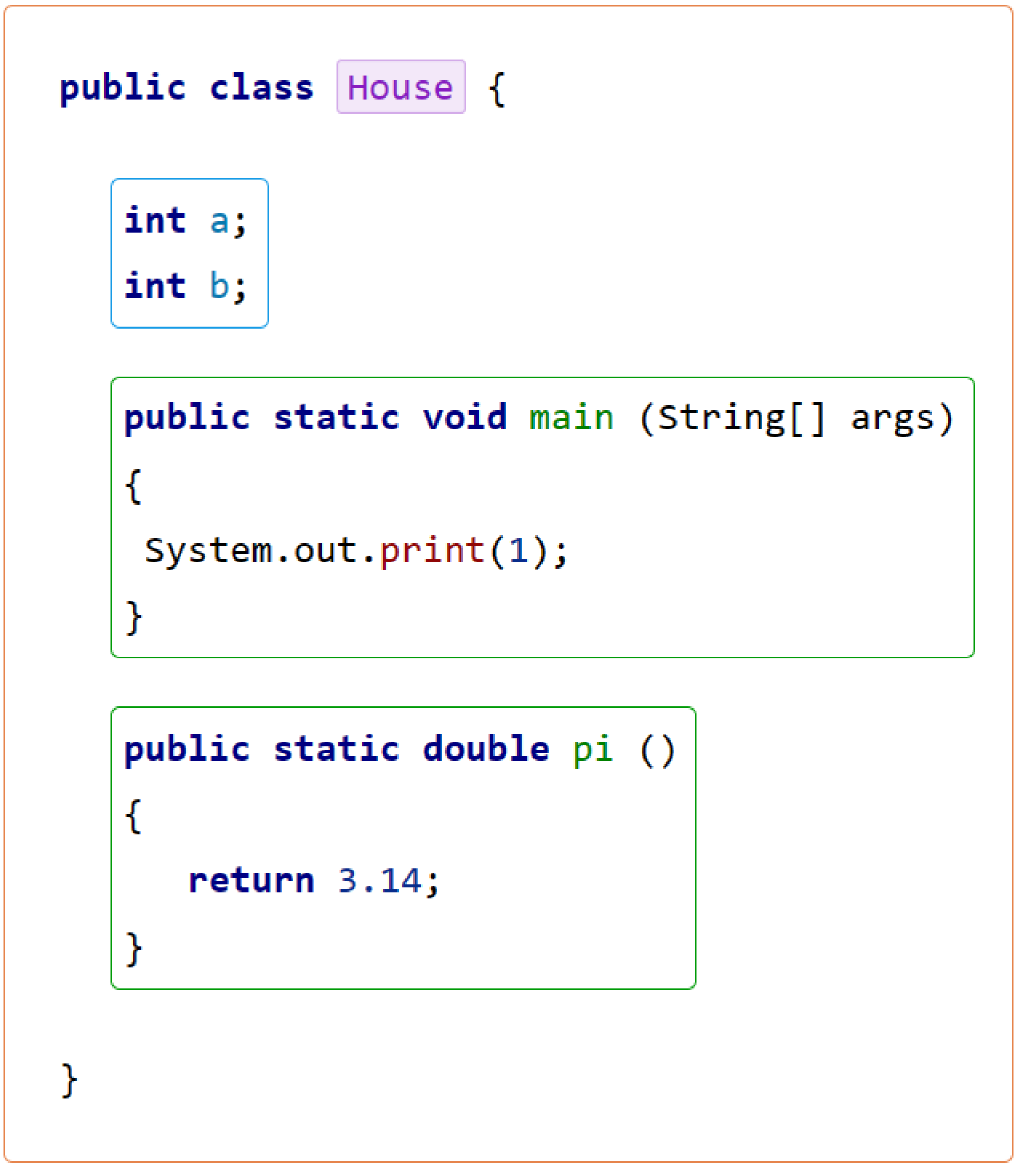
`House.java` *file*

The body of the class can contain *variables* (also called fields) and *methods* (functions). It looks something like this:



`House.java` *file*

And here's a specific example:



```
public class House {

    int a;
    int b;

    public static void main (String[] args)
    {
     System.out.print(1);
    }

    public static double pi ()
    {
       return 3.14;
    }

}
```

`House.java` *file*

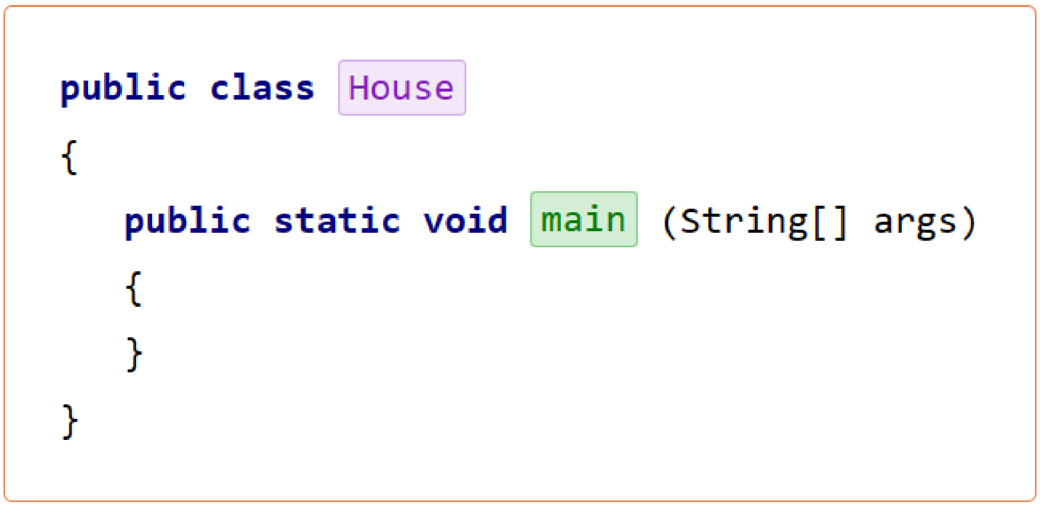
In the example above, `a` and `b` are variables, and `main` and `pi` are methods.

---

3. `main()` method

Classes can have variables and methods, but they don't have to. There can be classes without variables and classes without methods. There can even be classes with neither methods nor variables. Although such classes would make little sense.

A minimal program must consist of at least one *class*, which must have at least one *method* (function) that marks the program's starting point. This method must be named `main`.
A minimal program looks like this:

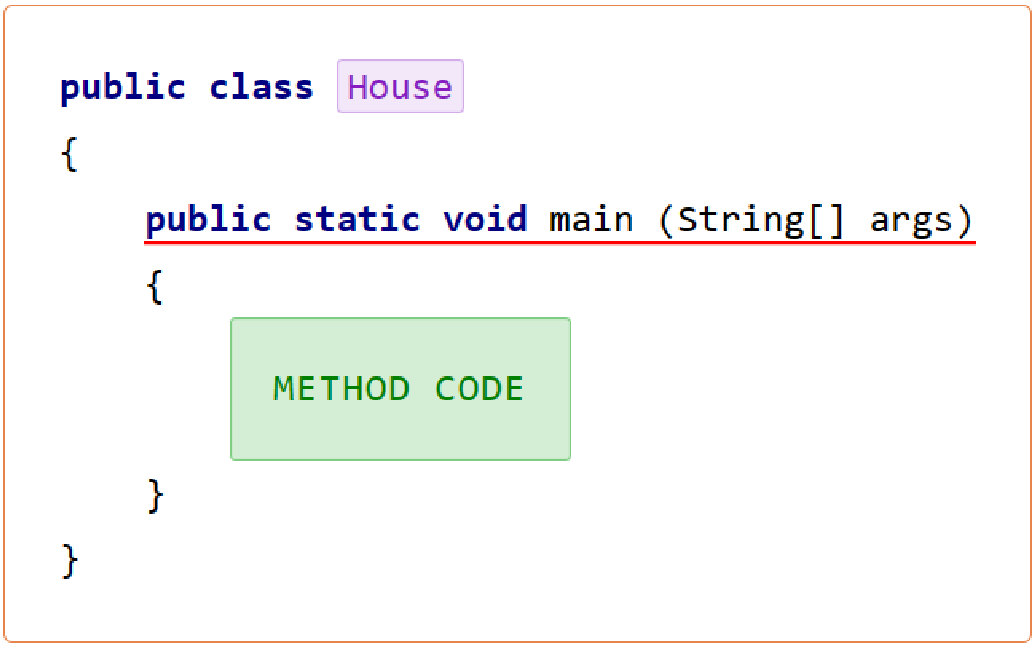


```
public class House
{
    public static void main (String[] args)
    {
    }
}
```

Minimal program

Note that the `main` method in the example above does not contain commands. That's right: the minimal program does not have a single command. That's precisely what makes it minimal.

The class that contains the program's starting point can have *any name*, but the `main` method, where the program starts executing, always takes the same form:



```
public class House
{
    public static void main (String[] args)
    {
        METHOD CODE
    }
}
```

The part highlighted in red – the declaration of the `main` method – is immutable