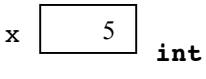
Variables and declarations

A *variable* is a name with an associated value. Throughout this course, we draw a variable as a named box with the value in the box. For example, here is a variable `x` associated with the value 5. Sometimes, we draw a line instead of a box, as shown to the right below:

Java is *strongly typed*, meaning that it has properties like the following.

1. *Each variable must have a type*. When we want to make the type of a variable explicit, we put the type to the right of the box.



2. *Only values of that type may be associated with the variable*. So, variable x, declared above, cannot contain a boolean value **true** or **false**.

3. Each expression has a type, which depends on the types of its operands, and

4. An operation can be applied only to operands of the appropriate type.

For example, an **`int`** variable may contain only an **`int`** value. A **`boolean`** variable may contain only a **`boolean`** value, and so on. Also, operation `b && c` is legal only if `b` and `c` are of type `boolean`.

Matlab is not strongly typed —one second, a variable may contain a complex number, the next second, an array of characters. The programming language C is very weakly typed —nothing prevents one from treating a string (character array) like "whatever" as a struct, an array of 32-bit integers, or anything else.

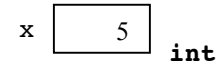
A later short essay tells you the advantages and disadvantages of strong typing.

Because of the strong typing principal, each variable must be declared before it is used. A variable declaration has the basic form

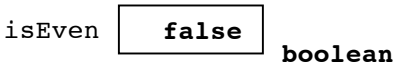
<type> <variable-name>

For example, here are two declarations and the corresponding variables, where we have annotated the variable itself with its type:

```
int x
```




```
boolean isEven
```



We make two points about declarations.

First, depending on where the declaration appears in a program, it may be followed by a semicolon or a comma, and it may also be preceded by an "access modifier", which indicates what part of the program can reference it, and other things. But the two basic parts —type and variable name— will always be required.

Second, the declaration is *not* a statement to be executed. It is simply an announcement that a variable is needed in a program. A variable declaration is *never* executed.

Declarations in the interactions pane

Let's put a declaration in the interactions pane and see what the value of the variable is:

```
int k;
k              // the initial value of k is 0
```

Note that if we try to declare k again, Java complains. Each variable can be declared only once.

Java conventions for variable names

Here are rules for identifiers —names of variables. They are:

(1) A variable contains only letters, numbers, and the characters `'_'` and `'$'`.

(2) The identifier does not start with a number.

But most Java programmers follow the convention that

(1) Identifiers do not contain `'$'` —this char is used at times by the compiler.

(2) The first letter of a variable name is not a capital.

(3) If the name consists of a series of words, the first letter of each word is capitalized, and

Here are examples of good variable names:

```
apple    numberOf7s    isEven    numberOfChildren
```

The length of variables names is always a cause for argument. Make them too short, and you don't know what they mean. Make them too long, and the program looks cluttered and unwieldy.

Later, we give conventions for variable names depending on where they are declared, keeping them short if their use is close to the declarations and making them longer if their use if far from their declarations.