Memory addressing and variables

1. How memory is organized

Every computer has *internal memory*. What is it? What properties does it have? And, most importantly, how does it benefit us?

Every program (including *programs* written in Java) is loaded into *main memory* before being executed. Main memory contains the program code (which is executed by the processor) as well as the program data (i.e. data that the program itself puts into memory).

What is memory and what is it like?

An Excel spreadsheet consists of *cells*. Each cell has its own *unique identifier* (`A1`, `A2`, ... `B1`, `B2`). If you know a *cell's identifier*, then you can always write some value into it or get whatever value is stored there. Computer *memory* is organized in a very similar way.

	A	B	C	D	E	F	G	H	I
1									
2		13	11						
3									
4	A	m	i	g	o				
5		G	e	r	m	e	s		
6									
7									

The program and program data are stored in *memory* when the program is running. All computer memory is comprised of small cells called *bytes*. Each cell has a unique identifier, or number, associated with it: `0`, `1`, `2`, `3`, ...; (the numbering starts from zero). If we know a *cell's number*, we can save data in it. Or get data from it. Some cells store the program's code, i.e. the set of commands for the processor. Others store the data used by the program. The cell's number is also called the *cell address*.

The processor knows how to execute commands that have been loaded into memory. Almost all processor commands are something like *take data from some cells*, *do something with them*, then *send the result to other cells*.

We combine hundreds of simple commands to get complex and useful commands.

When a variable is declared in code, a chunk of *memory* that isn't already being used is allocated for it. This is usually a few bytes. Declaring a variable requires that you indicate the type of information the program will store in it: numbers, text, or other data. After all, if you don't know the type of information to be stored, then it isn't clear how large of a block of memory needs to be allocated for the variable.

At the dawn of the computer age, programs worked directly with memory addresses, but then, for programmers' convenience, the cells began to be given names. A *unique* variable name is above all for the convenience of programmers, since the program handles plain memory addresses just fine.

---

2. Variables in memory

In all, Java has 4 data types for storing integers. These are `byte`, `short`, `int` and `long`.

Type	Size in bytes	Origin of the type's name
`byte`	`1`	*byte* is a deliberate respelling of bite to avoid confusion with bit
`short`	`2`	Short for *Short Integer*
`int`	`4`	Short for *Integer*
`long`	`8`	Short for *Long Integer*

Additionally, Java has 2 types for real numbers: float and double:

Type	Size in bytes	Origin of the type's name
`float`	`4`	Short for *Floating Point Number*
`double`	`8`	Short for *Double Float*

Each time program execution reaches a command to create a variable, a small block of memory is allocated for it (the size depends on the type of the variable).

> The address of a variable is the address of the first cell of the allocated memory block.

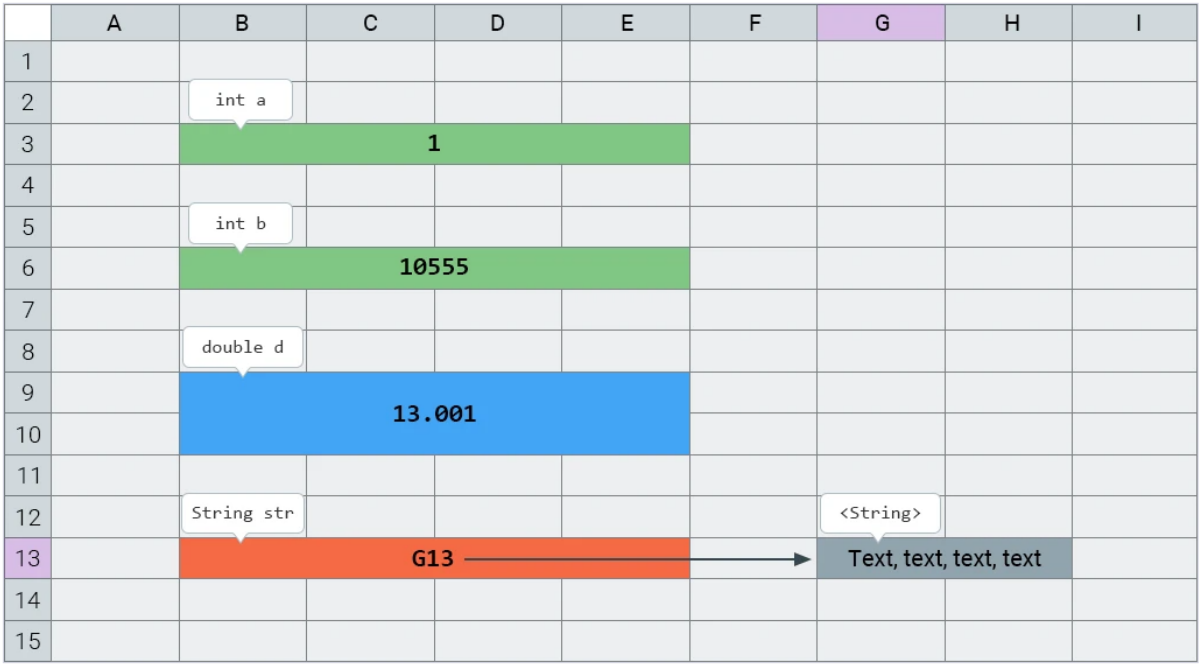
Java programs are not allowed to access memory directly. Any and all work with memory happens only through the Java virtual machine.

---

3. The `String` type in memory

The `String` type can store large amounts of data, which implies that it is not just a data type, but a full-fledged class.

The `String` object is placed in an allocated block of memory that stores the address of another block of memory in which the text is stored.



The `int a` variable occupies `4` bytes and stores the value `1`.

The `int b` variable occupies `4` bytes and stores the value `10,555`. We use a comma as the thousands separator. And we use a period as the decimal separator.

The `double d` variable occupies `8` bytes and stores the value `13.001`.

The `String str` variable occupies `4` bytes and stores the value `G13`, which is the address of the first cell of the memory block containing the text.

A text of the `String object` is stored in a separate block of memory. The address of its first cell is stored in the `str` variable.

4. Why numbering starts with zero in programming

People often wonder why programmers almost always start counting from zero. Well, the fact is that there are many situations when *it is more convenient to count from zero* (of course, there are also situations when it is more convenient to count from `1`).

The simplest example of is memory addressing. If your variable has been allocated `4` bytes of memory and you know that `X` is the address of the first byte, then what are the addresses of each byte? `X+0`, `X+1`, `X+2`, `X+3`. As simple as that, we have a group of bytes that can be accessed with indices `0`, `1`, `2`, `3`.

When we think of a relative address within a data block, indexing from zero is what makes sense. This is the main reason for *counting from zero*.