Variables in Java

In this lesson, an introduction to variables, their naming conventions, where they are stored in memory, and how much space they take, will be explained.

We'll cover the following

Introduction

Variables are just like containers which hold the values while the program is executed.

> A **variable** is a storage location paired with an associated symbolic name (an identifier), which contains some known or unknown quantity of information referred to as a value.

Variables in Java are **strongly typed**; thus they all must have a **data type** declared with their **identifier**. In this lesson, we will discuss *identifiers*, and we will leave data types for later.

Naming a variable

Just as every person has a name that helps identify them, every variable has a name associated with it. There are certain naming conventions that one must follow in order to decide a name for the variable. This name is called the variable's *identifier*. Variable names should be descriptive based on what data item they are declared to store.

There are three rules to create an identifier:

1. Characters from **A to Z**, as well as their lowercase counterparts, can be used.
2. Numbers from **0-9** can be used.
3. Special characters that can be used are **$** (the dollar sign) and _ (underscore).

> **Note:**
>
> 1. A variable must never start with a number
> 2. The identifier can not have spaces in it

While you can give a variable any name, the identifier must describe or be related to the data being stored inside it. The same way when you name a book, it is indicative of the story or characters inside it, the identifier must also be related to the data it represents.

Think of it this way, a book about wizards and magic being called **"Facts and Figures of the Modern World"**. Doesn't make sense, does it? Rather, if we called it, **"A Magical Journey"**, it would make much more sense. Below are a few examples of the identifiers we can use.

Let's say we have a book, a book has multiple details- the name of the book, the number of copies published, the author and the publishing company. Imagine if we gave the identifiers as 'a', 'b', 'c', and 'd'. Wouldn't that be confusing? Now we will give them identifiers in line with the naming conventions that we have described above:

```
//Declaring variables in java
 char author_name;
 char book_name;
 char publishing$company;
 int copies_published_number;
```

If we had written **Book name**, this would not have been a valid identifier because of the spaces in the title. Similarly, **#OfCopies** would have been an invalid identifier as well because it uses a special character that is not allowed. Neither will be **1_BookName** as it starts with a number.

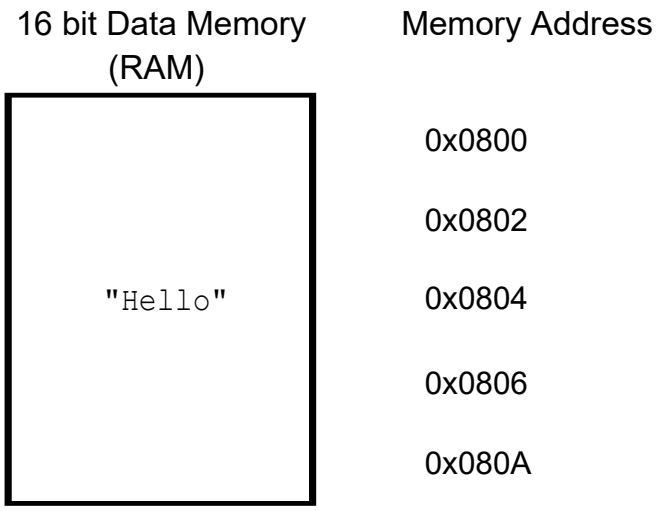
Where are the variables stored?

Variables, once declared in the program, are allocated space in memory. Memory itself is organized as a long collection of one byte after another. These bytes are

sequentially numbered, with the number being referred to as a memory address

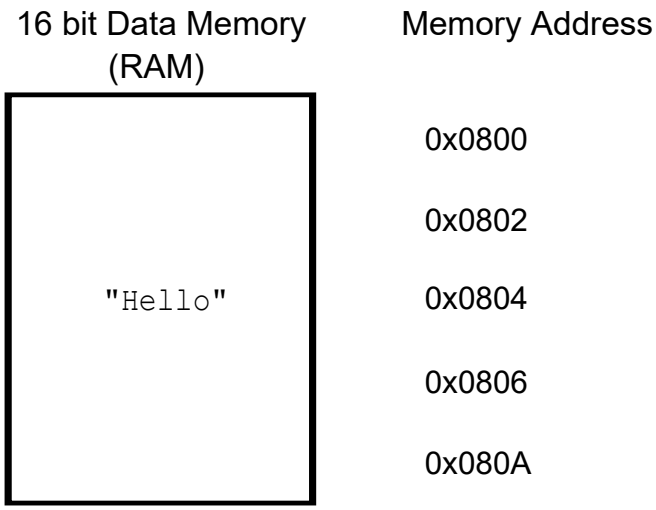
for the corresponding location. It is customary to use hexadecimal numbers to show the addresses of memory locations.

The value of a given variable is stored in some memory location. Referring to a piece of data using its address is tedious, that is why programming languages provide the abstraction of referring to each piece of data with a variable name.



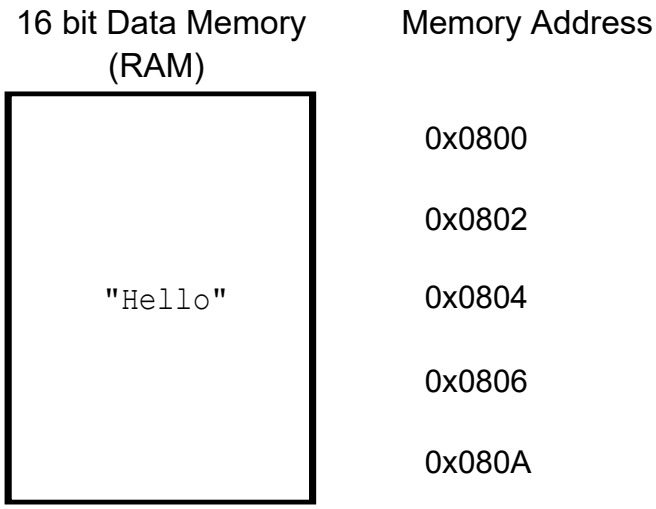
Storing "Hello" in memory at 0x0804 address

1 of 3



Is there an easier way to read the text stored at address 0x804?

2 of 3



greeting represents variable name. It refers to memory location 0x0804

3 of 3

Now that we have placed this variable in the machine's memory, one can raise the question of how much space does it take?

Size of a variable?

The number of locations occupied by a given variable gives the size of the variable in memory. For example, a shorter piece of text like "Hello!" occupies less memory than a longer one, like "Hello world!". Java programs have different **data types** that define the space that a variable takes. The smallest space that a variable can take is a data type called **byte** that takes **one byte**.

Now that we know some basic information about variables and how they are stored in memory let's look at the different data types that can be assigned to a variable in Java.