Variables

What is a variable?

- A variable is an address or an index to some place in the computers memory which holds data.
- It's called a variable because it can be changed.
- Each variable has a name and a type.
- The variable is always referred to by its name.
- A variable cannot be used in a program unless it has been declared.

An example of a variable...

- `int myNumber;`
- You are declaring the variable named "myNumber" of type int (integer). That is, a whole number.
- You can then assign a specific integer to your variable. It's like having a bucket, and adding a label to it. Then you put something inside it.
- `myNumber = 9;`
- What this is saying is: Put the integer 9 into the bucket labeled “myNumber".

Assigning data to a variable

- However, the bucket is designed to only hold a whole number. And if you put anything else into it besides a whole number, it will break, and your program will not run. For instance :
- `myNumber = 9.5; //Error!`
- This doesn't work because we are trying to put a decimal number into a bucket of type integer. The bucket labeled "myNumber" is expecting an integer. Nothing else will fit into that bucket.
- Even though the bucket is labeled "myNumber", the name has no actual meaning.
- For example if you make a bucket of type `int` but call it "myDecimalNumber", it still can only accept integer numbers!

Example

```
class Example
{
  public static void main ( String[] args )
  {
    long myNumber = 9;    //the declaration of the variable

    System.out.println("The variable contains: " + myNumber );
  }
}
```

Assigning data to a variable

- Putting data into a bucket is called variable assignment. You assign data to a variable by using the = sign. For all practical purposes, the label on the bucket equals the data inside the bucket.

- `myNumber = 123;`

- When you see the equals sign in a statement like this, it helps to say out loud to yourself: I am putting the number 123 into the integer variable called “myNumber”.

- What does this do?

- `int myNumber;`
- `myNumber = 123;`
- `myNumber = 321;`

Assigning data to a variable

- If you assign new data to a variable, you destroy the data that was already in there. In the above example, the first line creates an integer bucket. The second line puts the number 123 into the bucket. The third line puts the number 321 into the bucket and *completely destroys whatever was in there before*.

- You can think of it like this: When you first create the variable, there is a little blackboard in the bucket. When you assign data to the variable, you are taking out the blackboard, erasing whatever is on it, and writing the new data on the blackboard, then putting it back in the bucket.

- You can also create the variable and assign data to it at the same time:

- ```
  int myNumber = 1001; //both creates and assigns
  ```
  ```

Variable Declaration

There are several ways to declare variables:

- 1) dataType variableName;
  - Declares a variable, declares its data type, and reserves memory for it. It says nothing about what value is put in memory
    - Ex. int myNumber;
- 2) dataType variableName = initialValue;
  - Declares a variable, declares its data type, reserves memory for it, and puts an initial value into that memory.
    - Ex. Int myNumber = 123;

Variable Declaration

There are several ways to declare variables:

- 3) dataType variableNameOne, variableNameTwo;
  - Declares two variables, both of the same data type, reserves memory for each, but puts nothing in any variable.
  - Ex. int myNumber, otherNumber;
- 4) dataType variableNameOne = initialValueOne,
- variableNameTwo = initialValueTwo;
  - Declares two variables, both of the same data type, reserves memory, and puts an initial value in each variable.
    - Ex. Int myNumber = 123, otherNumber = 321;

Rules of Variable Naming Convention

- Variable names are **case-sensitive**.
- It can contain Unicode **letter, Digits and Two Special Characters**
  - Like Underscore and dollar Sign.
- Length of Variable name can be any number.
- It's necessary to use Alphabet at the start.
- Some auto generated variables may contain '**$**' sign.
- Subsequent characters may be **letters, digits, dollar signs, or underscore characters**.

Not Allowed

- **Dollar Or Underscore** is not allowed as first letter.
- **White space** is not permitted.
- **Special Characters** are not allowed.
- **Digit at start** is not allowed.
- Variable name must not be a **keyword or reserved word**.

Variable Naming

- If variable name contain two words then write first letter of second word in Capital Case.

Example : Multiple Words in Variable Name

```
sumOfNumber

firstNumber

skinColor
```

- If variable name contain single word then write that word in small case.

Example : Single Word in Variable Name

```
sum

first

last
```

Which of the following variable declarations are correct?

- int myPay, yourPay;
  - int myPay, yourPay; // OK
- long good-by ;
  - long good-by ; // bad identifier: "-" not allowed
- short shrift = 0;
  - short shrift = 0; // OK
- double bubble = 0, toil= 9, trouble = 8 byte the bullet ;
  - double bubble = 0, toil= 9, trouble = 8 // missing ";" at end.

Which of the following variable declarations are correct?

- int double;
  - int double; // bad identifier: double is a reserved word
- char thisMustBeTooLong ;
  - char thisMustBeTooLong ; // OK in syntax, but a poor choice // for a variable name
- int 8ball;
  - int 8ball; // bad identifier: can't start with a digit
- float a=12.3; b=67.5; c= -45.44;
  - float a=12.3; b=67.5; c= -45.44; // bad syntax: don't use ";" to separate variables

Operations

- Certain activities are so fundamental that they are built right into the language and have their own symbols. These symbols are called operators. Here are some mathematical operators: +, -, *, /, %, ++, = that work on data in integer buckets (variables of type `int`).

- For example, if we want to add two numbers together we can use the addition operator.

```
int x = 5;
int y = 6;
int z = x + y; //Now what is in the z bucket?
```

Operations on variables

- `int x = 5;`
- `int y = 6;`
- `int z = x + y;`
-
- Now `z` contains the number 11.

- In the third line we did is the following: took a number out of the integer bucket `x`, took another number out of the integer bucket `y` and then gave them to the CPU and told it to add those two numbers together. Then we took that number and put it inside of the bucket `z`.

- Try this:
- `int x = 5;`
- `int y = 6 + x;`
- `int z = x * y; //What number is assigned to z?`

Operations on variables

- `int x = 5;`
- `int y = 6 + x;`
- `int z = x * y; //What number is assigned to z?`

- After the second line, `y` contains the number 11, and `x` still contains the number 5. So at the third line we take the number 5 out of `x` and 11 out `y` and tell the CPU to multiply them together. We put the result into `z`. Since 5 times 11 equals 55, `z` now contains the number 55.