Variables and Types

Variables can be thought of as containers that hold values that can change while a program is running. A variable is created by writing a **declaration** that has the **variable name** and the **type** of the values that the variable will hold.

Variable names

Variable names are chosen by the programmer but must follow certain rules and should also follow certain conventions of good style. They must only have characters that are alphabetic (a-z and A-Z), $, _ or a number. The first character of a variable name must not be a number. Names are case sensitive, so "price" and "Price" are two different variables. Variable names should also avoid the Java keywords.*

Examples of variable names: age, studentId, number7, student_id

Good programming practice is to make the variables names meaningful for the values they will hold, either by using full descriptive words or some easily recognizable abbreviation. There are some capitalization conventions: one is that if the variable consists of more than one word, all but the first should be capitalized, the so-called "camel case". Examples are NetBeans, myFrame, goldStandard, payRate

Data Types

In this class, we will only use the most common data types for basic data. These will be

int	integer numbers: -7, … -1, 0, 1, 2, … 789, …
double	decimal numbers: 6.5, 1.998, -17.34
boolean	the two values **true** and **false**

For text data, we will use the type String. To write values of this type, put double quotes around the text.

String	example values are "Hello", "This is a message.", "Retweet #hcr"

These types are often called primitive types because they can be used with arithmetic or other operators. But String has a special status as also being an object type and has additional functions that can be used with values of the type.

---

* Java keywords

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

As we progress through the class, we will also use various object types. These types have functions that can be used and will be discussed more later. Some examples that we have seen so far include Frame, TextField, Button and Integer.

Variable Declarations

Every variable must have a declaration that occurs in the program sequence before the variable is used.

A simple declaration has a type followed by the variable name. Declarations occur as statements, so they are also followed by a semicolon.

```
int age;
double gpa;
String studentName;
boolean flag;
```

If you have more than one variable of the same type, they can all be put into one variable declaration.

```
int age, num1, num2;
double payAmount, payRate;
```

In programming languages, there are not special types for dealing with currency, usually double values are used for computing monetary amounts but the values can be displayed to the user as currency. For example, 4.5 would be displayed as $4.50.

Assignments

In order to give a variable a value, you assign a value to it. The value could be a simple value, or it could be the result of an expression. The format is as follows:

```
variable = value;
variable = expression;
```

Note that this use of the "=" sign means move the value on the right to the variable on the left. (This is different from what "=" means in most mathematical equations.)

The value may be one of a number of things:

An actual value	IQ = 115;
The value of another variable	yourIQ = myIQ;
The result of a computation	yourIQ = myIQ - 40;

Although arithmetic is covered more in another section, here are some more examples of computing values from arithmetic expressions:

The results of computing a multiplication:

```
salesTax = basePrice * taxRate;
```

The result of a concatenation of strings:

```
welcome = "Howdy" + ", " + "Pardner";
```

There are times in your program that you want a variable to start with a particular value. You can initialize a variable with an assignment statement following the declaration, or you can use a declaration that includes the initialization directly.

```
int counter = 0;
double totalPrice = 0.0;
```

Now that we've seen the forms of assignment, here are some more thoughts on what assignment means.

Since you can use expressions on the right hand side of an assignment, you should always think about evaluating the expression first. Whatever value results from that will be put into the variable on the left.

Example 1:

```
int bonus = 5;
int points = 10;

points = 25 + bonus;
```

To see what this assignment statement does, first evaluate the expression on the right. This means to take the value of bonus (5) and add it to 25, getting 30. Now take the value of that expression (30) and put it as the current value of points. So the final value of points is 30.

Example 2:

Assume that this assignment statement follows the first example:

```
points = points + bonus;
```

We evaluate the assignment statement as before: we first evaluate the expression on the right, taking the current value of bonus (5) and the current value of points (30), getting 35. Now take the value of that expression (35) and put it as the current value of points. So the final value of points is 35.

Note that in programming, unlike mathematics, "x = x + 1", is meaningful, and is quite commonly used to keep adding 1 to a variable.