Unit 2 ~ Variables

Unit 2A

Without variables, we can't really write meaningful programs. We need variables to keep track of scores, or results, or to have the program make choices. [1]A variable holds a value; that's its only purpose.

For example, let's say you have a variable called X. Let's do this:

```
X = 4
```

If I ask 'what is the value of X?', you would have to tell me 4. X right now equals 4 because that's how it is set.

```
X = "Hello world!"
```

Now X is equal to Hello world!

Primitive Variable Types

Type	Holds
boolean	true or false
byte	an integer between -128 and 127
char	a single character, such as 'A' or '6'
double	a decimal value, such as 3.4 or -54.634
float	a decimal value, -big to +big
int	an integer, such as 4 or -67
long	an integer value from -big to +big
short	an integer -32768 to 32767

Click this link for more in-depth information on primitive data types.
http://www.tutorialspoint.com/java/java_basic_datatypes.htm

String Type

We will also use the String data type quite a bit. It is important to note that it is NOT primitive (note that it is ***S**tring*, not ***s**tring* which indicates that it is an *object*). Strings hold sequences of characters, such as "kjdf45". Also note the double quote " rather than the single quote ' used for the char type.

Naming Variables

1. An identifier is a name that will be used to describe classes, methods, constants, variables; anything a programmer is required to define.

[1] http://www.java-made-easy.com/java-variables.html

2. The rules for naming identifiers in Java are:
    - Identifiers must begin with a letter.
    - Only letters, digits, or an underscore may follow the initial letter.
    - The blank space cannot be used.
    - Identifiers cannot be reserved words. Reserved words or keywords are already defined in Java. These include words such as *new*, *class*, *int*, etc.
3. Java is a case sensitive language. That is, Java will distinguish between upper and lower case letters in identifiers. Therefore:

    `grade` and `Grade` are different identifiers

4. Be careful both when naming identifiers and when typing them into the code. Be consistent and don't use both upper and lower case names for the same identifier.
5. A good identifier should help describe the nature or purpose of whatever it is naming. For a variable name, it is better to use

    `grade` instead of `g`, `number` instead of `n`.

6. However, avoid excessively long or "cute" identifiers such as:

```
gradePointAverageForStudentsAToZ
bigHugeUglyNumberThatIsPrettyPointlessButINeedItAnyway
```

Remember that the goal is to write code that is professional in nature; other programmers need to understand your code.

7. Programmers will adopt different styles of using upper and lower case letters in writing identifiers. The reserved keywords in Java must be typed in lower case text, but identifiers can be typed using any combination of upper and lower case letters.

6. The following conventions will be used throughout this course:
    - A single word identifier will be written in lower case only. Examples: `grade`, `number`, `sum`.
    - Class names will begin with upper case. Examples: String, DrawingTool, SketchPad, Benzene. This is known as **ProperCase**.
    - If an identifier is made up of several words, all words beyond the first will begin with upper case. Examples: `stringType`, `passingScore`, `largestNum`, `DrawHouse`, `SketchPad`. This is known as **titleCase**.
    - Identifiers used as constants will be fully capitalized. Examples: `PI`, `MAXSTRLEN`.

Reserved Words

Here are the reserved words for Java (keywords). You may not redefine any of these reserved words. These words ALWAYS appear in lowercase. You should not give methods or variables the same name as any of these keywords.

abstract	boolean	break	byte
case	catch	char	class
const	continue	default	do
double	else	extends	final
finally	float	for	future
generic	goto	if	implements
import	inner	instanceof	int
interface	long	native	new
null	operator	outer	package
private	protected	public	rest
return	short	static	super
switch	synchronized	this	throw
throws	transient	try	var
void	volatile	while	

Unit 2A Worksheet

Write whether the following are (V)alid or (I)nvalid identifiers in Java.

1) C3PO	1) V
2) 31Dollars	2) i
3) A/B	3) i
4) GiveMeAnA	4) V
5) Pi3141592645	5) V
6) Pi3.141592645	6) i
7) input	7) V
8) why_the_triple_underscore	8) V
9) thisisthelongestcomputeridentifierinthehistoryofthemodernworld	9) V
10) 7up	10) i
11) up7	11) V
12) Help!	12) i
13) HiThere	13) V
14) HI_THERE	14) V
15) A2A2	15) V
16) 2A2A	16) i
17) *blank	17) i
18) _blank	18) V
19) XXX	19) V
20) X-	20) i

Declaring Variables

A variable must be declared before it can be initialized with a value. The general syntax of variable declarations is:

data_type variableName;

for example:

int number; //int is the data_type; number is the variableName;
char ch;

Variables can be declared in a class outside of any methods or inside of a method.

Variables can also be **declared** and **initialized** in one line. The following example code illustrates these aspects of variable declaration and initialization.

```
// first is declared and initialized
// second is just declared
int first = 5, second;
double x;
char ch;
boolean done;

second = 7;
x = 2.5;
ch = 'T';
done = false;

int sum = first + second;
```

first and *second* are of type **int** as declared in this one line.

first is assigned the value 5

second is not assigned a value in this **declaration.**

Initialization is done using the assignment operator (=). Initialization can occur at declaration time or later in the program. The variable sum was declared and used in the same line.

Where variables are declared is a matter of programming style and need since this determines how and where they can be used. This is their **scope**.

Unit 2B Worksheet

1a. In 2 statements, declare a variable named *numBeads* and assign it the value 5.

int numBeads;

numBeads=5;

1b. In one statement, declare a variable named *numBeads* and assign it the value 5.

int numBeads=5;

2. What is the final value of *yourNumber* after the last statement executes? 13

```
int myNumber = 5;
int yourNumber = 4;
myNumber = yourNumber * 2;
yourNumber = myNumber + 5;
```

3. What is the final value of *yourNumber* after the last statement executes? 11

```
int myNumber;
int yourNumber = 4;
myNumber = yourNumber + 7;
yourNumber = myNumber;
```

4. Determine the appropriate data type for each of the following values:

a. the number of basketballs in a department store int

b. the price of a basketball double

c. the number of players on a basketball team int

d. the average age of the players on a basketball team double

e. whether a basketball player has received a jersey or not boolean

f. the first initial of a basketball player's first name char

Unit 2C

Math Operators

1. Java provides 5 math operators as listed below:

+	Addition, as well as unary +
-	Subtraction, as well as unary -
*	Multiplication
/	Floating point and integer division
%	Modulus, **remainder** of integer or floating point division

2. For all the operators, if both operands are integers, the result is an **integer**. Examples:

2 + 3 → 5	9 - 3→6
4 * 8 → 32	11/2 → 5

3. Notice that 11/2 is 5, and not 5.5. This is because ints work *only* with whole numbers. The remaining half is lost in integer division.
4. If either of the operands is a double type, the result is a **double** type. Examples:

2 + 3.000 → 5.000
25 / 6.75→3.7037
11.0 / 2.0→5.5

When an integer and a double are used in a binary math expression, the integer is promoted to a double value, and then the math is executed.

In the example 2 + 3.000 → 5.000, the integer value 2 is promoted to a double (2.000) and then added to the 3.000.

5. The **modulus operator (%)** returns the **remainder** of dividing the first operand by the second. For example:

10 % 3→1 because 10 divided by 3 = 3 remainder 1	2 % 4 → 2
16 % 2 → 0	27 % 7 → 6

6. To obtain the answer of 5.5 to a question like 11/2, we must **cast** one of the operands.

(**double**)11/2 → 5.5

The same effect can also result from 11.0/2

Unit 2C Worksheet

DIRECTIONS : Fill in each blank with the correct answer/output. Assume each statement happens in order and that one statement may affect the next statement.

```
char charOne = 'H';              //16 bit unsigned integer data type     0..65535
byte byteOne = 24;               //8 bit integer data type            -128..127
short notBig = 32767;            //16 bit integer data type         -32768..32767
int intOne = 327670;             //32 bit integer data type
long bigInt = 7;                 //64 bit integer data type

float littleDec = 32.22f;        //32 bit real number data type
double doubleOne = 123.456;      //64 bit real number data type
```

```
int x = 600;              out.println(x);

x = -80;                  out.println(x);

byte b = 5;               out.println(b);

short s = 32767;          out.println(s);

double d = 9.9;           out.println(d);

d = 5.2;                  out.println(d);

float f = 9.87f;          out.println(f);

long big = 555845;        out.println(big);

x = s;                    out.println(x);

x = b + 10;               out.println(x);

int z = 'A'+ 1;           out.println(z);

char c = 'A' + 1;         out.println(c);
```

1. 600
2. -80
3. 5
4. 32767
5. 9.9
6. 5.2
7. 9.87
8. 555845
9. 32767
10. 15
11. 66
12. B

from variableWorksheet0.docx

Unit 2C Lab

Create a RectanglePerimeter app that calculates and displays the perimeter of a rectangle with a width of 4 and a length of 13, using variables as appropriate. **Think ahead**: you are likely going to want to change the width and length later to make this program more useful.

char values are integers.

If you google an **ASCII** chart, you will see that 'A' has the value 65.

You should memorize the ascii values of 'A', a', and '0'.

Unit 2D

Assignment Operators

1. The statement

   ```
   number = number + 5;
   ```

   is an example of an **accumulation** statement. The old value of number is incremented by 5 and the new value is stored in number.

2. The above statement can be replaced as follows:

   ```
   number += 5;
   ```

3. Java provides the following assignment operators:

   ```
   +=  -=  *=  /=  %=
   ```

4. The following examples are equivalent statements:

`rate *= 1.05;`	`rate = rate * 1.05;`
`sum += 25;`	`sum = sum + 25;`
`number %= 5;`	`number = number % 5;`

5. Incrementing or decrementing by one is a common task in programs. This task can be accomplished by the statements:

   ```
   n = n + 1;   or   n += 1;
   ```

6. Java also provides an increment operator, ++. The statement n = n + 1 can be rewritten as ++n. The following statements are equivalent:

`n = n + 1;`	`++n;`
`sum = sum + 1;`	`++sum;`

7. Java also provides for a decrement operator, --, which decrements a value by one. The following are equivalent statements:

`n = n - 1;`	`--n;`
`sum = sum - 1;`	`--sum;`

8. The increment and decrement operators can be written as either a prefix or postfix operator. If the ++ is placed before the variable it is called a pre-increment operator (++number), but it can follow after the variable (number++), which is called a post-increment operator. The following three statements have the same effect:

```
++number;   number++;   number = number + 1;
```

9. Before looking at the difference between prefix and postfix unary operators, it is important to remember Java operators solve problems and often return values. Just as the assignment operator (=) returns a value, the ++ and -- operators return values. Consider the following code fragments:

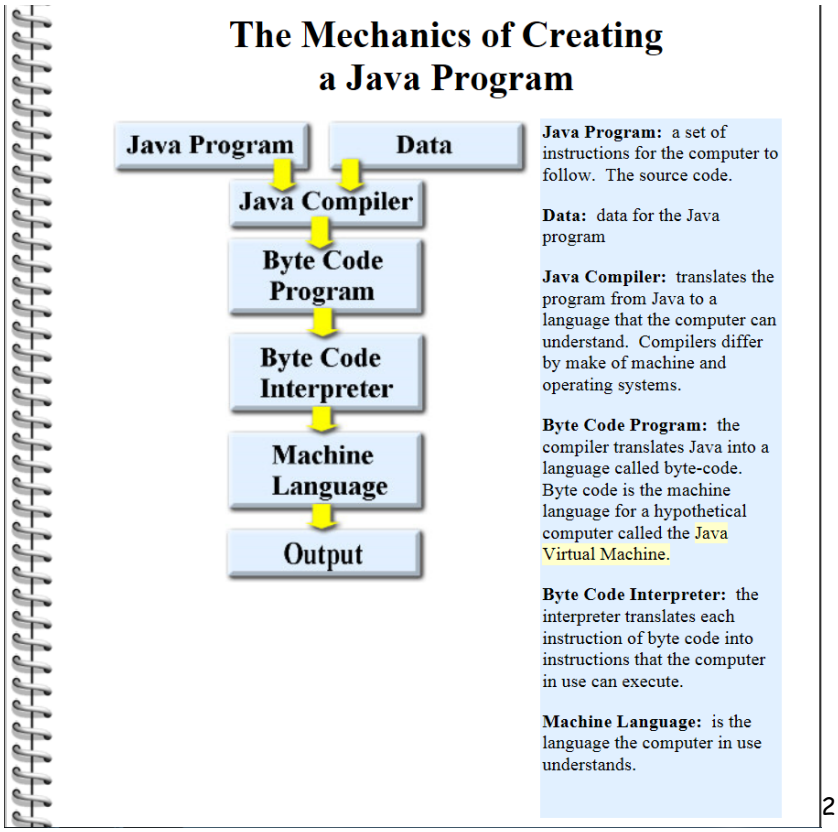
```
int  a=1, b;
b = ++a;
```

After execution of the above code a = 2 and b = 2

```
int  a=1, b;
b = a++;
```

After execution of the above code a = 2 and b = 1

10. The statement b = ++a uses the pre-increment operator. It increments the value of a and returns the new value of a.
11. The statement b = a++ uses the post-increment operator. It returns the value of a and then increments a by 1.



[2]

[2] http://mathbits.com/MathBits/Java/Introduction/mechanics.htm

Unit 2D Worksheet[3]

1. Rewrite the statements below using the appropriate assignment operator.

a) total = total +10; total+=10;

b) numStones = numStones -1; nS-=1;nS--;--nS;

c) days = days % 24; days%=24;

d) price = price * 12; price*=12;

2. Using the following declarations, rewrite the statements to include the appropriate type casting, rounding where necessary. If type casting is not necessary, write NA.

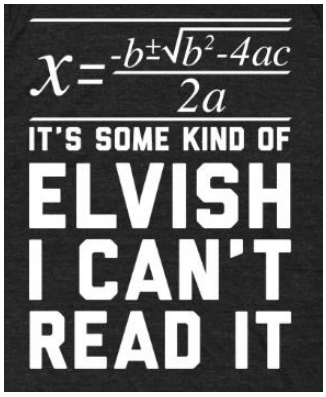
```
int j = 5;
double k = 1.6;
int y;
double z;
```

a. y = j * k;	y=j*(int)k;
b. z = j * k;	NA
c. z = k * k;	NA
d. j = k;	j=(int)k;
e. k = j;	//k=(double)j;-wrong// NA
f. y = j + 3;	NA

Unit 2D Lab

Create a Distance app that calculates and displays the total distance of a race with 3 segments. For now, the first segment is 12.2 km, the second is 10.6 km, and the third is 5.8 km. Use the assignment operators discussed in Unit 2D.

For Fun



[3] LVP Ch 4

Unit 2E

String data type

Strings store groups of characters. They are NOT primitive. They are an object and the importance of knowing this will become evident later on.

```
String fname = "Fred";
String lname ="Flintstone";
```

You may **concatenate** multiple strings with the + sign.

```
String fullName = fname + " " + lname;
System.out.println(fullName);                 //outputs Fred Flintstone
System.out.println(lname + ", " + fname);     //outputs Flintstone, Fred
```

Boolean data type

A boolean can store true or false. A boolean cannot store letters or numbers.

```
boolean go = true;
System.out.println(go);          //output true
boolean stop = false;
System.out.println(stop);        //output false
```

MIN and MAX

The MIN_VALUE and MAX_VALUE fields store the minimum and maximum values that can be stored in a particular type.

For example,

```
System.out.println(Short.MIN_VALUE);          //outputs 32768
```

prints out the smallest possible value of type short.

Constants

Values that don't change while the program is running are referred to as constants. They are declared with the keyword *final,* and they CANNOT be altered. We use UPPERCASE letters for their identifier, as well.

```
final double PI = 3.1415;
```

Unit 2 Lab A

Write an app that calculates and displays the area and circumference of a circle with a radius of 10.5.

Use variables and constants as appropriate. Add your comments and submit for marking.

Unit 2 Worksheet A[4]

DIRECTIONS : Fill in each blank with the correct answer/output. Assume each statement happens in order and that one statement may affect the next statement.

```
double z = 123.456;       long x = 7;
int a = 5, b = 2;         char var = 'H';
```

```
System.out.print( 3 + 3 * 3 );                  // LINE 1
System.out.print( a * (a % b) );                // LINE 2
System.out.print( b / a );                      // LINE 3
System.out.print( 'A'+ 5 * b);                  // LINE 4
System.out.print( 1 % 5 );                      // LINE 5
System.out.print( a % b );                      // LINE 6
System.out.print( b % a );                      // LINE 7
System.out.print( 'A'+ 5 );                     // LINE 8
System.out.print( (double)( a / b) );           // LINE 9
System.out.print( (double)a / b );              // LINE 10
System.out.print( var + 5);                     // LINE 11
System.out.print((char)(var + 5));              // LINE 12
a=var+2;
System.out.println( a );                        // LINE 13
z=var+5;
System.out.println( z);                         // LINE 14
var='A'+4;
System.out.println( var);                       // LINE 15
a*=2+5;
System.out.println( a );                        // LINE 16
var=(char)(z-25);
System.out.println( var );                      // LINE 17
a++;
System.out.print( a );                          // LINE 18
b--;
System.out.print( b );                          // LINE 19
++x;
System.out.print( x  );                         // LINE 20
System.out.print( --var );                      // LINE 21
```

1. 12
2. 5
3. 0
4. 75
5. 1
6. 1
7. 2
8. 70
9. 2.0
10. 2.5
11. 77
12. M
13. 74
14. 77
15. E
16. 518
17. 4
18. 519
19. 1
20. 8
21. 3

[4] MathCalcWorksheet1.doc

Unit 2 Worksheet B

[5]DIRECTIONS : Fill in each blank with the correct answer/output. Assume each statement happens in order and that one statement may affect the next statement.

```
char cVar = 'H';                //16 bit unsigned integer data type     0..65535
byte bVar = 24;                 //8 bit integer data type            -128..127
short notBig = 32767;           //16 bit integer data type         -32768..32767
int iVar = 327670;              //32 bit integer data type
long lVar = 7;                  //64 bit integer data type

float fVar = 32.22f;            //32 bit real number data type
double dVar = 123.456;          //64 bit real number data type
```

If any statements cause an error, write ***ERROR*** in the blank and treat the line that caused the error as if it did not exist.

```
int x = 128;            out.println(x);   //1
x = -98;                out.println(x);   //2
byte b = 24;            out.println(b);   //3
char c = 97;            out.println(c);   //4
double d = 9.9;         out.println(d);   //5
d = 5.2;                out.println(d);   //6
float f = 9.87f;        out.println(f);   //7
short s = 350;          out.println(s);   //8
int z = 'A'+5;          out.println(z);   //9
c = 'A'+5;              out.println(c);   //10
double w = 'a'+5;       out.println(w);   //11
long u = 'A'-48;        out.println(u);   //12
w = f+5;                out.println(w);   //13
b = (byte)x;            out.println(b);   //14
w = 'A' * 2.0;          out.println(w);   //15
f=w;                    out.println(f);   //16
u=982743L;              out.println(u);   //17
d = 3.2e2;              out.println(d);   //18
s=c;                    out.println(s);   //19
z = w;                  out.println(z);   //20
```

1. 128
2. -98
3. 24
4. a
5. 9.9
6. 5.2
7. 9.87
8. 350
9. 70
10. F
11. 102.0
12. 17
13. 14.87
14. -98
15. 130.0
16. Error
17. 982743
18. 320.0
19. Error
20. Error

Test on Units 1 & 2

Check for Theory Reviews on Blackboard

[5] variableworksheet1.doc