Variables and Expressions

Variables: *Declaring*, *Initializing*, *Assigning*

Lecture Contents

- ***Declaring*** and ***Initializing*** variables
- Identifier Conventions
- Reserved Words
- ***Assigning*** variables

Declaring a Variable

- Declaring a variable makes a place in memory to store a value.
- Variables are ***declared*** with the syntax:

```
<type> <identifier>;
```

Declaring a Variable

- Declaring a variable makes a place in memory to store a value.
- Variables are ***declared*** with the syntax:

```
<type> <identifier>;
```

- Example:

```
int myInteger;
```

`int myInteger`

  - Memory space is reserved to store data of type `int` (32 bits).
  - The ***identifier*** `myInteger` refers to this memory location
  - No value has been written to the storage (yet).

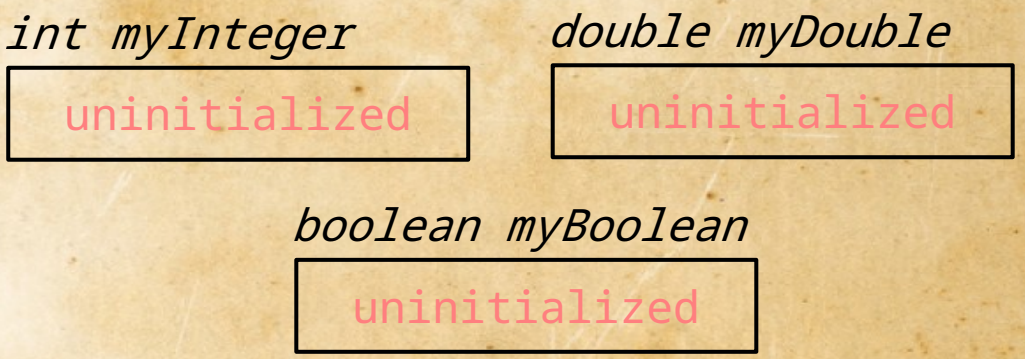
Declaring a Variable

- Declaring a variable makes a place in memory to store a value.
- Variables are ***declared*** with the syntax:

```
<type> <identifier>;
```

- Examples:

```
int myInteger;
double myDouble;
boolean myBoolean;
```



- Remember that identifiers for variables use lower ***camel case***
  - the first letter of every word, after the first word, is capitalized

Initializing a Variable

- ***Initialize*** means ***assign a value*** for the first time.
- This means to write a value into the memory space assigned to that variable.

Initializing a Variable

- ***Initialization*** uses an ***assignment operator*** (**=**).
- The Syntax is:

```
<identifier> = <value>
```

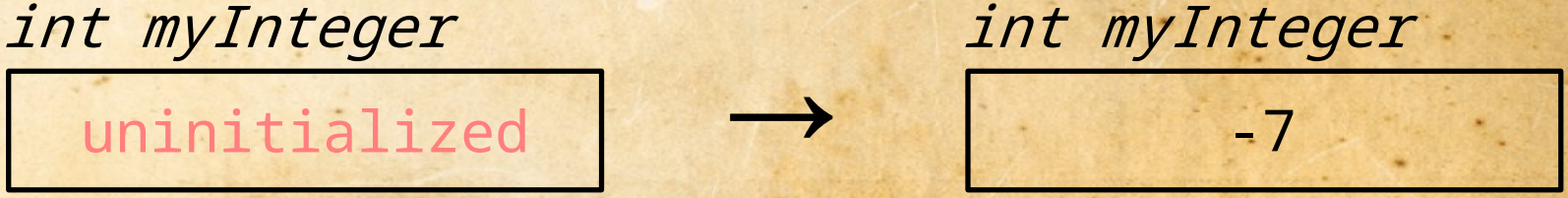
- A variable must be declared before it is initialized:

declaration:

```
int myInteger;
```

initialization:

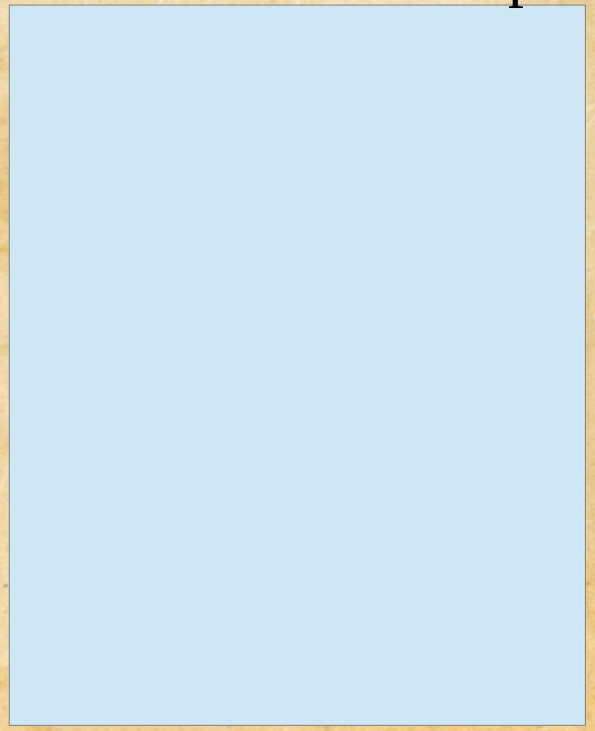
```
myInteger = -7;
```



Using Variables

- Create a class named `UnderstandingVariables`, add the following lines to the `main` method, and observe the output.

```
int x; // declaration
x = 3; // initialization
System.out.println("3 + 5");
System.out.println("3" + "5");
System.out.println(3 + 5);
System.out.println("x + 5");
System.out.println("x" + 5);
System.out.println(x + 5);
```



Using Variables

- Create a class named `UnderstandingVariables`, add the following lines to the `main` method, and observe the output.

```
int x; // declaration
x = 3; // initialization
System.out.println("3 + 5");
System.out.println("3" + "5");
System.out.println(3 + 5);
System.out.println("x + 5");
System.out.println("x" + 5);
System.out.println(x + 5);
```

```
3 + 5
35
8
x + 5
x5
8
```

Using Variables

- Anything within a set of quotation marks (") is treated as a `String` and not a number or variable
  - no calculations are performed.
  - the characters of the string will be printed.

```
System.out.println("3 + 5");    →   3 + 5
System.out.println("x + 5");    →   x + 5
```

Using Variables

- When a `String` is one of the operands
  - the other operator is converted to a `String` (if it is not already)
  - then the **+** operator performs ***concatenation*** (joining).

```
System.out.println("3" + 5);   →   35
System.out.println("x" + 5);   →   x5
```

Using Variables

- When a `String` is one of the operands
  - the other operator is converted to a `String` (if it is not already)
  - then the **+** operator performs ***concatenation*** (joining).

```
System.out.println("3" + 5);   →   35
System.out.println("x" + 5);   →   x5
```

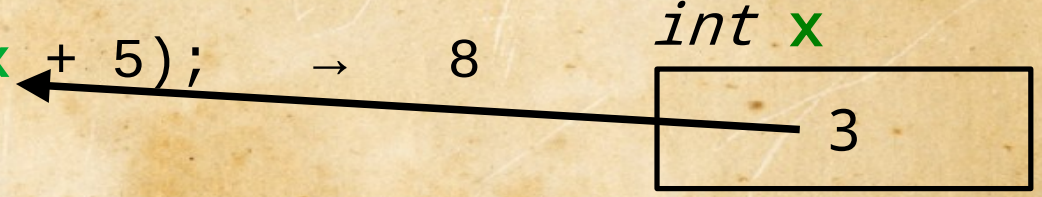
- If a number is enclosed in quotation marks, it is treated as a set of characters and not treated as an number.
- If text is enclosed in quotation marks, it is it is treated as a set of characters and not treated as an identifier.

Using Variables

- When an identifier is encountered in an expression, the identifier is replaced with its value that was stored in the variable.

```
System.out.println(x + 5);    →   8

System.out.println(x + "5"); →   35
```



In the second line of code, the value `3`, an `int`, is converted to a `String` (because the other operand is a `String`), then the two strings are *concatenated* (joined).

Combining declaration and initialization

- ***Declaration*** and ***initialization*** can be combined into a single statement using the syntax:

```
<type> <identifier> = <value>;
```

- Examples:

```
int myInteger = 65535;
double pi = 3.14;
boolean isSad = false;
```

Declaring and Initializing Variables

- ***Variables*** are *declared* and *initialized* with:
  - **<type> <identifier> = <value> ;**

```
int myInteger = 13;

double myReal = 1.78;

boolean isTrue = false;

String myText = "Hello World!";
```

Reserved Words in Java

- The following are reserved words in Java, so can not be used as identifiers:

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

Example: simple

- Declared with:
  - <type> <identifier> = <value> ;

```
public static void main(String[] args) {
    int myNumber = 13;
    System.out.println(myNumber);
}
```

```
13
```

Declaring and Initializing Variables

- ***Variables*** can be *declared* and *initialized* separately:
  - **<type> <identifier> = <value> ;**

```
int myInteger = 13;
```

```
int myInteger;
myInteger = 13;
```

```
double myReal = 1.78;
```

```
double myReal;
myReal = 1.78;
```

```
boolean isTrue = false;
```

```
boolean isTrue = false;
isTrue = false;
```

```
String myText = "Hello World!";
```

```
String myText;
myText = "Hello World!";
```

Identifier Conventions

- Class: PascalCase (UpperCamelCase)
- Variables/Methods: camelCase (lowerCamelCase)
- Final Variables: UPPER_CASE_WITH_UNDERSCORES

Examples

- Declared with:
  - <type> <identifier> = <value> ;

```
public static void main(String[] args) {
    int myNumber;
    myNumber = 13;
    System.out.println(myNumber);
}
```

```
13
```

Examples

- Variables must be *initialized* before they are used.

```
public static void main(String[] args) {
    int myNumber;
    System.out.println(myNumber);
    myNumber = 13;
}
```

Examples

- Variables must be *initialized* before they are used.

```
public static void main(String[] args) {
    int myNumber;
    System.out.println(myNumber);
    myNumber = 13;
}
```

```
ERROR:
The local variable myNumber may not have
been initialized
```

Variable *Assignment*

- Initialization is an assignment
- Variables can be assigned new values after being initialized

```
public static void main(String[] args) {
    int myNumber = 13;
    myNumber = 7;
    System.out.println(myNumber);
}
```

Declaring Variables

```
public static void main(String[] args) {
    int myNumber = 13;
    myNumber = 7;
    System.out.println(myNumber);
}
```

```
7
```

Declaring Variables

```
public static void main(String[] args) {
    int myNumber = 13;
    myNumber = myNumber + 7;
    System.out.println(myNumber);
}
```

Declaring Variables

```
public static void main(String[] args) {
    int myNumber = 13;
    myNumber = myNumber + 7;
    System.out.println(myNumber);
}
```

```
20
```

Declaring Variables

```
public static void main(String[] args) {
    int myNumber = 13;
    double myFloat = 6.5;
    System.out.println(myNumber/myFloat);
}
```

Declaring Variables

```
public static void main(String[] args) {
    int myNumber = 13;
    double myFloat = 6.5;
    System.out.println(myNumber/myFloat);
}
```

```
2.0
```

Declaring Variables

```
public static void main(String[] args) {
    int myNumber = 13;
    double myFloat = 2.5;
    myFloat = myNumber/myFloat;
    System.out.println(myFloat);
}
```

Declaring Variables

```
public static void main(String[] args) {
    int myNumber = 13;
    double myFloat = 2.5;
    myFloat = myNumber/myFloat;
    System.out.println(myFloat);
}
```

```
5.2
```

Declaring Variables

```
public static void main(String[] args) {
    int myNumber = 13;
    double myFloat = 2.5;
    myNumber = myNumber/myFloat;
    System.out.println(myNumber);
}
```

Declaring Variables

```
public static void main(String[] args) {
    int myNumber = 13;
    double myFloat = 2.5;
    myNumber = myNumber/myFloat;
    System.out.println(myNumber);
}
```

```
ERROR:
Type mismatch: cannot convert from
double to int
```

Declaring Variables

```
public static void main(String[] args) {
    int myNumber = 13;
    double myFloat = 2.5;
    myNumber = (int)(myNumber/myFloat);
    System.out.println(myNumber);
}
```

Declaring Variables

```
public static void main(String[] args) {
    int myNumber = 13;
    double myFloat = 2.5;
    myNumber = (int)(myNumber/myFloat);
    System.out.println(myNumber);
}
```

```
5
```

Declaring Variables

```
public static void main(String[] args) {
    int myNumber = 13;
    double myFloat = 2.5;
    myNumber = myNumber/(int)myFloat;
    System.out.println(myNumber);
}
```

Declaring Variables

```
public static void main(String[] args) {
    int myNumber = 13;
    double myFloat = 2.5;
    myNumber = myNumber/(int)myFloat;
    System.out.println(myNumber);
}
```

```
6
```

Declaring Variables

```
public static void main(String[] args) {
    int myNumber = 13;
    double myFloat = 2.5;
    myNumber = (int)myNumber/myFloat;
    System.out.println(myNumber);
}
```

Declaring Variables

```
public static void main(String[] args) {
    int myNumber = 13;
    double myFloat = 2.5;
    myNumber = (int)myNumber/myFloat;
    System.out.println(myNumber);
}
```

```
ERROR:
Type mismatch: cannot convert from
double to int
```

Declaring Variables

- ***Variables*** are declared with:
  - **<type> <identifier> = <value> ;**

```
int myInteger = 13;

double myReal = 1.78;

boolean isTrue = false;

String myText = "Hello World!";
```

AP Java Subset Primitive Types

- Integers
  - byte (1)
  - short (2)
  - **int** (4)
  - long (8)
- Real
  - float (4)
  - **double** (8)
- True/False
  - **boolean** (1 bit?)
- Letters
  - char (2)

Declaring and Initializing Variables

- A ***variable*** is some information/data stored in the computer.
- An ***identifier*** (or ***label***) is a name given to something so it can be referenced.
- ***Declaring*** a *variable* is telling the compiler that you will use a chosen *identifier* to refer to some particular type of data.
- When a value is ***assigned*** to a variable, that value is stored in the memory location referred to by the *identifier*.
- When a *variable* is ***initialized***, it is *assigned* a value for the first time.
- *Variables* are *declared* and *initialized* with:
  - ***<type> <identifier> = <value> ;***

Variables and Expressions

Variables: *Declaring*, *Initializing*, *Assigning*