Special Characters

//	**double slash**	**Marks the beginning of a single line comment.**
( )	**open and close parenthesis**	**Used in a method header to mark the *parameter list*.**
{ }	**open and close curly braces**	**Encloses a group of statements, such as the contents of a class or a method.**
“ ”	**quotation marks**	**Encloses a string of characters, such as a message that is to be printed on the screen**
;	**semi-colon**	**Marks the end of a complete programming statement**

Escape Sequences (Backslash Commands)

Sequence	Name	Meaning
\n	New line	Moves to beginning of next line
\t	Horizontal tab	Moves to next tab position Tab spacing is every 8 columns starting with 1. (Columns 9, 17, 25, 33, 41, 49, 57, 65, 73 ...)
\\	Backslash	Displays an actual backslash
\'	Single quote	Displays an actual single quote
\"	Double quote	Displays an actual double quote

```
System.out.println("\tGeorge\t\tPaul");
```

will tab, print *George*, tab twice more, and print *Paul*.

```
George          Paul
```

Console Output

- The `print` statement works very similarly to the `println` statement.
- However, the `print` statement does not put a newline character at the end of the output.
- The lines:

```
System.out.print("These lines will be");
System.out.print("printed on");
System.out.println("the same line.");
```

Will output:

```
These lines will beprinted onthe same line.
```

Notice the odd spacing? Why are some words run together?

Variable Assignment and Initialization (1 of 6)

- In order to store a value in a variable, an *assignment statement* must be used.
- The *assignment operator* is the equal (=) sign.
- The operand on the left side of the assignment operator must be a variable name.
- The operand on the right side must be either a literal or expression that evaluates to a type that is compatible with the type of the variable.

Variable Assignment and Initialization (2 of 6)

```
// This program shows variable assignment.

public class Initialize
{
  public static void main(String[] args)
  {
    int month, days;

    month = 2;
    days = 28;
    System.out.println("Month " + month + " has " +
                       days + " Days.");
  }

}
```

The variables must be declared before they can be used.

Variable Assignment and Initialization (3 of 6)

```
// This program shows variable assignment.

public class Initialize
{
  public static void main(String[] args)
  {
    int month, days;

    month = 2;
    days = 28;
    System.out.println("Month " + month + " has " +
                       days + " Days.");
  }

}
```

Once declared, they can then receive a value (initialization); however the value must be compatible with the variable's declared type.

Variable Assignment and Initialization (4 of 6)

```
// This program shows variable assignment.

public class Initialize
{
  public static void main(String[] args)
  {
    int month, days;

    month = 2;
    days = 28;
    System.out.println("Month " + month + " has " +
                       days + " Days.");
  }

}
```

After receiving a value, the variables can then be used in output statements or in other calculations.

Variable Assignment and Initialization (5 of 6)

```
// This program shows variable initialization.

public class Initialize
{
  public static void main(String[] args)
  {
    int month = 2, days = 28;

    System.out.println("Month " + month + " has " +
                       days + " Days.");
  }

}
```

Local variables can be declared and initialized on the same line.

Variable Assignment and Initialization (6 of 6)

- Variables can only hold one value at a time.
- Local variables do not receive a default value.
- Local variables must have a valid type in order to be used.

```
public static void main(String [] args)
{
  int month, days; //No value given…

  System.out.println("Month " + month + " has " +
                          days + " Days.");
}
```

Trying to use uninitialized variables will generate a Syntax Error when the code is compiled.

Arithmetic Operators (1 of 2)

- Java has five (5) arithmetic operators.

Operator	Meaning	Type	Example
+	Addition	Binary	`total = cost + tax;`
-	Subtraction	Binary	`cost = total – tax;`
*	Multiplication	Binary	`tax = cost * rate;`
/	Division	Binary	`salePrice = original / 2;`
%	Modulus	Binary	`remainder = value % 5;`

Integer Division

- Division can be tricky.

  In a Java program, what is the value of 1/2?

- You might think the answer is 0.5…
- But, that's wrong.
- The answer is simply 0.
- Integer division will truncate any decimal remainder.

Operator Precedence

- Mathematical expressions can be very complex.
- There is a set order in which arithmetic operations will be carried out.

	Operator	Associativity	Example	Result
Higher Priority	- (unary negation)	Right to left	`x = -4 + 3;`	-1
	* / %	Left to right	`x = -4 + 4 % 3 * 13 + 2;`	11
Lower Priority	+ -	Left to right	`x = 6 + 3 – 4 + 6 * 3;`	23