

Basic Syntax

CS 110: Introduction to Computer Science



Before We Begin

Before getting started, we have to set up our computers to be able to run Java code.

If you are using your own laptop, get the Lecture "**Installing Java**" and **FOLLOW THOSE DIRECTIONS NOT THESE.**



About Java

- Most programming environments are split into two parts:
 - The language itself
 - The **Integrated Development Environment (IDE)**.
- The language is like a dictionary
- The IDE is like Word.
- You can't write a paper without both



Writing Code



Now we can start!

- During the first half of this semester, we're going to focus on learning how to program and use these eight concepts.

1. Sequential
2. Conditional
3. Iteration
4. Variables
5. Arithmetic
6. Input/Output
7. Arrays (Lists)
8. Functions



1. Sequential

- Code that is contained within a function executes in the order that it's written.
- Every line of code (with a few exceptions) ends with a semicolon ;

```
Magic.println("Hello World");  
Magic.println("Goodbye");  
Magic.println("Not yet");
```



Comments

- **Comments** are segments of your code that do not run.
- Comments are **extremely important** because they are notes to your future self about what you've done.
- Also, in your **future job**, comments are notes to your colleagues about what you've done.



Comment Grammar

- **Single line** comments start with `//` and ignore everything until the end of the line.
- **Multiline** comments start with `/*` and ignore everything until `*/`

```
Magic.println("This is run");  
//This is a single line  
Magic.println("This is run");
```

Single Line

```
Magic.println("This is run");  
/* This is a multiline comment  
This is part of the same one  
So is this */  
Magic.println("This is run");
```

Multiline



2. Conditional & 3. Iteration

- These are fairly large topics and we'll visit them in the next unit



4. Variables

- Variables are how we store data for later use.
- In Java, there are different **types** of variables depending on what you want to store. The main types we'll use are
 - Integers written as `int`
 - Rational numbers written as `double`
 - Strings of characters written as `String`.



Variable Example

```
class Starter {  
    public static void main(String[] args) {  
        int x=20;  
        double y= 2.5;  
        String hi = "Hello";  
        int z =x + 30;  
        Magic.println("The value of z is "+z);  
    }  
}
```



Declaring a variable

- Before a variable is used it must be declared.

`type name;`

- In each function, every variable name **must be unique!**



Variable Name: Rules

- 1.No Spaces!
- 2.Only letters, numbers, and underscore _
- 3.First character needs to be a letter.



Naming Variables

- Part of producing well documented code is picking good variable names.
- Think of variables like nouns.
- When possible, you should pick descriptive names.

```
String greeting = "Hello";
```

Good

```
String b = "Goodbye";
```

Bad



camelCapitalization

- To use multiple words in a variable name, we use **camel capitalization**, where the first letter of every word is capitalized instead of using a space.
- By convention, **the first letter in variables is lower case**.

```
double waterTempInCelsius = 2.3;
```



Assigning Values

- To assign a value to a variable you use the = sign.
- Three rules
 1. Variable must match type assigned to it
 2. The variable that is assigned must be on the **left hand side** of the =.
 3. Variables must be assigned a value at least once before using it elsewhere.
- As a shortcut, you can assign a variable when you declare it.
- **Let's go through some examples.**



Using Variables

- Variables, once declared and assigned, can be used like a number in any expression as long the type is correct.
- **Let's go through simple examples using integers and doubles.**



Types

- Can you easily convert 20 from an integer to a rational number?
 - Yes, it becomes 20.0
- Can you easily convert 3.4 from a rational number to an integer?
 - Not without losing some information.
- To prevent mistakes Java allows us to **convert** between types



Explicit Conversion

- **Explicit conversion** is when tell Java specifically, you want to convert between types.
- The syntax for this is like so

`(new-type) value`

- For example, if we wanted to convert 4.3 to an integer:

`(int)4.3`

- Conversions from doubles → integers always round down.
- **Let's go through some examples.**



Implicit Conversion

- When you convert between doubles to integers you always have to use explicit conversion.
- When you convert between integers to doubles you CAN use explicit conversion. Or not.
- This is called **implicit conversion**.
- You can only use this if **no information will be lost**.
 - This is why you can't use it from double to integer



5. Arithmetic

- The basic arithmetic operators are all here: $+$, $-$, $*$, $/$, and $()$.
- Order of operations still applies like normal math.
- There isn't an exponent built in, but we'll talk about how to use it later.



Division

- What is $4/2$?
- What is $5.0/2.0$?
- What is $5/2$?
- Division between integers will produce an integer and **ALWAYS ROUND DOWN.**



Remainder

- Java also has an operation that is designed for integers.
- The **modulus** operator calculates the remainder of two integers.
- The modulus operator is `%` and has the same order of operations as division.
- So, `12%4` is 0, `13%4` is 1, `14%4` is 2, `15%4` is 3, and `16%4` is 0.
- **Hint:** `%2` is how to determine if a number is even or odd.



String Math

- Java lets you add strings together.
- You can even add numbers to Strings.
- This is called **concatenation** and is a great way to build up larger more complicated strings.
- **Let's go do some examples**



	int	double	String
+	addition	addition	concatenate
-	subtract	subtract	N/A
*	multiply	multiply	N/A
/	integer division	true division	N/A
%	remainder	remainder	N/A
()	order of operations	order of operations	order of operations



Modifying a Variable

- Frequently, we want to increase the value of a variable we could write this

```
x = x + 30;
```

- But this too long and is prone to typos. So, there is a shortcut

```
x += 30;
```

- This works for all operands

```
x += 30;  
x -= 20;  
x *= 2;  
x /= 3;  
x %= 7;
```



Shortcut

- Adding and subtracting one are so common, there is an even shorter version of those:

```
x = x+1;  
x += 1;  
x++;
```

```
x = x-1;  
x -= 1;  
x--;
```



Helpful Hints (Keyboard Shortcuts)

- Here are a few keyboard shortcuts to make life better
- **Command (⌘) + C**: Copy
- **Command (⌘) + V**: Paste
- **Command (⌘) + X**: Cut
- **Shift (⇧) + →/←**: Select text
- **Option (⌥) + →/←**: Skip right or left a word
- **Command (⌘) + →/←**: Skip to start or end of line
- **Option (⌥) + Shift (⇧) + →/←**: Skip words and select
- **Command (⌘) + Shift (⇧) + →/←**: Skip to start or end of line and select
- **Command (⌘) + /**: Comment/Uncomment selected lines.

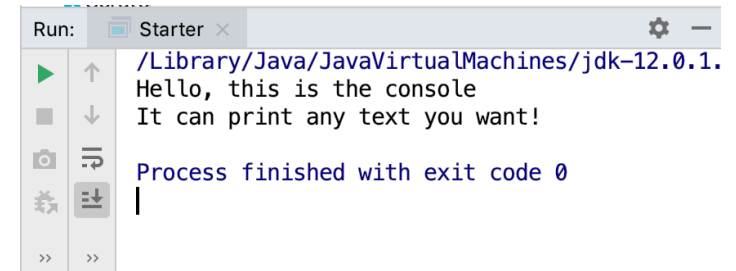


6. Input/Output

- Input and Output can take many forms.
- It can be reading from and writing to files
- It can be pictures and movies
- It can be 3D Graphics
- It can be **Graphical User Interfaces (GUI)**
- It can be printing
- For right now, it will be **the console**.
- (Don't worry, we'll draw too. Just not now)



The Console

A screenshot of an IDE's console window. The title bar says 'Run: Starter x'. The console output shows the path '/Library/Java/JavaVirtualMachines/jdk-12.0.1.' followed by two lines of text: 'Hello, this is the console' and 'It can print any text you want!'. Below this, it says 'Process finished with exit code 0' and a cursor is visible on the next line. The left sidebar contains icons for running, stepping through, and other debugging actions.

```
Run: Starter x
/Library/Java/JavaVirtualMachines/jdk-12.0.1.
Hello, this is the console
It can print any text you want!
Process finished with exit code 0
|
```

- The Console, sometimes called **standard input** or **standard output**, is where we get most of our simple input output.
- The console use to be VERY important for I/O, but has been largely replace by GUIs.
- The main uses today are
 - Programmers using it for simple I/O for themselves or other programmers (That's our use).
 - Specific utilities called **terminal applications** that we won't cover until 400 level classes.



Console Input

- To allow the user to input values, we will use three functions
 - `Magic.nextInt()`
 - `Magic.nextDouble()`
 - `Magic.nextLine()`
- **Note:** These only GET input. If you use them without a prompt it will look like your code is broken.
- **Let's go try these out and see how they work.**



Output: Drawing

- We can draw using Magic!

```
Magic.drawRectangle(20,30,40,50, "red");  
Magic.drawEmptyRectangle(60,70,50,50, "blue");  
Magic.drawOval(100,100,20,20, "green");  
Magic.drawLine(200,200,250,300, "black");  
Magic.drawEmptyOval(400,200,22,33, "cyan");
```

- Colors we can use are: Red, Blue, Green, Black, White, Magenta, Cyan, Gray, DarkGray, Yellow, Pink
- *We'll learn animation and better drawing in the second half of the semester.*



7. Arrays

- We'll talk about arrays after we cover iteration in the next unit.
- For now, all you need to know about arrays is that they are just lists of values.



8. Functions

- Functions will deserve an entire unit on its own and we'll cover all of the other 7 concepts fully before discussing them in detail.
- For now though, just remember that in Java Functions are where all of the code we want to execute will live.



Next Steps

- In this unit we covered the basic concepts you'll need for the first half of this class.
- Remember, the challenge of programming is learning how to take these basic concepts and use them to build something larger.
- You'll be surprised what you can do.

