COMP10062: Week 1 Guide

Sam Scott, Mohawk College, 2021

0. Reading for this Week

You should look at the textbook sections for each week, preferably before we cover the material in class. For this week, you should **read sections 1.2, 1.4, and 2.1 to 2.4** (omit the optional sections of 2.3).

1. Hello, World! (Section 1.2)

The world's most basic computer program…

In Python	In Java

`print("Hello, World!")`	`public class Hello {`
	`    public static void main (String[] args) {`
	`        System.out.println("Hello, World!");`
	`    }`
	`}`

Notes

Java is an **object-oriented language**. It requires a **class** definition and a **main method** in order to execute the print statement. (A "method" is a function.) Python is an **imperative** language. It can execute the print statement without all the extra syntax.

Java code blocks use **braces** { }. Indenting is important for human readability, but it has no meaning in the language (see the **Documentation Standards** handout on Canvas).

IntelliJ Tips:

Try the psvm-TAB and sout-TAB shortcuts.

Use CTRL-ALT-L to automatically indent your code.

A Pause Statement

The `Thread.sleep()` method causes the program to pause for the number of milliseconds you specify. A millisecond is a thousandth of a second If you want to pause for half a second, you would use:

```
Thread.sleep(500);
```

A program that uses `Thread.sleep()` must make the following change to the main method header:

```
public static void main(String[] args) throws InterruptedException
```

Later in the course, you'll learn more about what "`throws InterruptedException`" means.

Commenting

In Java, you use // to create a single line comment. It works exactly like # in Python.

You can also create comments that span multiple lines like this:

```
/* multi-line
comment */
```

In this course, we will use multi-line comments in JavaDoc format. See the **Documentation Standards** handout on Canvas for more on how to create JavaDoc comments properly.

2. Variables, Input and Output (Section 2.1, 2.3)

In Python

	- No variable declaration
`x = 5`	- This is ok
`x = "5"`	- This is also ok
`y = 2.3`	- This is also ok
	Python is dynamically-typed.

In Java

`int x;`	- x declared to hold integers only
`x = 5;`	- This is ok
`x = "5";`	- This is a syntax error
`y = 2.3;`	- This is also a syntax error (y not declared)
`double y;`	- y declared to hold decimal numbers
`y = 2.3;`	- This is ok
	Java is statically-typed.

In Python

```
x = input("x: ")
xint = int(x)
print(xint)


name = input();
print("hi", name)
```

In Java

```
Scanner input = new Scanner(System.in);
int x;

System.out.print("x: ");
x = input.nextInt();
System.out.println(x);

String name;
name = input.next();
System.out.println("hi " + name);
```

Java Notes

Keyboard input in Java is accomplished with a `Scanner` object

- Must be declared and created before it can be used (see first line of Java code above)
- Must have "`import java.util.Scanner`" as the first line of the program
- Use `nextInt()` and `nextDouble()` for numeric input
- Use `next()` to get a single word, or `nextLine()` to get an entire line[1]

IntelliJ Tips:

Put the cursor on a red class name and press Alt-Enter to select and import the class.

Use CTRL-ALT-O to remove unused imports.

`System.out.println()` is like Python's `print` except it only takes a single argument

- Concatenate with + to create a single line of output from multiple variables
- `System.out.print()` prints without a carriage return (like Python's `print` with `end=""`)

Formatting output (i.e. justifying it, rounding to a fixed number of decimal places, etc.) is accomplished using the `System.out.printf()` method. For more on that, see the optional part of section 2.3 of the textbook, as well as the sample code for this week.

[1] There are some complications involved when you use `nextLine()`. See the *Gotcha* in Section 2.3 of the text.

3. Graphical Output (Section 1.4)

Where to Get Stuff

- See section 1.4 of the text for a full discussion of how to do graphical output.
- Get **FXGraphicsTemplate.java** from Canvas for starter code.
- **IMPORTANT:** When you create a Java project in IntelliJ, don't select "JavaFX". It will create a project that is not configured for easy use by beginners. Always select "Java".

More than Just Curves

The textbook waits until chapters 3, 4 and 5 to give you anything more than black ovals and arcs. But that's no fun. Here's a quick roundup on how to use colors and draw text and other shapes as well. See section 5.4 of the textbook for more.

Ovals and Arcs

```
gc.strokeOval(x, y, width, height);
gc.fillOval(x, y, width, height);
gc.strokeArc(x, y, width, height, startAngle, arcAngle, arcType);
gc.fillArc(x, y, width, height, startAngle, arcAngle, arcType);
```

- `arcType` can be `ArcType.OPEN`, `ArcType.CHORD` or `ArcType.ROUND`

Rectangles, Lines and Text

```
gc.strokeRect(x, y, width, height); gc.strokeText(string, x, y);
gc.fillRect(x, y, width, height);   gc.fillText(string, x, y);
gc.strokeLine(x1, y1, x2, y2);
```

Style

```
gc.setStroke(color);                     gc.setLineWidth(width);
gc.setFill(color);                       gc.setFont(font);
```

Colors

`Color.BLACK`, `Color.RED`, `Color.BLUE`, etc.

- Get the full list by typing "`Color.`" Into IntelliJ and waiting for the pop-up.

`Color.rgb(red, green, blue, opacity)`

- red, green and blue are integers in the range 0 to 255.
- opacity is an *optional* double in the range 0.0 to 1.0

`Color.web("css color")`

- Any CSS color will work

```
import javafx.scene.paint.Color;
```

- **GOTCHA:** ALT-Enter might import `java.awt.Color` instead. Make sure you get the right one.

Fonts

```
new Font("font name", size);
```

- Use the "System" font or a web-safe font to be sure the user has it (https://www.w3schools.com/cssref/css_websafe_fonts.asp).

```
import javafx.scene.text.Font;
```

- **GOTCHA:** ALT-Enter might import `java.awt.Font` instead. Make sure you get the right one.

Full Documentation

Look up the `GraphicsContext`, `Color` and `Font` classes in the JavaFX API

4. Data Types and Expressions (Section 2.1, 2.4)

In Python		In Java	
		`int a;`	
`a=754839758943793`	- Unbounded integers	`a=754839758943793;`	- Syntax error
`b='$'`	- A string	`char b = '$';`	- Single quotes = char
`c="$"`	- Also a string	`String c = "5";`	- Double quotes = String
`d='Hello!'`	- Yet another string	`String d = 'Hello!'`	- Syntax error

Notes

- In Java, data types come in two forms: **Primitive** and **Object**.
- Primitive data types represent single values. They have no attributes or methods.
    - There are 8 primitive types, listed in section 2.1 of the textbook.
- Object data types represent chunks of data (**fields**) with code attached (**methods**).
    - There are thousands of these. The most important for now are `String`, `Scanner`, and `GraphicsContext`.

Arithmetic Expressions

In Python	In Java	Java Notes
`x = 5`	`int x = 5;`	- combined declaration and assignment is allowed in Java
`y = "hi"`	`String y = "hi";`	
`a = x // 2;`	`int a = x / 2;`	- a = 2
`b = x / 2`	`double b = x / 2.0;`	- b = 2.5
`c = x ** 2;`	`int c = (int)Math.pow(x,2);`	- c = 25
`d = y + " there"`	`String d = y + " there";`	- d = "hi there"
`e = y + str(x)`	`String e = y + x;`	- e = "hi5"
`f = y * 5`	`String f = y * 5;`	- Syntax error in Java

More Java Notes

Most arithmetic that works in Python will work the same way in Java (`+`, `-`, `*`, `/`, `%` and brackets), but:

- Java's `/` operator is equivalent to python's `//` when both operands are integers.
- Java has no `**` operator. Use `Math.pow()` instead (always returns a `double`).
- To convert types in Java, put the type name in brackets in front of the expression. This is known as **type casting** or just "casting" for short.
- Java does automatic type casting for `String` concatenation.
- Java does not support * for `Strings`.

The Math Class: See section 6.2 of the text or the Java 8 API for more info on the many useful math methods provided by this class.

Other important considerations

- 5 is an `int`, 5.0 is a `double`
- Java will automatically cast an `int` as a `double` in order to do arithmetic, assignment, etc.
- Java cannot automatically cast a `double` as an `int`.
- For arithmetic operations, if one operand is a `double`, the result will also be a `double`.

Named Constants

- Use the `final` keyword when declaring a variable to make a named constant.
- Use named constants instead of literal values (see **Documentation Standards** on Canvas).
- Most programmers use `ALL_CAPS` when naming constants.

5. String Manipulation in Java (Section 2.2)

Strings are objects in java. `String` is not a primitive data type. However the `String` data type is so basic and important to most programming that there is special support in the language for it.

In addition to concatenating strings with the + operator, there are also a number of **methods** that you can call on any given String. A method is just a function that is attached to an object.

Page 88 of the textbook gives a rundown of various useful `String` methods. For a bigger (and harder to read) list, you can go to the Java API.

IntelliJ Tip: To get code from eLearn to IntelliJ, first unzip it, then find the unzipped week1examples folder, copy it and paste it into the SRC folder of the IntelliJ Project window.

Other Cool Stuff (All Totally Optional)

Input Dialogs

Section 2.5 of the textbook introduces the `JOptionPane` class. This is an easy way to do input and output using dialogs instead of `System.out.println()` and the `Scanner`. It can be a nicer way to do I/O when you're running a JavaFX program.

Formatted Output

Section 2.3 has a section on how to use `System.out.printf()` to do nicely formatted output (rounding doubles, justifying text, etc.). For drawing text in a JavaFX application, the `String.format()` method works the same way as `System.out.printf()` but it returns a string instead of sending it to standard output.

Dr. Java

Do you miss being able to execute code in the Python Shell? If so, go get Dr. Java. It's a stripped-down development environment with an "interactions pane" that you can type commands into just like the Python shell. The best part is, it's a single executable file. There's nothing to install and you can run it off a USB key.

Useful Links

The Java API (for info on String, Scanner, Thread, etc.):
https://docs.oracle.com/en/java/javase/15/docs/api/index.html

The Java FX API (for info on GraphicsContext, Color, Font, etc.):
https://openjfx.io/javadoc/15/

Dr. Java (for an "interactions pane" similar to the Python Shell):
http://www.drjava.org