

Printing to the screen

Use standard method calls to print to the screen.

WE'LL COVER THE FOLLOWING

- Exercise: Beatles
- String concatenation
- Special characters: printing quotes, newlines, tabs, etc.
 - Exercise: translate from Python
- Formatted printing with `format`

`print` commands aren't the most exciting thing in coding, but eventually, you'll want to print something out, even if only for debugging or testing your code. You won't find anything too surprising in Java's methods for printing, but there are some slight differences from C, Python, and Javascript.

You can print the text "Hello, World!" to the screen using a the method call `System.out.println("Hello, World!");`.

Why so much typing for a simple print? Frequently, method calls require an **object** to act on in Java. In the method call `c.circle(150, 100, 25);` from the first section, the method `circle` is called to draw a circle, acting on the object referenced by `c`, a `Canvas` object that we first had to create.

`System.out` is also the name of an object: the main printed output device for your program. So `System.out.println` calls the method `println` on the `System.out` object.

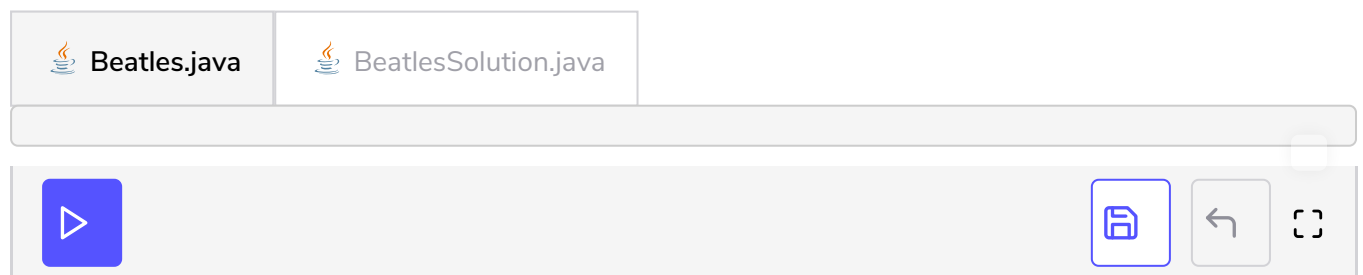
`println` requires one parameter, which should be a string of text, like `"Hello, World"`, or something that Java can convert to a string, like `6`. Strings are marked by double quotes `"` in Java.

Exercise: Beatles

As practice writing a complete program, write a program that prints out the text:

```
Hello, hello.  
I don't know why you say goodbye.  
I say hello.
```

You'll need a class name that matches the name of the file, a `main` function (don't forget the `public static void` or the required parameters), and a few `println` calls. Run your code to make sure it works, and then take a look at the solution in the second tab.



note: The `System.out.println` method prints a newline after the text. Use `System.out.print` if you don't want the newline.

String concatenation

Strings can be concatenated in Java using the `+` operator.

`System.out.println("My name is " + "Inigo Montoya")` first concatenates the two strings, and then prints the result. As long as one of the items is a string, the two items are converted to strings before concatenation. So

```
System.out.println("My favorite number is " + 42);
```

converts `42` to a string, concatenates, and prints. Order of operations matters. What would the following print?

```
System.out.println(42 + 22 + " is my favorite number.");
```

Special characters: printing quotes, newlines, tabs, etc.

Just like in Python, Javascript, or any other language influenced by C, you can




use backslash codes. `\n` will give you a newline, and `\t` will give you a tab.

In some languages, like Python, you can use either single, double, or triple quotes to create a string, and a clever choice will allow you to print quotes of a different type within a string. Java doesn't have this capability, so if you want to print double quotes, you'll need the code `\"`.





Exercise: translate from Python

The first tab below has some Python code that prints a quotation from a well-known novel. Run it. In the second tab, write a Java program that prints the same output. It's ok if you don't know Python – you should be able to see the intent of the code by running it. A few things to note:

1. Every Java program needs a class and a `main` method.
2. Python concatenates adjacent string literals. Java does not. If you want to split the long line into multiple lines, you'll need to concatenate several strings, as done in the sample solution.
3. Python's `print` statement automatically prints a newline; `System.out.println` is the equivalent.

 ged.py	 Ged.java	 GedSolution.java
--	--	--

```
# Python code to print some dialog from a novel.
print('"Good," said the boy, for he had no wish '
      'to tell the secret to his playmates, '
      'liking to know and do what they knew not and could not.')
```



Formatted printing with `format`

Java provides a method `format` that works like the `printf` function in `C`, or like the string substitution operator `%` in Python. If you know how one of those works, just skip to the example at the end of this section. Otherwise, read on.

Sometimes, you'd like to print out a combination of text data and values, but you'd like to keep your code clean and readable without a lot of string


concatenation. For example, maybe you'd like to print out the value of π . The

`format` method will let you use a *format string* as the first parameter, and substitute later parameters into that string.

```
System.out.format("%f is an approximation of pi.", 3.14159).
```

The `%f` is a *format specifier* marks the location where the parameter `3.14159` should be substituted. The letter `f` in `%f` indicates that the value will be a floating point number. For an integer, use `%d`.

Sometimes you'd like your output to be formatted nicely. You can do things like round the number as it is inserted, or pad the number with spaces. A format specifier like `%9.2f` would print a floating point number with 2 digits after the decimal, left-padded with spaces to take up 9 characters (including the decimal). A few examples:

 `printExamples.java`

```
class printExamples {  
    public static void main(String args[]) {  
        System.out.format("My favorite number is %.3f.\n", 3.141592654);  
  
        // part of a times table:  
        System.out.format("%2d %2d %2d\n", 4, 6, 8);  
        System.out.format("%2d %2d %2d\n", 8, 12, 16);  
    }  
}
```

