

Lecture 33:

INTRODUCTION TO JAVA

CSC111: Introduction to CS through Programming

R. Jordan Crouser

Assistant Professor of Computer Science

Smith College

Material for this lecture based on:

<http://interactivepython.org/runestone/static/java4python/index.html>

Final Project “Miniseries”

- **Typical semester:** big reception in Ford Atrium where teams demonstrate their final projects to CS Department



CSC111: Final Project Demos

Final Project “Presentations”

- ~~Typical semester: big reception in Ford Atrium where teams demonstrate their final projects (as seen in video)~~
- **Pandemic version:**
 - Too many projects to do “live” presentations via Zoom
 - Instead, FP3: Prototype II will be a **5-minute video** which we’ll combine to make a multi-part “**miniseries**”
 - The week following break will be **fully asynchronous** (you can watch the “episodes” at your convenience)

What to include in your video

- A brief overview / description of your project
 - Describe the “big idea”
 - A sentence or two about the intended audience
 - Great place to include pictures of your paper prototype / persona
- An overview of the **code**
 - Pull up your repl and give us a quick tour ☺
 - Did you use object-oriented programming, or something more function-oriented? Tell us why!
- The best part: a **demonstration** of your project in action
 - Could be you giving us a walkthrough, or a recording of someone else interacting with it (just make sure you have their permission!)
 - If you choose the latter, you’re welcome to include their feedback in your video (e.g. “I really like this feature”, etc.)

Final Project “Presentations”

- ~~Typical semester: big reception in Ford Atrium where teams demonstrate their final projects (as seen in video)~~
- **Pandemic version:**
 - Too many projects to be able to do “live” presentations via Zoom
 - Instead, FP3: Prototype II will be a **5-minute video** which we’ll combine to make a multi-part “miniseries”
 - The week following break will be **fully asynchronous** (you can watch the “episodes” at your convenience)
- Since we will be asynchronous anyway, we’ll push the due date for FP3 to **Tuesday Dec. 1st at 11:59pm EST** (roughly +2 additional days to work on it if you don’t finish it before break)

Discussion

What's good about Python as a
programming language?



Discussion

Why bother studying **more than one**
programming language?



Dynamic vs. formal

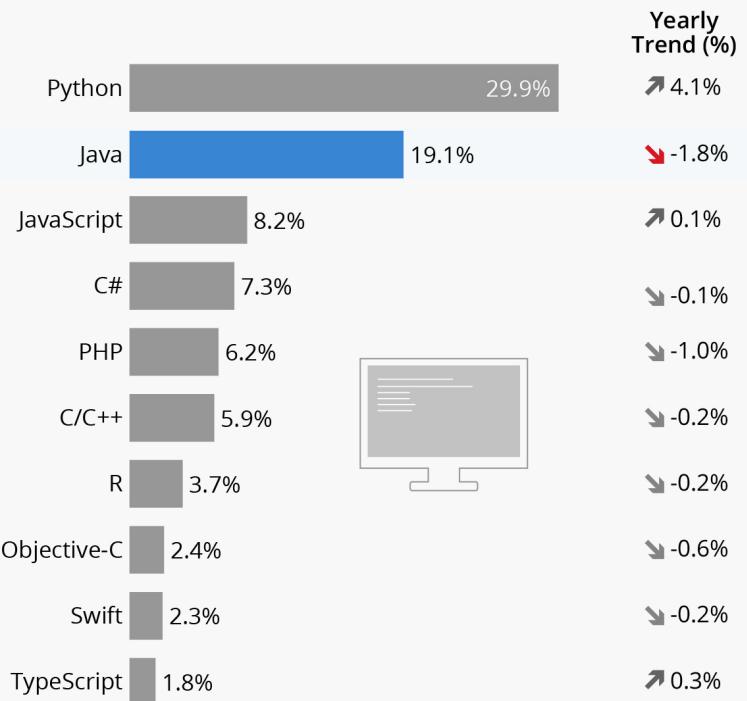


Why Learn Java?

- Java is one of the most widely taught languages (including in CSC212!!)
- Java is an **industrial strength** language that can be used to build large systems by large groups
- If you know Java learning C++, etc. is easy!

Python Remains Most Popular Programming Language

Popularity of each programming language based on share of tutorial searches in Google



Yearly trend compares percent change from Feb 2019 to Feb 2020
Sources: GitHub, Google Trends



Let's compare...

multi-paradigm
interpreted language
with dynamic typing
and automatic memory management



Let's compare...

an object-oriented
compiled language
with strong static typing
and automatic memory management



First big difference

an object-oriented
compiled language
with strong static typing
and automatic memory management



Compiled vs. interpreted languages

	COMPILER	INTERPRETER
What it takes in:	an entire program	a single line of code (or a single instruction)
What it returns:	intermediate object code (e.g. classes)	<nothing>
Relative speed:	faster	slower
Memory usage:	uses more memory (↑ objects take space ↑)	uses less memory (no intermediate objects)
Work is done:	just once	every time the code is executed
Reports errors:	after checking the entire program	after each instruction is run



Next big difference

an object-oriented
compiled language
with strong static typing
and automatic memory management



Example: hello world

```
def main():
    print("Hello world!")
```

Example: hello world

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

Several similarities...

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

Several similarities...

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



a built-in function
to output stuff to the console

Some minor differences...

curly braces indicate containment
(instead of tabs)

```
public class Hello {
```



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

```
}
```

```
}
```

(in fact, we could any Java program on a single line!)

Some more significant differences...

can't do anything
without a class



```
public class Hello {
```

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

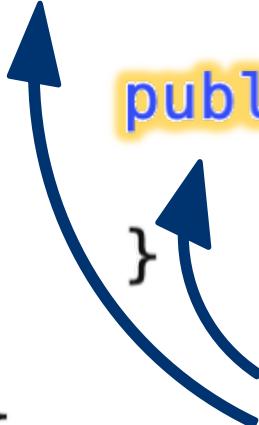
```
}
```

(in fact, we could any Java program on a single line!)

Some more significant differences...

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

seems like we have to give
a bunch more information
about our function / class definitions



(in fact, we could any Java program on a single line!)

Discussion

So how to we **run** it?



Two-step process

- Remember: java is a **compiled** language
- Step 1: compile



A screenshot of a Mac OS X terminal window titled "demos — bash — 43x5". The window shows the command "tardis:demos jcrouser\$ javac Hello.java" being typed in.

- Step 2: run



A screenshot of a Mac OS X terminal window titled "demos — bash — 43x5". The window shows the command "tardis:demos jcrouser\$ java Hello" followed by the output "Hello World!".

Basic rules of Java

1. Every Java program must define a **class**:
 - all code is inside the class
 - the file has the same name as the class
 2. Everything in Java must have a **type** (nothing is **assumed**)
 3. Every Java program must have a function called
- ```
public static void main(String[] args)
```

see why we developed  
this habit early?



# Java: be specific!

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

# Java: be specific!

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



we can call this method  
from **outside** the class

# Java: be specific!

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



the method belongs to the class,  
but not any **one instance**

# Java: be specific!

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

the method doesn't  
**return anything**



# Java: be specific!

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

the method takes an  
**array (list) of Strings**  
as input



# And the line that does the actual work?

```
System.out.println("Hello world!");
```

# And the line that does the actual work?

```
System.out.println("Hello world!");
```



this is a  
built-in **object**

# And the line that does the actual work?

```
System.out.println("Hello world!");
```



that object  
has an **attribute**  
(which is also an object)

# And the line that does the actual work?

```
System.out.println("Hello world!");
```



that object  
has a **method**

# And the line that does the actual work?

```
System.out.println("Hello world!");
```

every statement ends  
with a **semi-colon**



# Basic data types in Java

| Primitive | Object  |
|-----------|---------|
| int       | Integer |
| float     | Float   |
| double    | Double  |
| char      | Char    |
| boolean   | Boolean |

# A slightly more interesting past program

```
def main():
 F = int(input("Enter the temperature in F: "))
 C = (F - 32) * 5.0/9.0
 print("The temperature in C is: ", C)

if __name__ == "__main__":
 main()
```

# The same thing in Java

```
import java.util.Scanner;

public class TempConv {
 public static void main(String[] args) {
 Double F;
 Double C;
 Scanner in;

 in = new Scanner(System.in);
 System.out.println("Enter the temperature in F: ");
 F = in.nextDouble();

 C = (F - 32) * 5.0/9.0;
 System.out.println("The temperature in C is: " + C);

 System.exit(0);
 }

}
```

# The same thing in Java

```
import java.util.Scanner;

public class TempConv {
 public static void main(String[] args) {
 Double F;
 Double C;
 Scanner in;

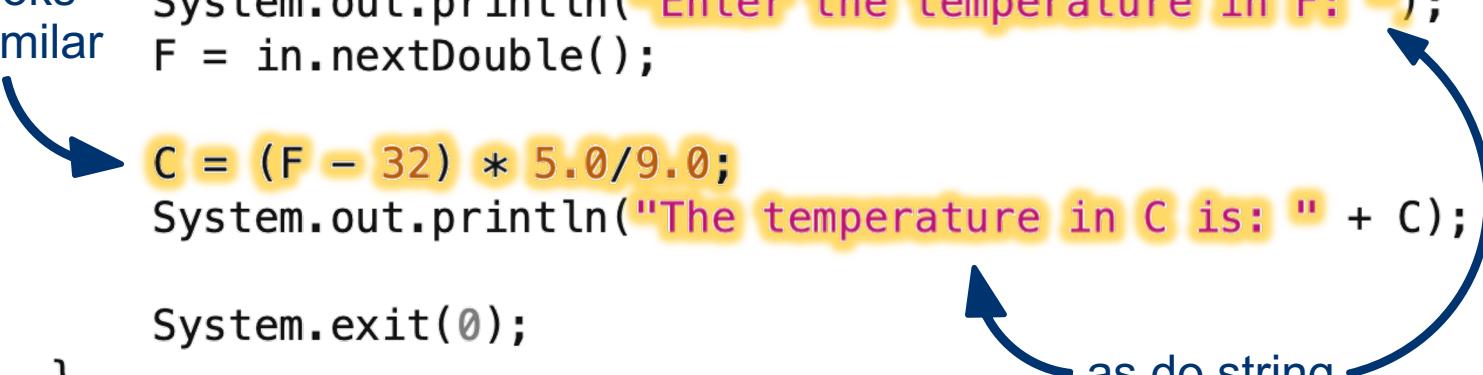
 in = new Scanner(System.in);
 System.out.println("Enter the temperature in F: ");
 F = in.nextDouble();

 C = (F - 32) * 5.0/9.0;
 System.out.println("The temperature in C is: " + C);

 System.exit(0);
 }
}
```

math looks pretty similar

as do string literals



# The same thing in Java

```
import java.util.Scanner;

public class TempConv {
 public static void main(String[] args) {
 Double F;
 Double C;
 Scanner in;

 in = new Scanner(System.in);
 System.out.println("Enter the temperature in F: ");
 F = in.nextDouble();

 C = (F - 32) * 5.0/9.0;
 System.out.println("The temperature in C is: " + C);

 System.exit(0);
 }
}
```

importing works roughly the same way  
(but there are WAY MORE built-in classes,  
and they're organized hierarchically!)

# The same thing in Java

```
import java.util.Scanner;

public class TempConv {
 public static void main(String[] args) {
 Double F;
 Double C;
 Scanner in;
 like
 input(...)

 in = new Scanner(System.in);
 System.out.println("Enter the temperature in F: ");
 F = in.nextDouble();

 C = (F - 32) * 5.0/9.0;
 System.out.println("The temperature in C is: " + C);

 System.exit(0);
 }

}
```



# The same thing in Java

```
import java.util.Scanner;

public class TempConv {
 public static void main(String[] args) {
 Double F;
 Double C;
 Scanner in;

 in = new Scanner(System.in);
 System.out.println("Enter the temperature in F: ");
 F = in.nextDouble();

 C = (F - 32) * 5.0/9.0;
 System.out.println("The temperature in C is: " + C);

 System.exit(0);
 }

}
```

but the  
prompt  
is separate



# The same thing in Java

```
import java.util.Scanner;

public class TempConv {
 public static void main(String[] args) {
 Double F;
 Double C;
 Scanner in;
 as is the call
 to get the
 actual input
 ↘
 in = new Scanner(System.in);
 System.out.println("Enter the temperature in F: ");
 F = in.nextDouble();
 C = (F - 32) * 5.0/9.0;
 System.out.println("The temperature in C is: " + C);

 System.exit(0);
 }

}
```

# Declaring variables

```
import java.util.Scanner;

public class TempConv {
 public static void main(String[] args) {
 Double F;
 Double C;
 Scanner in;

 unlike python,
 we have to
 tell Java
 in advance
 what type
 each variable
 will be
 in = new Scanner(System.in);
 System.out.println("Enter the temperature in F: ");
 F = in.nextDouble();
 C = (F - 32) * 5.0/9.0;
 System.out.println("The temperature in C is: " + C);

 System.exit(0);
 }

}
```

# Declaring variables

```
import java.util.Scanner;

public class TempConv {
 public static void main(String[] args) {
 Double F;
 Double C;
 Scanner in;

 unlike python,
 we have to
 tell Java
 in advance
 what type
 each variable
 will be
 in = new Scanner(System.in);
 System.out.println("Enter the temperature in F: ");
 F = in.nextDouble();
 C = (F - 32) * 5.0/9.0;
 System.out.println("The temperature in C is: " + C);

 System.exit(0);
 }

}
```

# Cleaning up

```
import java.util.Scanner;

public class TempConv {
 public static void main(String[] args) {
 Double F;
 Double C;
 Scanner in;

 in = new Scanner(System.in);
 System.out.println("Enter the temperature in F: ");
 F = in.nextDouble();

 C = (F - 32) * 5.0/9.0;
 System.out.println("The temperature in C is: " + C);

 System.exit(0);
 }
}
```



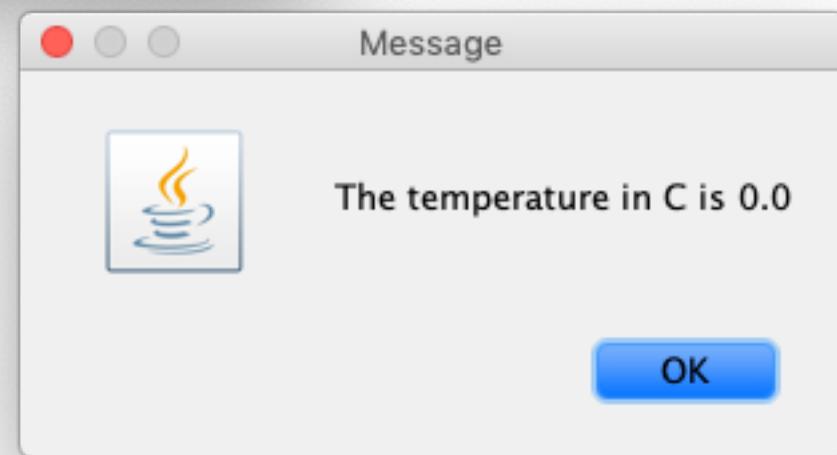
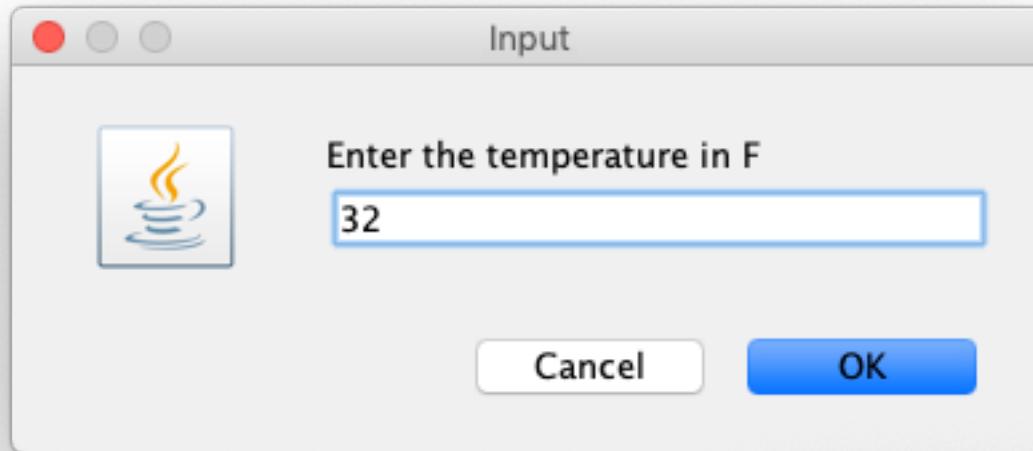
it's generally good practice to  
close the JVM when you're done  
(the 0 means “zero errors”)

# Discussion

Does this look like any **Java applications**  
you've encountered?



# A more realistic Java program



# Not a problem!

```
import javax.swing.*;

public class TempConvGUI {

 public static void main(String[] args) {
 String fString;
 Double F, C;

 fString = JOptionPane.showInputDialog("Enter the temperature in F");
 F = new Double(fString);
 C = (F - 32) * 5.0/9.0;

 JOptionPane.showMessageDialog(null, "The temperature in C is " + C);
 System.exit(0);
 }
}
```

# Not a problem!



javax.swing is the Java equivalent  
of tkinter (graphics)

```
import javax.swing.*;

public class TempConvGUI {

 public static void main(String[] args) {
 String fString;
 Double F, C;

 fString = JOptionPane.showInputDialog("Enter the temperature in F");
 F = new Double(fString);
 C = (F - 32) * 5.0/9.0;

 JOptionPane.showMessageDialog(null, "The temperature in C is " + C);

 System.exit(0);
 }
}
```

# Not a problem!

```
import javax.swing.*;

public class TempConvGUI {

 public static void main(String[] args) {
 String fString;
 Double F, C;

 fString = JOptionPane.showInputDialog("Enter the temperature in F");
 F = new Double(fString);
 C = (F - 32) * 5.0/9.0;

 JOptionPane.showMessageDialog(null, "The temperature in C is " + C);

 System.exit(0);
 }
}
```

which gives us input dialog boxes

# Not a problem!

```
import javax.swing.*;

public class TempConvGUI {

 public static void main(String[] args) {
 String fString;
 Double F, C;

 fString = JOptionPane.showInputDialog("Enter the temperature in F");
 F = new Double(fString);
 C = (F - 32) * 5.0/9.0;

 JOptionPane.showMessageDialog(null, "The temperature in C is " + C);
 System.exit(0);
 }
}
```

 as well as  
output dialog boxes  
(and much, much more!)

# Big takeaways

- Just like spoken languages, there are **many** different programming languages - each **expressive** in its own way
- This semester, you learned **fundamental ideas** in computer science (using Python as a medium):
  - variables
  - loops
  - conditional statements
  - functions
  - ...and many more!
- These fundamental ideas are **still valid** in other languages!