Objectives

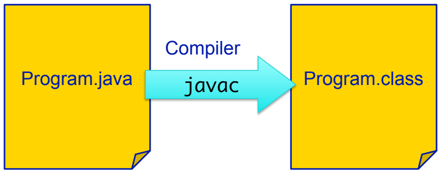
- Basics of Java Syntax
- Java fundamentals
  - Primitive data types
  - Static typing
  - Arithmetic operators
  - Relational operators

Review

- What are qualities of good software?
- What is Java?
  - Benefits to using Java?
- Linux:
  - What is the syntax of the `cp` command?
    - How do you copy an entire directory?
  - How do you make a directory?
  - How do you view the contents of a directory?

Review: Java Programming Language

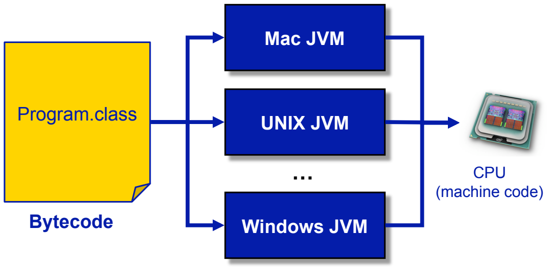
- Entirely object-oriented
- Similar to Python



Written in Java Programming Language

Bytecode: machine code for a virtual CPU

Review: Java Virtual Machine (JVM)



- Same **bytecode** executes on each platform

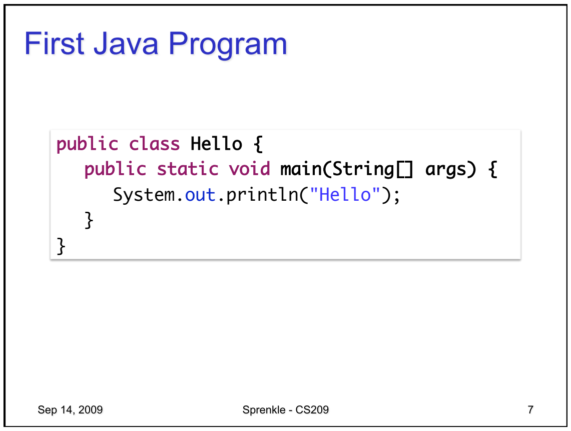
Review: Benefits of Java

- Rapid development of programs
  - Large library of classes, including GUIs, Enterprise-level applications, Web applications
- Portability
  - Run program on multiple platforms without recompiling
- Statically-typed language
  - Compiling - find some errors before execution, performance benefits

Python Review

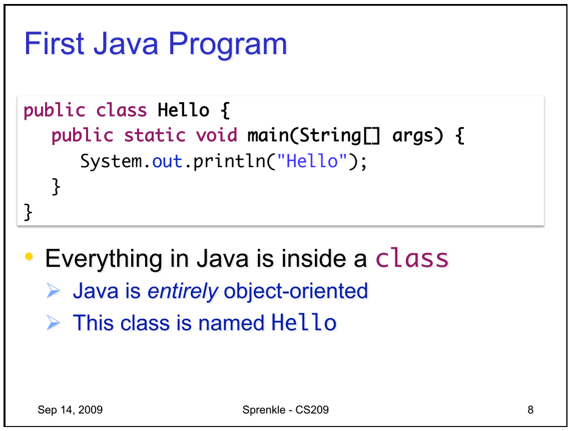
```
# a Python program
def main():
     print “Hello”

main()
```

- What does this program do?



First Java Program

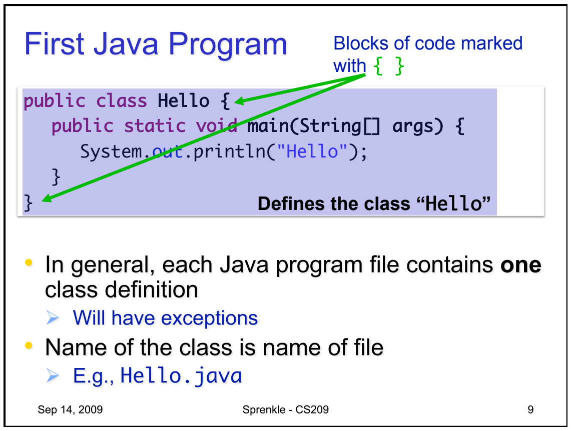
```
public class Hello {
   public static void main(String[] args) {
      System.out.println("Hello");
   }
}
```



First Java Program

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

- Everything in Java is inside a `class`
  - Java is *entirely* object-oriented
  - This class is named `Hello`



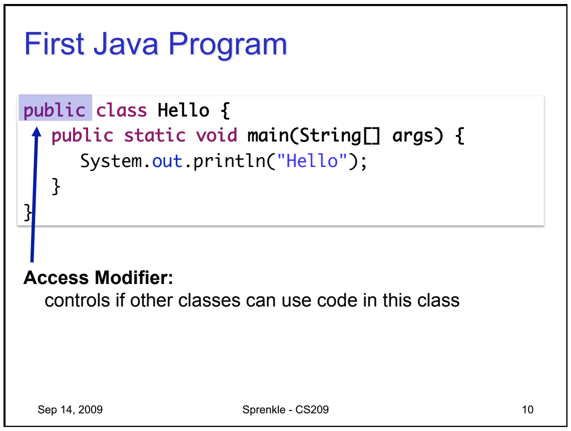
First Java Program

Blocks of code marked with `{ }`

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

Defines the class “Hello”

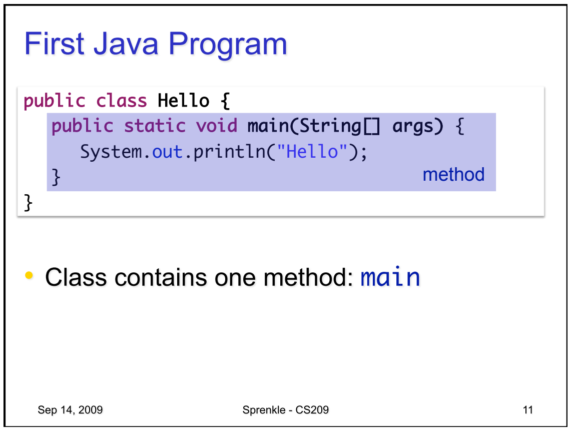
- In general, each Java program file contains **one** class definition
  - Will have exceptions
- Name of the class is name of file
  - E.g., `Hello.java`



First Java Program

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

Access Modifier:
controls if other classes can use code in this class

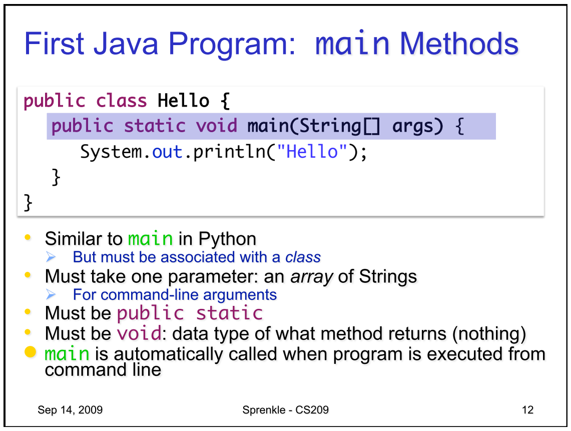


First Java Program

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

method

- Class contains one method: `main`



First Java Program: `main` Methods

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

- Similar to `main` in Python
  - But must be associated with a *class*
- Must take one parameter: an *array* of Strings
  - For command-line arguments
- Must be `public static`
- Must be `void`: data type of what method returns (nothing)
- `main` is automatically called when program is executed from command line

First Java Program

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

- Method contains one line of code
  - What do you think it does?

First Java Program: Print Statements

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

- Calls the `println` method on the `System.out` object
- `println` takes one parameter, a `String`
- Displays string on terminal, terminates the line with new line (\n) character

First Java Program: Comments

```
/**
 * Our first Java class
 * @author Sara Sprenkle
 */
public class Hello {
   public static void main(String[] args) {
      //print a message
      System.out.println("Hello");
   }
}
```

- Comments: `/* */` or `//`

Code Style

```
/**
 * Our first Java class
 * @author Sara Sprenkle
 */
```

- **Comments** at top of program
  - Must include your name
  - High-level description of program
- Proper **indentation**
  - Similar to Python
  - Everything within sets of {} is indented the same

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

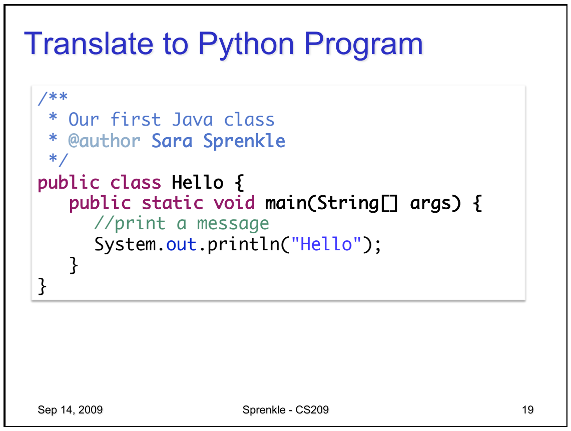
Where are the Differences?

```
# a Python program
def main():
     print "Hello"

main()
```

```
/**
 * Our first Java class
 * @author Sara Sprenkle
 */
public class Hello {
   public static void main(String[] args) {
      //print a message
      System.out.println("Hello");
   }
}
```

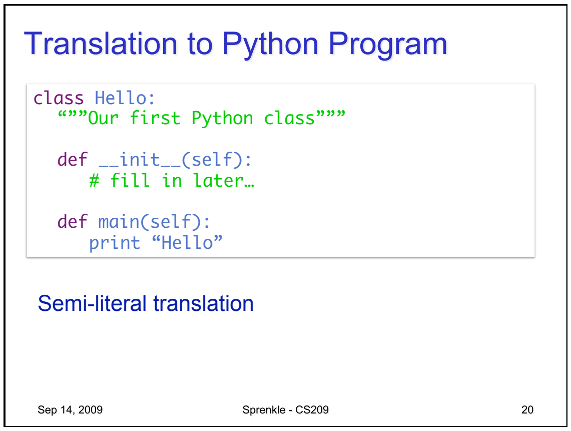
Java vs Python

- ***Semantics*** the same, ***syntax*** different
  - Blocks of code
  - End statements
- Access modifiers
- Data type declarations
- Class-based programs
- Compiled
- We'll see more differences as we go…



Translate to Python Program

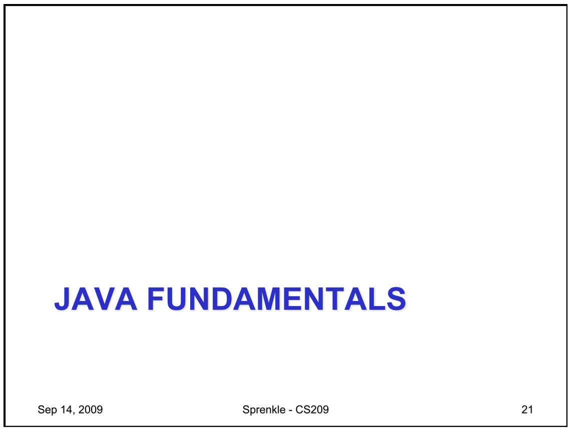
```
/**
 * Our first Java class
 * @author Sara Sprenkle
 */
public class Hello {
   public static void main(String[] args) {
      //print a message
      System.out.println("Hello");
   }
}
```



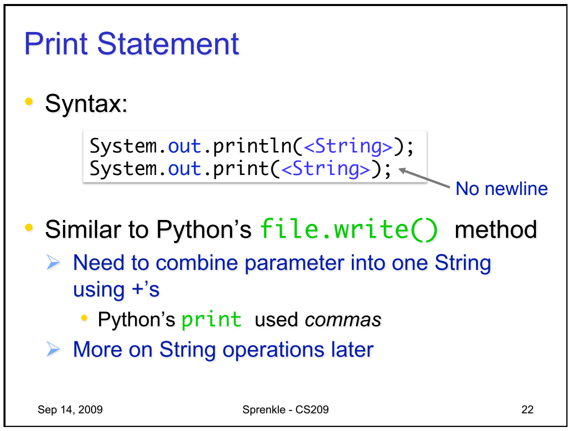
Translation to Python Program

```
class Hello:
   “””Our first Python class”””

   def __init__(self):
       # fill in later…

   def main(self):
       print “Hello”
```

Semi-literal translation



JAVA FUNDAMENTALS



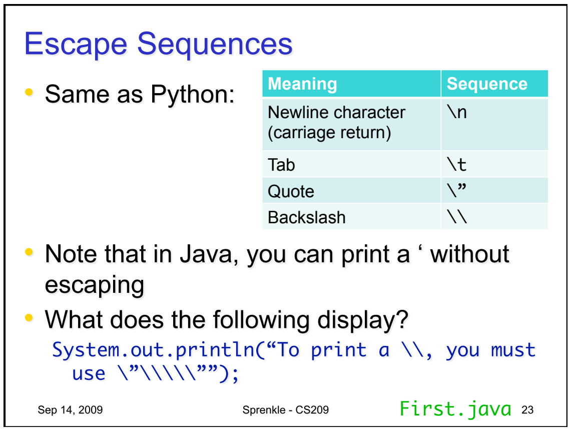
Print Statement

- Syntax:

```
System.out.println(<String>);
System.out.print(<String>);
```

No newline

- Similar to Python’s `file.write()` method
  - Need to combine parameter into one String using +’s
    - Python’s `print` used *commas*
  - More on String operations later



Escape Sequences

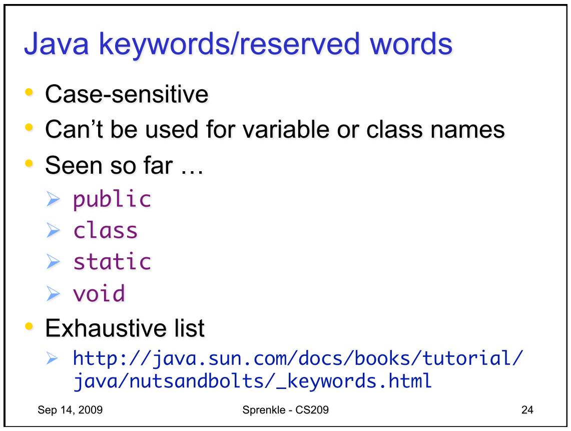
- Same as Python:

Meaning	Sequence
Newline character (carriage return)	\n
Tab	\t
Quote	\”
Backslash	\\

- Note that in Java, you can print a ‘ without escaping
- What does the following display?

```
System.out.println(“To print a \\, you must
  use \”\\\\\””);
```

First.java



Java keywords/reserved words

- Case-sensitive
- Can’t be used for variable or class names
- Seen so far …
  - `public`
  - `class`
  - `static`
  - `void`
- Exhaustive list
  - http://java.sun.com/docs/books/tutorial/java/nutsandbolts/_keywords.html

Data Types

- Java is ***strongly-typed***
    - Every variable must be a **declared type**
- All data in Java is an ***object*** except for the **primitive data types**:

`int`	4 bytes (-2,147,483,648 -> 2,147,483,647)
`short`	2 bytes (-32,768 -> 32,767)
`long`	8 bytes (really big integers)
`byte`	1 byte (-128 -> 127)
`float`	4 bytes (floating point)
`double`	8 bytes (floating point)
`char`	2 bytes (Unicode representation), **single** quotes
`boolean`	`false` or `true`

Variables

- Must be **declared** before used
    - **Syntax**: `<datatype> <name> [= value];`

Optional assignment

- Variable names typically start with lowercase letter
    - '_' also a valid first character
    - Convention: Subsequent words are capitalized
        - Called "Camel Casing"

Variable Examples

- Must be **declared** before used
    - **Syntax**: `<datatype> <name> [= value];`
- Examples:
    - `int x;`
    - `double pi = 3.14;`
    - `char exit = 'q';`
    - `boolean isValid = false;`

Note *must* use single quotes for `chars`

Camel Casing

Floats in Java

- Decimal literals are considered doubles
- This code won't compile:

```
float f = 3.14;
```

- Compiler error message:

```
Float.java:13: possible loss of precision
found    : double
required: float
        float f = 3.14;
```

- To fix code, add an `f` to specification of number or declare as `double`

`Float.java`

Python Transition **Warning**

- You cannot redeclare a variable name in the same scope
- OK:

```
int x = 3;
x = -3;
```

- Not OK:

```
int x = 3;
int x = -3;
boolean x = true;
```

Compiler errors

More Data Type Information

- Default data types
    - Same as Python
    - Result of integer division is an `int`
        - Example: `4/3 = ??`
- Casting
    - Similar to Python for primitive types
    - Example: `4/(double) 3`

Benefits of Static Typing

- Look at `dynamic_typing.py`
- Discussion questions
  - What is the type of `data` at the end of the program?
  - How difficult is this program to understand?
  - If you had to debug this program, how easy/difficult would it be?
  - What is a benefit of dynamic typing?

`alternative_dynamic_typing.py`

Benefits of Static Typing

- Easier to remember type of variable
  - Know what operations that can be executed on a variable
- Compiler can check that you're only using valid operations for this type
- More benefits later this semester

More *Why Java?*

- More **structure** emphasizes/requires better **design**

Constants

- Read-only variables
  - Cannot be assigned new values
- Keyword `final` precedes data type
  - Example within a method:

```
final double CM_PER_INCH = 2.540;
```

Class Constants

- Constant variable for all methods in class or for multiple classes
  - Much more common than constant instance variables
- Requires `static` keyword
  - `static`: "for class"
  - Also used for methods (will see more later)

```
static final double CM_PER_INCH = 2.540;
```

Arithmetic, Relational Operators

- Java has most of the same operators as Python:
  - Arithmetic operators: `+, -, *, /, %`
    - **No power operator:** `**`
  - Relational operators: `==, !=, <, >, <=, >=`
    - Evaluate to a `boolean` value
  - Increment and decrement
    - `+= x, -= y,` etc.
    - Additional shortcut for += 1, -=1: `++ , --`

`Conversion.java`

Output Redirection: >

- In UNIX, we can redirect output to a file
  - For example

```
ls *.java > java_files.out
```

  - Above command saves the output from the `ls` command into the file named `java_files.out`
- This is how you will save output from your Java programs initially
  - For example

```
java Intro > out
```

Programming Assignment 0

- Write a program called `Intro.java`
  - Displays information about yourself
- Fix compiler and logic errors in a program
- Write a simple program using arithmetic

- See Course Web Page
  - Schedule page
  - Bookmark it!