Introducing Java

CSC207 Summer 2018

What does it mean to run a program?

What is a program?

A set of instructions for a computer to follow.

To *run* a program, it must be translated from a high-level *programming language* to a low-level *machine language* whose instructions can be executed.

Roughly, two flavours of translation:

- Interpretation
- Compilation

Java Naming Conventions

- The Java Language Specification recommends these conventions
  - Generally: Use `camelCase` not `pothole_case`.
- **Class name**: A noun phrase starting with a capital.
- **Method() name**: A verb phrase starting with lower case.
- **Instance variable**: A noun phrase starting with lower case.
- **Local variable or parameter**: ditto, but acronyms and abbreviations are more okay.
- **Constant**: all uppercase, pothole_case.
  - E.g.: MAX_ENROLMENT

JavaDoc

- External Documentation (JavaDocs) vs. Internal Documentation (Comments)
- Like a Python docstring, but more structured, and placed above the method, classes, and variables.

```
/**
 * This method takes x and y, does something with it, and
 * returns the sum.
 *
 * @param x  The double to add
 * @param y  The integer to add
 *
 * @return The sum of x and y
 *
 * @throws PiException  If pi is not 22/7 today.
 * @see Integer
 */
public void sums_of_nums(double x, int y) { ... }
```

- starts with /**, not /*
- Classes: **@author**, **@version**
- Methods: **@param**, **@return**, **@returns**, **@see**

- This is where the Java API documentation comes from!
- In IntelliJ: Tools → Generate JavaDoc

Constructors

- A constructor has:
  - the same name as the class
  - no return type (not even void)
- A class can have multiple constructors, as long as their signatures are different.
- If you define no constructors, the compiler supplies one with no parameters and no body.
- If you define any constructor for a class, the compiler will no longer supply the default constructor.

`this`

- `this` is an instance variable that you get without declaring it.
- It's like `self` in Python.
- Its value is the address of the object whose method has been called.

Instance Variables

```
public class Circle {

    private String radius;

}
```

- radius is an instance variable. Each object/instance of the Circle class has its own radius variable.

Defining methods

- A method must have a return type declared. Use void if nothing is returned.
- The form of a return statement:

  `return` *`expression`*`;`

  If the expression is omitted or if the end of the method is reached without executing a return statement, nothing is returned.
- Must specify the accessibility. For now:

  `public` - callable from anywhere

  `private` - callable only from this class
- Variables declared in a method are local to that method.

Parameters

- When passing an argument to a method, you pass what's in the variable's box:
  - For class types, you are passing a reference. (Like in Python.)
  - For primitive types, you are passing a value. (Python can't do anything like this.)
- This has important implications!
- You must be aware of whether you are passing a primitive or object.

Encapsulation

- Think of your class as providing an abstraction, or a service.
    - We provide access to information through a well-defined interface: the public methods of the class.
    - We hide the implementation details.
- What is the advantage of this “encapsulation”?
    - We can change the implementation — to improve speed, reliability, or readability — and no other code has to change.

Conventions

- Make all non-final instance variables either:
  - *private*: accessible only within the class, or
  - *protected*: accessible only within the package.
- When desired, give outside access using “getter” and “setter” methods.
- [A final variable cannot change value; it is a constant.]

Access Modifiers

- Classes can be declared public or package-private.
- Members of classes can be declared public, protected, package-protected, or private.

Modifier	Class	Package	Subclass	World
`public`	Yes	Yes	Yes	Yes
`protected`	Yes	Yes	Yes	No
`default (package private)`	Yes	Yes	No	No
`private`	Yes	No	No	No

What does it mean to run a program?

What is a program?

A set of instructions for a computer to follow.

To *run* a program, it must be translated from a high-level *programming language* to a low-level *machine language* whose instructions can be executed.

Roughly, two flavours of translation:

- Interpretation
- Compilation

Interpreted vs. Compiled

- Interpreted (like Python)
  - Translate and execute one statement at a time
- Compiled (like C)
  - Compile the entire program (once), then execute (any number of times)
- Hybrid (like Java)
  - Compile to something intermediate (in Java, bytecode)
  - The Java Virtual Machine (JVM) runs this intermediate code

Compiling Java

If using command line, you need to do this manually.

First, compile using “javac”:

```
jsin@laptop$ javac HelloWorld.java
```

This produces file “HelloWord.class”:

```
jsin@laptop$ ls

HelloWorld.class  HelloWorld.java
```

Now, run the program using “java”:

```
jsin@laptop$ java HelloWorld

Hello world!
```

Most modern IDEs offer to do this for you (IntelliJ does).

But you should know what’s happening under the hood!