Java

(Recommended Reading: Budd Chapters 1-4)

- Object Oriented Language
- Descends from C++ and Smalltalk (Syntax looks like C++).
- Fixes many C++ features.
- Garbage collected.
- Good for résumés.
- Has safety properties.

Java Programs

1. Source code: `MyClass.java`
2. Compiled to byte-code: `MyClass.class`
3. Virtual Machine
   (a) Interprets byte-codes
   (b) Compiles byte-codes to machine code and runs natively.

Java Environment

javac compiles `.java` to `.class`.

java executes `.class` in runtime.

javadoc generates class documentation from `.java` files.

"Main Program" is class file with *static* function:

```
void main(String[] args);
```

Command line arguments are in array `args`.

A first Program

```
public class HelloWorld extends Object
{
   /**
    * Prints a familiar greeting.
    **/
   public static void main(String[] args)
   {
      System.out.println("Hello World!");
   }
}
```

To run this:

```
fiji% javac HelloWorld.java
fiji% java HelloWorld
Hello World!
```

Class Library

Java comes with an extensive class library.

Many common data structures are already implemented.

The the class reference is online and available in book form.

Date Example

Found class `java.util.Date`. Program to print out date:

```
import java.util.Date;

public class PrintDate
{
   /**
    * print out the date.
    **/
   public static void main(String[] args)
   {
      Date now = new Date();
      System.out.println(now.toString());
   }
}
```


Types

So far we have seen `Date`, `String`, and `OutputStream` objects.

Java also supports *primitive* types:

- boolean – either `true` or `false`.
- char – 16 bit unicode.
- byte – 8 bit signed.
- short – 16 bit signed.
- int – 32 bit signed.
- long – 64 bit signed.
- float – 32 bit signed.
- double – 64 bit signed.

Variables

Java has hybrid object model – not everything is an oject.

A variable value is either

1. A reference to an object (like C++ references) or `null`.

```
Date now = new Date();
System.out.println(now.toString());
```

2. A primitive type

```
int i = 0;
double pi = 3.14159;
```

Command Line Arguments

```
public class ReverseArgs
{
   public static void main(String[] args)
   {
      for (int i=args.length-1; i <= 0; i--)
         System.out.println(args[i]);
   }
}
```


Values

object values are references to data in the heap. Why?

primitive values are just values.

Rooted Class Hierarchy

All classes are descendents of `Object`.

Some inherited methods are:

Object clone() Makes shallow copy. No constructor called.

boolean equals(Object) Shallow comparison of objects.

Class getClass() Gets the runtime class of this object.

String toString() Converts object to a string.

Making Classes

```
public class Point
{
   private int x;
   private int y;

   public Point() { x = 0; y = 0; } // Constructor
   public int getX() { return x; }
   public int getY() { return y; }
}
```

`super` and `this`

Inside a method definition

super is a reference to the parent object.

this is a reference to the object itself.

```
public Point(int x, int y)
{
   this.x = x;
   this.y = y;
}
```

These variables can be used to avoid ambiguity between instance, and parameter variables with the same names.

These can be used in first line of constructors to specify parent constructors (with `super`) or alternative construtor (with `this`).

```
public Point() { this(0,0); }
```


More Point Methods

Define the `toString()` method for `Point` so that

```
(new Point(5,6)).toString()
```

is “(5,6)”

Strings

Strings are a builtin class in Java, `java.lang.String`.

There is special + operator that does concatination of Strings:

```
String myString = "this" + "is" + "my" + "string";

Date now = new Date();
System.out.println("Today is: " + now);

System.out.println("I have " + 5 + "dollars.");
```

Method `int String.length()` returns string's length.

Arrays

Arrays in Java are objects. They are 0-indexed.

```
int halfsize = 10;
int[] array = int[2*halfsize];

for (int i=0; i < array.length; i++)
   array[i] = i;


String[] strings1 = { "hi", "there" };

// or

String[] strings2 = new String[2];
strings2[0] = "hi";
strings2[1] = "there";
```