COE318 Lecture Notes Week 5 (Week of Sept 29, 2014)

Announcements (REPEAT!)

- Midterm (20% of total mark) on Wednesday, October 15. Covers weeks 1–6.

Topics

- Conditionals and loop in Java (and C)
- The enhanced for loop
- Static instance variables and methods
- Notes on Lab 5
- Design and implementation: an “in class” interactive demonstration
- Q&A

Conditionals and loops in Java (and C)

- This is mainly a review from your C course.

```
package basicloopsconditionals;

/**
 * @author Ken Clowes
 */
public class BasicLoopsConditionals {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        //Basic "if"...
        int i = 5;
        int j = 9;
        int max = j;

        //Using just "if" with a default "max"
```

```
        if (i > j) {
            System.out.println("i is bigger than j");
            max = i;
        }
        System.out.println("Max: " + max);

        //Using "if" ... "else", no default
        i = 12;
        j = 3;
        if (i > j) {
            max = i;
        } else {
            max = j;
        }
        System.out.println("Max of " + i + " and "
                + j + ": " + max);

        //Using "?" ternary operator
        i = 5;
        j = 9;
        max = (i > j) ? i : j;
        System.out.println("Max of " + i + " and "
                + j + ": " + max);

        //Using "switch" statements (including "break")
        i = 2;
        switch (i) {
            case 1:
                System.out.println("ONE");
                break;
            case 2:
                System.out.println("TWO");
                break;
            case 3:
                System.out.println("THREE");
                break;
            default:
                System.out.println("unknown");
                break;
        }

        //Unlike C, the Java switch also works with strings
        String s = "two";
        switch (s) {
            case "one":
                System.out.println("un");
```

```
                break;
            case "two":
                System.out.println("deus");
                break;
            default:
                System.out.println("Can't translate " + s);
                break;
        }

        //while loops
        i = 0;
        j = 0;
        int k = 0;
        while (i <= 10) {
            j += i++;
            k++;
        }
        System.out.println("j: " + j + " k: " + k);

        k = 0;
        i = 10;
        j = 2;
        while (--i > j++) {
            k++;
        }
        System.out.println("k: " + k);

        //For loops
        k = 0;
        j = 14;
        for (i = 0; i < j; i++) {
            k++;
            j = j - 2;
        }
        System.out.println("k: " + k);
    }

}
```

- You can download this code here or download the Netbeans project as a zip file.

The enhanced for loop

- Consider:

```
String [] names = {"Alice", "Bob", "Cathy", "Dave"};
//instead of
for(int i = 0; i < names.length; i++) {
    System.out.println(names[i]);
}

//use the "enhanced for loop" (aka the "for each" loop)
for(String name : names) {
    System.out.println(names);
}
```

- Read "`for(String name : names)`" as if it read "for each (String) name in the collection names do"

Static variables and methods

- A method or instance variable that is declared *static* exists as part of the class and no objects of the class need be created to access any static variable or method.
- A static variable is also called a ***class variable***.
- This explains why `main(...)` is a static method. The program must start somewhere and no objects have been created before the program starts! So it must begin execution at a static member of a class; the Java language requires that the name of this method must be "main", that it be public and void and take an array of Strings as its argument.
- All objects (if any) share the static variables. Unlike ordinary (non-static) instance variables which can and usually do have different values in distinct objects, if a static variable is changed by any object, that change is seen by all objects of the class.
- Here is a simple example:

```
public class Foo {

    static int i = 5;
    int j; //ordinary instance variable

    public Foo(int k) {
        j = k;
    }

    public static void f() {
        i++;
    }
```

```
    public static void main(String[] args) {
        System.out.println("Foo.i: " + Foo.i);//prints Foo.i: 5
        Foo.f();
        System.out.println("Foo.i: " + Foo.i);//prints Foo.i: 6
        Foo g = new Foo(2);
        Foo h = new Foo(3);
        h.f();
        System.out.println("g.i: " + g.i);//prints g.i: 7
    }
}
```

- You can download this code here or download the Netbeans project as a zip file.
- NOTE: Although it is legal to access a static variable as belonging to an object (such as "g.i" above), it is standard practice to access it with its class name. Thus "Foo.i" is preferable to "g.i" although both do exactly the same thing.

Joint Project: the "Ball World" (summary)

The following has been decided:

- A class called **Ball**.
  - Characteristics (instance variables) include:
    - **position**
    - **velocity**
    - **radius**
    - **mass**
  - Behaviour (methods)
    - various **getters** and **setters**
    - **getMomentum**
    - **getKineticEnergy**
- Some kind of class (or classes) for **obstacles** (things a Ball can bounce off) such as Walls, Paddles and so forth.
- A class called **BallWorld** that creates the Balls and obstacles and manages their evolution.

Other discussions:

- Issues such as incorporating forces (such as gravity) into the model were raised: to be deferred.

What's next?

What we have now is a “back of the envelop” informal “design”.

What do we do next?

Some possibilities:

- Work on testing code.
- Implement the classes. (Which ones? What order?)
- Complete the design.
- Verify that there is nothing essential missing in our informal design; identify things (if any) that can be deferred until later...

References (text book and online)

- *Head First Java*: Chapter 5
- Flow control: http://docs.oracle.com/javase/tutorial/java/nutsandbolts/flow.html
- Classes and objects: http://docs.oracle.com/javase/tutorial/java/javaOO/index.html