CMPSC 111
Introduction to Computer Science I
Fall 2013
Bob Roos
http://cs.allegheny.edu/~rroos/cs111f2013

Recitation 1
29–30 August 2013
Due at the end of class
"Checkmark" grade

Summary

An important component of learning to program in Java is just becoming familiar with a different way of expressing commands. This non-programming exercise is just for fun, but it's to try to make you more familiar with the "Java way" of writing things.

Example: Instructions for Making a Milkshake (English)

```
Chill the glass.
Allow the ice cream to soften slightly.
Add ice cream, milk, chocolate sauce to the blender.
Run the blender.
Remove the milkshake from the blender and add it to the glass.
```

Identifying the Objects and Actions

The instructions above are in the form of commands (imperative sentences). Each one involves various objects (the "nouns") and actions (the "verbs").

We are going to play around with converting the above instructions into a Java-like notation which I call "Java-ish".

For instance, here are the "objects":

- `glass`
- `iceCream`
- `blender`
- `milk`
- `chocolateSauce`
- `milkshake`

(We think of the final `milkshake` as a separate object constructed from the individual objects `milk`, `iceCream`, etc.)

Here are the "actions" (but in Java-ish we don't call them that; we call them "methods"):

- `chill()`
- `allowToSoften()`
- `add(...)` [two different methods have this name—one for adding objects to the blender, one for adding objects to the glass]
- `run()`
- `remove()` [note that there is an object that is retrieved or "returned" as a result of this method, namely, the milkshake]

In Java-ish, we usually do not capitalize names of objects or methods. That's why we wrote "`milk`" rather than "`Milk`" and "`chill()`" rather than "`Chill()`".

We also don't allow spaces in names of objects and methods; that's why we write "`allowToSoften()`" as one word. However, we *can* use internal capital letters to improve readability—"`allowToSoften()`" is easier to read than "`allowtosoften()`").

Finally, the "methods" are distinguished from the "objects" by including a set of parentheses at the end of the method name.

To "apply" a method to an object, we use the "`.`" notation like this:

object`.`*method(...)*

We also sometimes say that methods are *invoked* by an object.

Some of the methods have empty "`(...)`"s, like "`run()`". Others have things inside the parentheses, supplying additional objects needed to carry out the method's action. For instance, we put the object "`milk`" in the parentheses of "`add(milk)`" to tell us *what* gets added.

Finally, some methods return an object as a result of carrying out the action; we can retrieve the object resulting from a method by "saving it" as follows:

object = *object*`.`*method(...)*

as shown in the example below.

Finally, every command in Java-ish ends with a semicolon "`;`".

Instructions for Making a Milkshake (Java-ish)

```
glass.chill();
icecream.allowToSoften();
blender.add(icecream);
```

```
blender.add(milk);
blender.add(chocolateSauce);
blender.run();
milkshake = blender.remove();
glass.add(milkshake);
```

Getting Money from a Cash Machine

English	Java-ish
Insert debit card into cash machine	`cashMachine.insert(debitCard);`
Type PIN number into cash machine	`cashMachine.type(pinNumber);`
Type amount into cash machine	`cashMachine.type(amount);`
Press the "Enter" button	`cashMachine.pressEnter();`
Remove money ejected by cash machine	`money = cashMachine.remove();`
Eject debit card from cash machine	`debitCard = cashMachine.eject();`

Doing Laundry

English	Java-ish
Put dirty clothes in washer.	`washer.insert(dirtyClothes);`
Put coins in washer.	`washer.insert(coins);`
Run washer.	`washer.run();`
Get wet clothes out of washer.	`wetClothes = washer.empty();`
Put wet clothes in dryer.	`dryer.insert(wetClothes);`
Put coins in dryer.	`dryer.insert(coins);`
Run dryer.	`dryer.run();`
Empty clean clothes from dryer.	`cleanClothes = dryer.empty();`

The Assignment

Think of a simple task and write down imperative instructions for it, in English.

List the objects and methods (actions), choosing names according to the Java-ish rules (no spaces, don't begin with upper case, etc.)

Describe the task in Java-ish. Don't forget the semicolons!

Keep things simple and concrete. (And keep it G-rated!)

Feel free to work with someone else—bounce ideas off one another, try things out—but everyone should do an individual task.

Hand in a written page with your name, today's date, and the above exercise. I'll share the most creative or interesting ones with the class next week!

General Guidelines for Recitation Sessions

- **Experiment!** Recitation sessions are for learning by doing without the pressure of grades

or "right/wrong" answers. So try things! The best way to learn is by trying things out.

- **Submit *something*.** Your grade is just 0 or 1, depending on whether or not you attempt the work and submit something.
- **Try to Finish During Class.** Recitation exercises are not intended to be the equal of laboratory assignments. If you simply can't finish, hand in what you have done and send me a finished version later in the day.
- **Help One Another!** If your neighbor is struggling and you know what to do, offer your help. Don't "do the work" for them, but advise them on what to type or how to handle things.
- **Review the Honor Code policy on the syllabus.** Work you hand in should be your own.