CS 121 – Intro to Programming:Java - Lecture 10

Professor Robert Moll (+ TAs) CS BLDG 276 - 545-4315
moll@cs.umass.edu course web page: http://twiki-edlab.cs.umass.edu/bin/view/Moll121/WebHome
THE WEBSITE LISTS IMPORTANT ANNOUNCEMENTS, AND ALSO ASSIGNMENTS!!!!!!

Announcements

Fifth OWL assignment due next Tuesday;
Third Programming assignment due 10/3 (see website)
Freestyle assignment due Monday / trapdoor tutorial at end of ch3
Midterm posted (W evening 10/15) - old midterm up

OWL acct / edlab acct (PW = oit name, e.g. pbaker, UN = student id)
Principal theme today: methods

Methods organize (sub)jobs at the statement level

They’re the fundamental mechanism for combining elementary operations together to make reusable, more complex operations

You can build an entire “world” with methods: very complex chores rely on complex chores, which rely on elementary chores, which rely on primitives.. etc.
public class SimpleCoins {

    static final int HEADS = 1;
    static final int TAILS  = 0;

    public int flip(){
        if (Math.random() < 0.5)
            return TAILS;
        else
            return HEADS;
    }

    public int multiFlip(int flips) {
        int total = 0;
        for(int j = 0; j < flips; j++) {
            total += flip();
        }
        return total;
    }

}
public int multiFlip(int flips) {
    int total = 0;
    for (int j = 0; j < flips; j++) {
        total += flip();
    }
    return total;
}
public String babyName(){
    String realName = this.getName();
    String who = ("Baby" + realName);
    return who;
}

Why this, in: this.getName();
1) How is babyName used? Answer:

   String someName = jill.babyName();

   This might appear in main somewhere

2) When the babyName method is written, a calling object (here: jill) has not been identified

3) So “this” is a kind of Java pronoun that stands in for the calling object to be named later(!)

4) Happens in English:

   “When she came in, Jill hung up her coat”
Until now: methods are passed, then return primitives, or maybe Strings

We need to study parameter passing more carefully, look at how parameter values can, cannot change.

Let’s look first at methods that return objects
Infant kid = new Infant("Jill",1);
Infant kidTwin = kid.makeTwin("Ivan");

------ a new Infant method ------

public Infant makeTwin(String name){
    int twinAge = this.getAge();
    Infant i = new Infant(name,twinAge);
    return i;
}
public class Infant{
  ..
  public Infant makeTwin(String)
    twinAge = this.age;
  Or
    twinAge = this.getAge();
  ..

  p.s.v main(...){
    myKid = new Infant(“Jill”, 1);
    kidTwin =
      myKid.makeTwin(“fred”);
Recall the cell model of Java variables:

```java
int number = 5;

number = number + 1;
```

![Diagram of variable cell model](image)
Parameter passing in Java

Consider this method:

```java
public void change(int x){x = x + 1;}
```

-- now this code -----

```java
int a = 3;
change(a);
System.out.println(a);
```

What's the value of `a`? (ans: still 3)
Cells and parameter passing: part I

Value of a copied to x. Copy works just in one direction!
An aside on computer memory - RAM

It’s laid out as cells/words, and each has an address

A typical memory cell

1001010010

Sometimes these cells have names: n, or pos, s, etc
The way this works for Java primitives, e.g. ints is straightforward

    int pos = 7;

means:

7

pos
Cells and Objects

Key idea: myKid - an Infant object name - does not hold the myKid object information. Instead, it holds the address of the location where that information is stored.

name: Jill
age: 7
This is the pirate treasure model:

The box on the ship does not hold the treasure. It holds the map that tells the location of the treasure.
Consider this statement:

\[
\text{myKid.anotherMonth();}
\]

Where:

\[
\text{public void anotherMonth()}{
    \text{age = age + 1;}
}\]

The anotherMonth method does not alter the myKid cell; it alters the information at the location referenced by the address in the myKid cell.
Recall: anotherMonth method increments age by 1

```java
myKid.anotherMonth();
```

```
memory address
```

```
101010010110100101
......
```

```
myKid
```

```
class: Infant
name: Ted
age : 3 -> 4
```

```
101010010110100101
```

```
101010010110100101
```

```
memory address
```
The key example:

public class Flight{
    String id;
    String start;
    String end;
    boolean arrived;
    public Flight(String id, String s,
                String e, boolean here)
    {
        this.id = id;
        start = s; end = e; arrived = here;
    }
    ....
    myFlight = new Flight("CE777","JFK","LAX",false);
Suppose that there is also an Airport class, which has a method called

\[
\text{landFlight(Flight f)}
\]

which lands flight f
airportLAX.landFlight(myFlight);

public void landFlight(Flight f){
    f.setArrived(true);
}

We want the **landFlight** method to change an attribute of myFlight - and we can do it, because **myFlight** is literally a reference to data, and we aren’t changing that reference (that address). We’re jumping to that address and changing information there.
Address of myFlight object information
The landFlight parameter does not change - it’s the address of the Flight object information. So our calling principle is not violated. But the referenced object itself does change state: The plane has arrived.
Method Coding and Patterns of Method Development

Loops where a value accumulates (Accumulator)

Looking for some condition to become true (Innocent->Guilty)

Generally: generate and test

------

How to think about nested loops (almost always: use a helper method)

How do you turn a loop that does something into a method that does that thing?
How do I turn this:

```java
int k = 150;
int sum = 0;
for(int j = 1; j < k; j++){
    if(k%j == 0) sum = sum + j;
}
```

into a method??
The “innocent until proven guilty” approach - how can I tell (via a method) if a String includes a vowel (a-e-i-o-u)

innocent = true;

Generate the characters in the string

test for a vowel; if you find one, innocent = false;