CS 121 – Intro to Programming:Java - Lecture 9

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)
Midterm posted (W evening 10/15) - old midterm up
OWL acct / edlab acct (PW = oit name, e.g. pbaker, UN = student id)
Last chance on #2; the drill on #3 +
Some cancellations:

Wednesday discussion in Hasbrouck 138 is cancelled. If you’re in that discussion, go instead (at the same time: 230-320) to either

LGRT 321

LGRT 323

Also: no office hours today 1-2 (but there are office hours beginning at 2, until 3:40)
Program 3 is due Friday. If you haven’t already done so, be sure to do the preliminary project I propose:

Read in a string from the keyboard, then print the characters in the string to the console, in a column (using the string method charAt)
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;
}
Common Errors

```java
public int multiFlip(int flips) {
    int total = 0;
    for(int j = 0; j < flips; j++) {
        total += flip();
    }
    System.out.println( total);
}

public int multiFlip(int flips) {
    int total = 0;
    for(int j = 0; j < flips; j++) {
        total += flip();
    }
    return "total";
}
```
Suppose we're interested in testing if two consecutive flips are the same. How would we add this to the class?

Return type?

Parameters?

Need a return statement?
public boolean doubleFlip(){
    // 2 flips: are they the same?
    int flip1= flip(); int flip2 = flip();
    if (flip1 == flip2)
        return true;
    else
        return false;
}

public boolean doubleFlip(){
    int flip1= flip(); int flip2 = flip();
    return (flip1 == flip2);
}

public boolean doubleFlip(){
    return (flip() == flip());
}
Suppose you flip n (some number of coins) 100,000 times, and you want to know how often (with what frequency - a decimal fractional amount) exactly k heads comes up [for example: I toss 20 coins, with what frequency, over 100000 tosses, do I get 3 heads??]

What’s the return type?

What are the parameters?

Need a return statement?
// hCt = headCount

public double patFreq(int coins, int hCt){
}

public double patFreq(int coins, int hCt) {
    double freq = 0.0;

    return freq;
}
public double patFreq(int coins, int hCt){
    double freq = 0.0;
    int ctr = 0;
    for(int j = 0; j < 100000; j++){
        if (multiFlip(coins) == hCt) ctr++;
    }
    freq = (double)ctr / 100000;
    return freq;
}
How about this method, in Infant class:

It’s called babyName. It returns an Infant’s name, with the word “Baby” pasted on the front.

Return type?

Parameter(s)?

Return statement?
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");
}
Let’s do a method called divSum: you give it a (positive) integer, it returns the number of integers that divide the number evenly.

Example: you give it 10, it should return 3 (since 1, 2, 5 are the divisors of 10)

Write that method:

1) Header line

2) Return statement

3) Body
class SimplePt{

    private int x;  private int y;

    public SimplePt(int xx, int yy){
        x = xx;  y = yy;
    }

    public int getX(){
        return x;
    }

    public int getY(){
        return y;
    }

    public double dist(SimplePt other){
        double deltaX = (this.getX() - other.getX());
        double deltaY = (this.getY() - other.getY());
        return Math.sqrt(deltaX*deltaX + deltaY*deltaY);
    }
}
SimplePt p = new SimplePt(12, 15);
SimplePt q = new SimplePt(9, -4);

System.out.println(p.dist(q));
SimplePt p

(x,y)

(x,0)

Projection of p onto x axis
public SimplePt xProject(){
    SimplePt q = new SimplePt(this.getX(), 0);
    return q;
}

---- (main, somewhere else..) ----

SimplePt p = new SimplePt(12, 15);
SimplePt r = p.xProject();
Recall the cell model of Java variables:

```java
int number = 5;
```

number

```
5
```

```
6
```

```java
number = number + 1;
```

number

```
5
```
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!
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.

| 1101..1010 | name: Jill |
| 1101..1010 | age: 7 |
Consider this statement:

    myKid.anotherMonth();

Where:

    public void anotherMonth(){
        age = age + 1;
    }

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

myKid.anotherMonth();
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);
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.