Announcements

Ch 5 OWL assignment due Thursday, 5 PM

Programming assignment 4 due 10/19

Grading complete on prog 1, 2

Midterm posted- W evening 10/14. 7 - 8:15 PM, Thompson 102, 104. Old midterm at CourseAdmin link

TA Office hours:

  M 1-4;  TU 3:45-5;  W 4-6 (4-5:15 until 10/10);  TH 2 - 5;  F 12:15 - 3:15;  -  LGRT 223
OPENING FOR THE SEMESTER - SUNDAY, SEPTEMBER 27

Peer Tutoring
Supplemental Instruction

OPEN: Sunday 5:00 – 10:00 P.M.
Monday – Thursday 1:00 – 10:00 P.M.

10th floor – W. E. B. Du Bois Library
Students helping students achieve academic success.
www.umass.edu/lrc
545-5334
Learning Resource Center Tutoring Schedule

CMPSCI 121

SUN 5:00pm-10:00pm

MON 1:00pm-10:00pm

TU 6:30pm-10:00pm

WED 4:00pm-10:00pm

TH 6:30pm-10:00pm
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;
}
Where does method input come from?

• parameters

• object attributes

• more global sources - Math.PI
What can methods do?

• Produce output - via return statements
• Change object state - alter instance variables

  myKid.anotherMonth();

• Write stuff on the wall (ahem: the screen)

  System.out.println("Hi Julie");
String someName = jill.babyName();

The calling object calls the method - it's written before any object created; created after class
is written requires Infant obj name
to form its return String
public String babyName()
{
    String realName = this.name;
    String who = ("Baby" + realName);
}

return who;
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 showing the cell model of Java variables, with an initial value of 5, an assignment to `number = number + 1`, and the result of 6.]
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!

Actual parameter  formal parameter

change
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:

pos

7
Cells and Objects

Key idea: myKid - an Infant object name - does not hold myKid object information.

It holds the address of the location where that information is stored.
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.

The data in the myKid cell is the map to the myKid information
Consider this statement:

    myKid.anotherMonth();

Where:

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

    The anotherMonth method does **not** alter cell labelled myKid

It alters the information at location referenced by the address in the myKid cell.
anotherMonth method increments age by 1
myKid.anotherMonth();
Another example:

There’s an Airport class, a flight class

airportLAX.landFlight(myFlight);
The key example:

```java
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

```java
public void landFlight(Flight f){...}
```

which lands flight f
```
airportLAX.landFlight(myFlight);

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

landFlight method changes attribute of myFlight

myFlight holds a reference to data -- we are not changing that reference (that address).

We’re jumping to that address and changing information there.
```
Address of myFlight object information

myFlight

method parameter

landFlight method

101110

101110

myFlight

method parameter
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.
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??

• What does the fragment do?
• Where does info come from?
• What is the effect of the code (output?)
How can I tell (via a method) if a String includes a vowel (a-e-i-o-u)

ans = false;

Generate the characters in the string
test for vowel; if find one, ans = true;

Example cries out for a helper method
Formatted printing: printf

double bucks = 129.95;
System.out.println(bucks) - a loser
System.out.printf("$%6.2f", bucks);
Prints $129.95

double bigBucks = 12345.67;
System.out.printf("$%,8.2f", bigBucks); $12,345.67
System.out.printf("$%,8.2f\n", bigBucks);
$12,345.67 + newline
Writing header lines / return smts

• Count digits in a string
• Any perfect squares between a and b?
• At 6%, how much will $750 earn after 11 years?
• Change Vince’s (an infant’s) name to Cicero.
Algorithms for computing running averages

Alg$_1$: when $a_n$ comes in, calc $(a_1 + ... + a_n)/n$, print answer

How many operations with 10,000 $a$'s?
50,000,000 (!)
Alg$_2$: when $a_n$ comes in,

$\text{sum} = \text{sum} + a_n$

divisor++

print (sum/divisor)

How many operations with 10,000 $a$'s?

30,000 (!)