Announcements

Ch 5 Embedded problems due Thursday

Programming Assignment 4 up today, or tomorrow

Ch 5 OWL assignment due 10/13

Online midterm coming end of next week

Next week: Tuesday is Monday schedule (so discussions are on Tuesday, no Tuesday lecture)

Exam discussed at end of class today
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.
Java.util - includes Scanner Random, etc.

Infant class
Car class, etc.

myCar.unusedCap();

Looping; Conditional statements;

Packages

Classes

Methods

Statements
public class Car{
    private String make; // manufacturer
    private double fuelCapacity;
    private double fuelAmount;

    public Car(String what, double cap, double amt){
        make = what;
        fuelCapacity = cap;
        fuelAmount = amt;
    }

    public double fillUpCost(double gallonCost){
        return ((fuelCapacity - fuelAmount) * gallonCost);
    }

    public double getFuel(){return fuelAmount;}

    public void setFuel(double amt){fuelAmount = amt;}
}
public double getFuel(){return fuelAmount;}

Characteristics

• An accessor method

• Does not alter calling object

• Has a return statement / type matches with method return type.

• No parameters - all information comes from calling object
public double getFuel(){return fuelAmount;}

both double

Characteristics

• An accessor method
• Does not alter calling object
• Has a return statement / type matches with method return type.
• No parameters - all information comes from calling object
public void setFuel(double amt){fuelAmount = amt;}

Characteristics

• A mutator method

• **Does** alter calling object

• Has no return statement / return type is void

• One parameter - all information comes from parameter
public void setFuel(double amt){fuelAmount = amt;}

Characteristics

• A mutator method

• Does alter calling object

• Has no return statement / return type is void

• One parameter - all information comes from parameter
public double fillUpCost(double gallonCost) {
    return ((fuelCapacity - fuelAmount) * gallonCost);
}

**Characteristics**

- Not a mutator method
- Does not alter calling object
- Has a return statement / return type is double
- One parameter - information comes both from calling object and from parameter parameter
public double fillUpCost(double gallonCost) {
    return ((fuelCapacity - fuelAmount) * gallonCost);
}

**Characteristics**  
- return type double
- Not a mutator method
- Does not alter calling object
- Has a return statement / return type is double
- One parameter - information comes both from calling object and from parameter
The Math class and static methods

• Not all methods are invoked by objects.
• The Math class, for example
• Doesn’t make sense to attach a function like sqrt or max to an object
• All Math fns are static, Math.sqrt() etc.

• Character.isLetter(…) is also a static call - Character is a class (not the same as char)
myKid.anotherMonth();

jacksCar.setFuel(12.0);

vs

double r = Math.sqrt(2);

boolean what = Character.isDigit('a');
double r = Math.max(3.5,7.1);

double r = Math.sqrt(2.0);

double r = Math.sin(.7);

double r = Math.min(3.5,7.1);

double r = Math.pow(2,5);  // 2^5 returned as double
An unusual function in the Math class:

\[
\text{Math.random()}
\]

Gives a random value \( r, 0 \leq r < 1 \)

What does random mean? Basically: generating many in a row will reveal no pattern...

Repeated calls are uniformly distributed between 0 and 1.
Write code that considers some large number of random numbers and checks to see if the average of those numbers is 1/2 (0.5).
sum = 0.0;

for(int j = 0; j < 1000000; j++)
    sum = sum + Math.random();

> sum/1000000
0.5002762821348795
public class SimpleCoins {

    static final int HEADS = 1; // a constant!
    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 = total + flip();
        }
        return total;
    }
}

public int multiFlip(int flips) {
    int total = 0;
    for (int j = 0; j < flips; j++) {
        total += flip();
    }
    return total;
}
public void greeting(String name){
    System.out.println("hello " + name);
}

void used for: object state changes; announcements

(If you’re sitting at a slot machine, you pull the lever, and a message appears: “You’ve won $100” - that’s void;

    when the 400 quarters come pouring out, a return stmt is required, and your return type is int or double)
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 = 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);
}
```

应该是void，不应当返回任何东西。

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

不应该返回一个int类型，返回的是一个String类型，且不应当返回total的值。
Where does method data 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");
  ```
Suppose we want to test if 2 consecutive flips are the same. How would we add -doubleFlip- to SimpleCoins?

Return type?

Parameters?

Need a return statement?
public boolean doubleFlip(){ // 2 flips: 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 $n$ 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 flip 20 coins, with what frequency do I get 3 heads (over 100000 tosses - possible ans: 0.0087)??]

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;
}
Method **divCount**: 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 divCount

1) Header line

2) Return statement

3) Body
public int divCount(int k) {
    int ct = 0;
    for (int j = 1; j <= k/2; j++)
        if (k % j == 0) ct++;
    return ct;
}
The Art of method development

Write a method that reports the number of vowels in a string...

So--- int count = vCount(s);

public vCount(String s){
    int ct = 0;
    for(int j = 0; j < s.length(); s++)
        if (isVowel(s.charAt(j))) ct++;
    return ct;  }
It’s the breakup of the method into two - a main method and a helper - which is the essence here.

public boolean isVowel(Char c) {
    return (c == 'a') || (c == 'e') || (c == 'i') || (c == 'o') || (c == 'u');
Caesar Cipher
Used by Caesar himself to communicate with his generals.

A simple linear shift cipher - nowadays - useless.

So if the shift amount is 2

ZOO -> BQQ

Shift of 5? 26? 31?
import java.util.*;

public class CipherTest{
    public static void main(String[] args){
        Scanner sca = new Scanner(System.in);
        System.out.println("enter a phrase");
        String s = sca.nextLine();
        System.out.println("enter a shift amount");
        int k = sca.nextInt();
        Cipher c = new Cipher(k);
        System.out.println(s);
        System.out.println(c.encrypt(s));
    }
}

public class Cipher{
    private int shift;

    public Cipher(int s){
        shift = s;
    }

    public String encrypt(String plain){
        plain = plain.toUpperCase();
        String result = "";
        for(int j = 0; j < plain.length(); j++){
            result = result + letterShift(plain.charAt(j));
        }
        return result ;
    }
}
private char letterShift(char c) {
    // expects c to be either a non-letter or upper case
    if (Character.isLetter(c)) {
        int cPos = (c - 'A'); // a position from 0 to 25
        cPos = (cPos + shift) % 26; // deal with overflow
        return ((char) ('A' + cPos)); // remake as caps
    } else return c;
}
Exam

Probably released Thur night - you have until Sat to do it.

You have exactly 75 minutes to do it - you can't start, stop, restart.

Open book.

Of course: do it alone!

Timing - the exam, and drop date.

Covers ch 1-5. Sample exam at CourseAdministration on website. Also: review disc sheets, OWL problems, embedded problems, and so forth