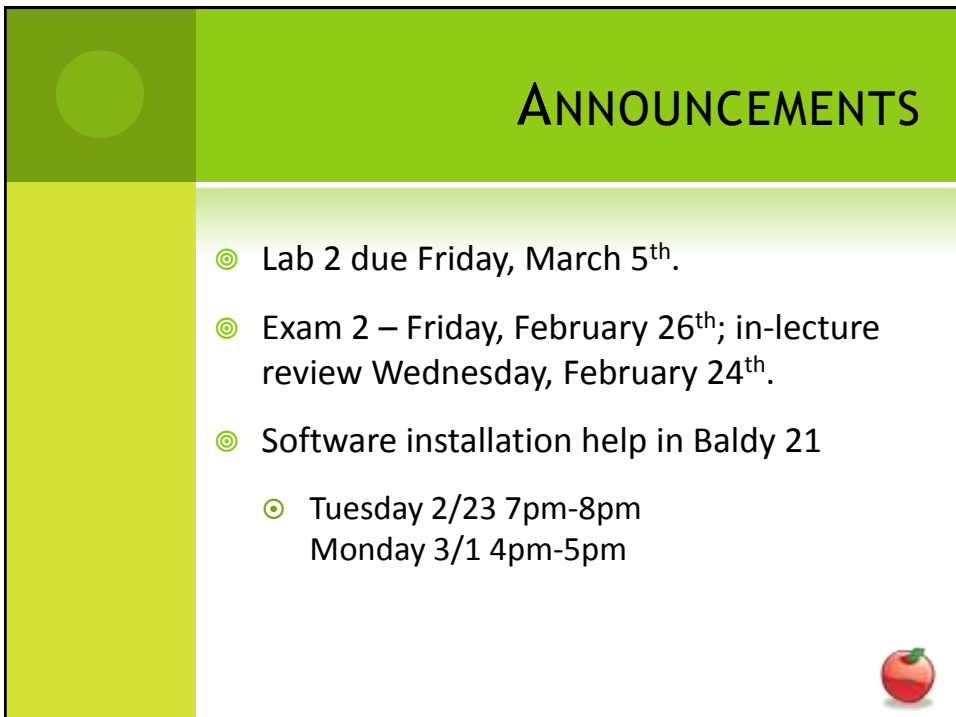



CSE 113 A

February 22-26, 2010



ANNOUNCEMENTS

- ⦿ Lab 2 due Friday, March 5th.
- ⦿ Exam 2 – Friday, February 26th; in-lecture review Wednesday, February 24th.
- ⦿ Software installation help in Baldy 21
 - ⦿ Tuesday 2/23 7pm-8pm
 - ⦿ Monday 3/1 4pm-5pm



3

EXERCISE (FROM FRIDAY'S LECTURE)

- ⊙ Make the ambulance add 5 flowers to the screen when the ambulance is at $y = 36$.

- 1- Open Ambulance class in editor
- 2- Begin to define a new method in Ambulance class
- 3- Make sure there is a call to the new method in `act()`



4

EXERCISE

- ⊙ When inside the method, we brainstormed what types of actions the method would need to do.
- ⊙ We decided that we needed an if-statement
 - ⊙ We did not need an else – why?
 - ⊙ Because there was no “other” alternative. If a condition was met, we performed an action. If it was not, we simply did nothing. If there was another action to perform, then we would have needed an else.
- ⊙ Then, we decided what our condition for the if should look like (if our y-coordinate was 36)



5

EXERCISE

- ⦿ Then we decided what should happen if the condition was met (add 5 flowers to the world)
- ⦿ We then began to code both the condition and the insertion of the flowers into the world.
- ⦿ Creating 5 flowers at a time was as simple as copying and pasting the code to create one flower, five times. To create 50, paste 50 times, 500, paste 500 times. But there must be a way we can get the computer to do all this repetition...



6

LOOPS

- ⦿ Mechanism that allows our programs to repeat
- ⦿ There are many different types of loops, we focused today on one called the for-loop
- ⦿ The for-loop is good for counting (doing things a specific number of times).



7

FOR-LOOP (BASIC SYNTAX)

```
for ( /* initialization, condition, increment */ )  
{  
    //code to be repeated  
}
```



8

FOR-LOOP (INITIALIZATION)

- ⦿ The initialization part of a for-loop (typically) creates and initializes a loop counter. The loop counter is simply a variable whose value we can check (in the condition part) and change (in the increment part).
- ⦿ A sample initialization step looks like this:

```
int count = 0;
```
- ⦿ You can name your variable whatever you'd like and initialize it to whatever value is appropriate to your task.



9

FOR-LOOP (CONDITION)

- ⦿ The condition part of a for-loop tells the loop when to stop. While the condition is true, the code will keep getting repeated, when it is false, the repetition will stop. Typically, the condition involves the loop counter variable's value.
- ⦿ A sample condition looks like this:

```
count < 100;
```
- ⦿ The condition must evaluate to true or false (must be a boolean expression) and typically gives a clue to the number of times the loop will execute.



10

FOR-LOOP (INCREMENT)

- ⦿ In increment part of a for-loop indicates how the loop counter will be incremented. In many cases, the counter is incremented by one each time (so it keeps count of how many times the loop has executed).
- ⦿ A sample increment step looks like this:

```
count = count + 1
```
- ⦿ Note that there is a (syntactic) shortcut for the above that is typically used:

```
count++
```



11

FOR-LOOP (REPEATED CODE)

- ⦿ This is the code that is to be repeated. There is no restriction on what is placed in the {}. Any valid Java code can be repeated.



12

FOR-LOOPS (A NOTE ABOUT THESE GUIDELINES)

- ⦿ The usage of the for-loop described here corresponds with its most common usage. However, the only thing that the Java language specifies is that within the parentheses there be three statements separated by commas and that the statements in between the {} are the code that is repeated.
- ⦿ So, that means

```
for(;;)
{
}
```
- ⦿ Is a perfectly syntactically correct for-loop. That loop will run forever (it will never stop), but perform no actions. This is not pointed out to panic anyone, but rather to indicate that there may be other uses of the for loop that you can see that do not conform to the usage we've discussed here.

