


# CSE 113 B

March 28 – April 1, 2011

## ANNOUNCEMENTS - GRADES

- ⦿ UBLearns grades just updated Monday, March 28<sup>th</sup> (11:00am). Please check grades.
- ⦿ If an EXAM grade is incorrect, you need to bring the exam to me – preferably in office hours.
- ⦿ If you have a question about the grading of a lab, you should come to talk to me so that we can look at your submission and the grading.
- ⦿ If a grade is incorrectly recorded on UBLearns from Web-CAT, an email is fine. 

## ANNOUNCEMENTS - LAB

- ⊙ Lab 7 started in lab this week.
- ⊙ Labs 6 & 7 will be graded by Web-CAT, but the grading is not functional at this time.
- ⊙ Practice Assignment 6 has been posted and grading is functional.
- ⊙ Practice Assignment 7 will be posted at some point soon.



## ANNOUNCEMENTS - PRACTICAL EXAM 3

- ⊙ May 5, 6, 9, 10
- ⊙ Schedule will be posted after resign date (Friday, April 1<sup>st</sup>) on the Practical Exam 3 information page (which will be linked off of the Schedule page).
- ⊙ Information about what material will be on the exam is also posted there.



## ANNOUNCEMENTS - EXAMS

- ⊙ Pick up Exam 1 & 2 if you have not already done so.
- ⊙ Exam 3 Monday, April 11<sup>th</sup> in lecture (covers Chapters 6-8).
- ⊙ Review for Exam 3 on Friday, April 8<sup>th</sup>.
- ⊙ Review sheet will be posted on or around April 1<sup>st</sup>.



## GREENFOOTIMAGE

- ⊙ Class that represents an image.
- ⊙ You can load an image that has been pre-drawn (like we've been doing).
- ⊙ OR
- ⊙ You can draw an image yourself.



7

## GREENFOOTIMAGE

- ⊙ setColor method allows us to set the current drawing color
- ⊙ setColor takes as a parameter, a `java.awt.Color`



8

## JAVA.AWT.COLOR

- ⊙ There are several pre-defined colors in Java that you can use
- ⊙ `java.awt.Color.PINK`, `java.awt.Color.RED`,  
`java.awt.Color.ORANGE`, `java.awt.Color.YELLOW`,  
`java.awt.Color.GREEN`, `java.awt.Color.CYAN`,  
`java.awt.Color.BLUE`, `java.awt.Color.MAGENTA`,  
`java.awt.Color.LIGHT_GRAY`, `java.awt.Color.GRAY`,  
`java.awt.Color.DARK_GRAY`, `java.awt.Color.BLACK`,  
`java.awt.Color.WHITE`



9

## JAVA.AWT.COLOR

- ⦿ You can also create a color using  
new java.awt.Color(red, green, blue)
- ⦿ where you substitute a number within the range 0-255 for each of red, green, and blue



10

## GREENFOOTIMAGE

- ⦿ Once we have set the color for drawing, we can do many things (see list of methods from GreenfootImage)
  - ⦿ fill
  - ⦿ fill/draw oval
  - ⦿ fill/draw rectangle
  - ⦿ draw line
  - ⦿ draw text



11

## FOR-LOOP (BASIC SYNTAX)

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



12

## 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.



13

## 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.



14

## 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++
```



15

## 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.



16

## 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.

