Decisions
Lecture 6 (7/11/2007)

Useless Fact of the Day

• The last FIFA World Cup was watched by 1.1 billion people (1/6 the world population), making it the most widely-viewed sporting event in the world (surpassing even the Olympic Games).

Topics

• Some more misc. useful stuff to know
• The if statement
• AND and OR
• Comparison and Equality

Commenting

• Very useful!
• So both you and others can look at your code and understand what you mean it to do very quickly
• Two ways:

```c
// a commented line
/* a comment which spans
 * more than one line
 * and is also a poem
 * because of this rhyme */
```
A Mini-Lecture on Scope

- The **scope** of a variable defines what parts of the code can “see” it (use it), and what parts can’t.
- A variable can be seen anywhere within the squiggly braces (body section) where it is declared, after its declaration.

```
public class Bounce extends GameLoop
{
    // I declare ball here:
    Sprite ball;
    public void startGame ()
    {
        // I can use ball anywhere in the class!
        ball = new EllipseSprite(1, 1);
        // I can NOT use x here:
        x = 4;
        // I declare x here:
        int x = 3;
        // I CAN use x here:
        x = 12;
    }
    public void advanceFrame (double timePassed)
    {
        // I can’t use x here:
        x = 14;
    }
    ...
}
```

The **if** Statement

```
if (A)
{
    // we do this stuff if statement A was true.
}
else if (B)
{
    // we do this stuff if A was false but B was true.
}
else
{
    // we do this stuff if both A and B were false.
}
```

Boolean Expression

- Any expression that evaluates to a boolean
- Examples:
  - `5 == 7` evaluates to `false`
  - `'a' == 'a'` evaluates to `true`
  - `6 > 4` evaluates to `true`
  - `4 <= 4` evaluates to `true` (= “less than or equal to”)
  - `ball.isVisible()` evaluates to `true` if the ball’s visible, `false` if it’s not
  - `true == false` evaluates to `false`
AND and OR

- We can combine multiple boolean expressions into one
  - AND: &&
  - The statement \( A \land B \) is true if and only if the individual statements \( A \) and \( B \) are both true
  - OR: ||
  - The statement \( A \lor B \) is true if and only if at least one of the individual statements \( A \) and \( B \) is true

Example:
- \( 5 == 7 \land 7 == 7 \) → false
- \( 5 == 7 \lor 7 == 7 \) → true
- \( (5 == 7 \land 7 == 7) \lor 4 == 4 \) → true

Equality

- When using the == operator is awesome:
  - comparing integers (Ex: \( 13 == 13 \) → true)
  - comparing characters (chars) (Ex: \( 'a' == 'b' \) → false)
  - comparing booleans (Ex: ball.isVisible() == star.isVisible())
    - (the “isVisible()” method returns a boolean)
- When using the == operator can be very bad:
  - comparing doubles
    - Ex.: \( (\text{Math.sqrt}(13) \times \text{Math.sqrt}(13)) == 13 \) → false
  - comparing Strings or other objects
    - Ex.: String one = “lool”;
      String two = “lo”;
      two = two + “l”;
      two == one → false

Comparing doubles

- Why shouldn’t you use == with doubles?
  - Round-off error makes things imprecise
  - \( \text{Math.sqrt}(13) \times \text{Math.sqrt}(13) \rightarrow 12.999999999999998 \)
  - 12.999999999999998 does not equal 13
  - Instead of ==, we must compare within a tolerance
    - We assume doubles, \( x \) and \( y \), to be equal if they are within some small tolerance of each other
    - Instead of using \( x == y \), use something like one of these two things:
      - \( x - y < 0.0000005 \land x - y > -0.0000005 \)
      - \( \text{Math.abs}(x - y) < 0.0000005 \)

Comparing Strings

- Why can’t you use == with Strings (or other objects)?
  - == checks to see if two things are the same object
  - == does not check the contents of the objects to see if they are the same
  - Two Strings, then, can store the same data (“hello”, for example), but since they are different objects, == returns false when comparing them
  - Use the \textbf{String.equals(String)} method (an instance method)
    - Instead of asking string1 == string2, use:
      - string1.equals(string2)
Comparing Strings

- Other useful String methods:
  - startsWith(String prefix)
  - equalsIgnoreCase(String s)
  - compareTo(String s)
    - compares two strings lexicographically (Ex.: “cat” > “cab”)
    - result is a negative number if the string is less than s, positive if it is greater, and zero if they are equal
  - toLowerCase() and toUpperCase()
  - substring(int beginIndex) and substring(int beginIndex, int endIndex)
    - returns a new substring which is a substring of this string

Game-Relevant Decisions

- Number comparison
  - Is the number of bad guys left zero? Is the number of points greater than 1000? Is the current level greater than 3?
- Intersection
  - Is the bullet intersecting the space ship?
    - Often simplified to: Is the point within the rectangle?
  - Are the two space ships intersecting?
    - Often simplified to: Are the two rectangles/circles intersecting?