TOPIC 8

MORE ON WHILE LOOPS

Notes adapted from Introduction to Computing and Programming with Java: A Multimedia Approach by M. Guzdial and B. Ericson, and instructor materials prepared by B. Ericson.

Outline

- How to use logical operators
- How to use while loops in general
- How to do keyboard input

While Loops

- Recall that the basic syntax for a ***while loop*** is:

```
while (test)
{
    body of loop
 }
```

where
- test is a condition that is true or false
- body of the loop consists of the statements to be executed while the condition is true

While loops

- We will now look at while loops that do not just involve counting
- Our tests will be more general than just checking the end value for a counter, as in a previous example:

```
int total = 0;
int counter = 1;
while (counter <= 100)
{
total = total + counter;
counter = counter + 1;
}
...
```

Recall: Relational Operators

- **Relational operators**
  - > >= < <= == !=
  - Compare two operands of the same type
- **Relational expressions**
  - Express a condition that evaluates to true or false
  - Example: (counter <= 100)
- Sometimes we want to test for several conditions being true
  - We can combine relational expressions using **logical operators**

Logical Operators

- **and operator &&**
  - Used to check if several things are true
  - Example: (x > 0) && (x <100)
    - Expression evaluates to true if both (x >0) and (x < 100)
  - **Short circuiting** : evaluation stops if the first condition turns out to be false
    - Why does Java do this?

Logical Operators

- **or operator ||**
  - Used to check if at least one of several things is true
  - Example: (x > 0) || (x <100)
    - Expression evaluates to true if either (x >0) or (x < 100)
  - Java does short circuiting for || also

Logical Operators

- **Exclusive or operator ^**
  - Used to check if one and only one of the things is true
  - Example: (x < 0) ^ (y <0)
    - Expression evaluates to true if either (x < 0) or (y < 0) but not both
  - Can this be short-circuited?

Logical Operators

- **not *operator*** **!**
- Used to change the value of a condition to its opposite

  !true is false

  !false is true
- Example: !(x == y)
  - Expression evaluates to true if (x == y) is false, evaluates to false if (x ==y) is true

Truth Tables for && and || in Java

A	B	A&&B
true	true	true
true	false	false
false	*any*	false

A	B	A||B
true	*any*	true
false	true	true
false	false	false

Using && (and) and || (or)

- Check that a value is in a range
  - Example: check for valid pixel color values
    - Is some value between 0 and 255 inclusive?
    - 0 <= x <= 255 is written in Java as

      0 <= x && x <= 255

      or

      x >= 0 && x <= 255
- Check if at least one of several things is true
  - Example: Is the color black or white?

Keyboard Input

- The textbook provides a class called SimpleInput
  - In the bookClasses folder
- It contains methods that make it easy for use to input data from the keyboard:
  - getNumber for getting doubles
  - getIntNumber for getting integers
  - getString for getting strings
- They are class methods (not object methods)

Simple Input Methods

- public static double getNumber(String message)
  - The parameter message is the prompt message to display to the user
  - The input window will keep appearing until a valid number type is input
  - The number typed by the user is returned as a double

SimpleInput Methods

- public static int getIntNumber(String message)
  - The number typed by the user is returned as an int
- public static String getString(String message)
  - The data typed by the user is returned as a String

While Loop Example

- Suppose we want a user to enter a number between 0 and 255 inclusive. If the number entered is not in the correct range, the user should be asked again to enter the number:

```
int number = -1;
while ( number < 0 || number > 255)
{
  number = SimpleInput.getIntNumber("Enter a
       number between 0 and 255 inclusive: ");
}
```

While Loop Example

- What will happen if the user enters
  the number 300?
  the number -2?
- Why is the variable number initialized to -1 and not to 0?

Keyboard and while

- Write a program that takes an image and changes the red value of all pixels to some specific value
- Exercise: write a method makeRed for the Picture class with the header
  `public void makeRed(int redValue)`
- Sample main program:

```
public class TestRed{
  public static void main(String args[ ]) {
     int redValue = 100;
     Picture p =
               new Picture(FileChooser.pickAFile());
     p.explore();
     p.makeRed(redValue);
     p.explore();
  }
}
```

Continued

- Now what if we wanted to see what the picture looks like with the red values set to 150? Or 200?
  - We would have to change the statement `int redValue = 100;` to `int redValue = 150;` etc.
  - We would have to *recompile* the program before we could run it again
- A more flexible way: ask the user to enter the new red value on the keyboard
  - And also check that the input is valid!

Continued

```
public class TestRed {
    public static void main(String args[]) {
      int redValue =  -1;
      Picture p = new Picture(FileChooser.pickAFile());
      p.explore();
      while ( redValue < 0 || redValue > 255)
      {
        redValue = SimpleInput.getIntNumber("Enter a
                 number between 0 and 255 inclusive: ");
      }
      p.makeRed(redValue);
      p.explore();
   }
}
```

Summary

- Relational Operators
- Logical Operators
- Simple Input
- From the Keyboard