Announcements

- Lab 1 was due at 9:45

- Program 2 Assigned

- Midterm?
  - Tentatively scheduled next Friday or early the following week
    - Date will be posted sometime this weekend
    - Survey posted to website to help determine timing

- Topics include all that we’ve covered through loops + one more new topic

- Program 2’s Due Date and the Midterm will not be on the same day
Questions?

Yesterday:

- More Branching Statements
  - Short-Circuit Evaluation
  - ? : Operator
  - switch Statements
  - System.exit(0)

- Introduction to while Loops
Today in COMP 110

- Lecture
  - More Loops

- Lab
  - Lab 3: Smiley
More Loops

- Loop Types
- Scope
- Advanced Loop Control
Loop Types

- While Loops
- Do...While Loops
- For Loops
- For Each Loops
While Loop

- Syntax
  - while (Boolean_Expression)
    Body_Statement

- Execution
  - If the Boolean_Expression evaluates to true, the Body_Statement is executed
  - As long as Boolean_Expression remains true, the Body_Statement is executed again
  - If the Boolean_Expression evaluates to false, flow of control continues to the next statement
While Loop Example

Code

System.out.println("Enter a series of positive integers and I will add them.");
System.out.println("Enter a negative number to indicate the end of the series.");

Scanner keyboard = new Scanner(System.in);

int currentValue = 0;
int sum = 0;

while (currentValue >= 0) {
    sum += currentValue;
    currentValue = keyboard.nextInt();
}

System.out.println("The sum of the positive values is " + sum);

Sample Execution

Enter a series of positive integers and I will add them. Enter a negative number to indicate the end of the series.
2
6
2
5
7
2
-1
The sum is 24
Sentinel Value

- A special value used to break a loop

- The value cannot match normal values used within the loop, for example
  - Use a negative number to end a loop over positive numbers
  - Use zero to end a loop of non-zero numbers
  - Use a word to end a loop of parsed (converted from string) numbers
While Loop Example

Code

```java
System.out.println("Enter a series of positive integers and I will add them.");
System.out.println("Enter a negative number to indicate the end of the series.");

Scanner keyboard = new Scanner(System.in);

int currentValue = 0;
int sum = 0;

while (currentValue >= 0)
{
    sum += currentValue;
    currentValue = keyboard.nextInt();
}

System.out.println("The sum of the positive values is "+sum);
```

Sample Execution

Enter a series of positive integers and I will add them. Enter a negative number to indicate the end of the series.

2
6
2
5
7
2
-1

The sum is 24
While v. Do...While Loops

**While Loop**

- Start
- Controlling Expression
  - True
    - Loop Body Statements
    - False
      - True
        - Loop Body Statements
        - False
          - End
          - Start

**Do...While Loops**

- Start
- Controlling Expression
  - True
    - Loop Body Statements
  - False
    - End

While v. Do...While Loops

**While Loop**
- **Syntax**
  - `while (Boolean_Expression)
    Body_Statement`
- **Body_Statement** may never be executed

**Do...While Loops**
- **Syntax**
  - `do
    Body_Statement
    while (Boolean_Expression);`
- **Body_Statement** will execute at least once
- do...while statement always ends with ;
Do...While Loops: The ;

- Syntax

  - do
    
    Body_Statement
    while (Boolean_Expression);

  - do
    
    Body_Statement_1
    Body_Statement_2
    // ...
    Body_Statement_n
    } while (Boolean_Expression);

- Examples

  - int i = 1;
    do
      i++;
    while (i <= 10);

  - int i = 1;
    do
      {
        System.out.println(i);
        i++;        
      } while (i <= 10);

- Recommend using block syntax, even for single statement loops
For Loops

- A “syntactic sugar” for a common form of while loop
  - Provides no new features, but is convenient to use

Syntax

```java
for (Init_Statement; Boolean_Expression; Update_Statement)
    Body_Statement
```

```java
for (Init_Statement; Boolean_Expression; Update_Statement)
{
    Body_Statement_1
    Body_Statement_2
    // ...
    Body_Statement_n
}
```
For Loop as While Loop

- For Loop
  - for (Init_Statement; Boolean_Expression; Update_Statement)
  - {
    Body_Statement_1
    Body_Statement_2
    // ...
    Body_Statement_n
  }

- While Loop
  - Init_Statement
  - while (Boolean_Expression)
  - {
    Body_Statement_1
    Body_Statement_2
    // ...
    Body_Statement_n
    Update_Statement
  }

- Like the while loop, if the Boolean_Expression starts out false, the body will not be executed
For Loop as While Loop

- These three loops are equivalent (except for the scope of i)
  - For Loop
    - `int i;
      for (i = 1; i <= 10; i++)
      {
        System.out.println(i);
      }
  
    - `for (int i = 1; i <= 10; i++)
      {
        System.out.println(i);
      }
  
  - While Loop
    - `int i = 1;
      while (i <= 10)
      {
        System.out.println(i);
        i++;
      }`
For Loop Variable Declaration

- Can declare the loop variable inside the `Init_Statement` of the `for` statement
  ```java
  for (int i = 1; i <= 10; i++)
  {
      System.out.println(i);
  }
  ```
For Loop Extended Syntax

- for (Init_Statement; Boolean_Expression; Update_Statement) Body_Statement

- Where Init_Statement is
  - Variable_Declaration
    - Declare & assign one or more variables of the same type
  - Statement
    - Perform any other one statement
    - No blocks, ifs, loops, or declarations
  - Statement_1, Statement_2, ...
    - Executes multiple statements in sequence
    - Often used to assign multiple variables with different types
    - No blocks, ifs, loops, or declarations
    - Comma-delimited, no semicolons

- Where Update_Statement is
  - Statement
    - Executes just one statement
    - Often used for a single assignment statement
    - No blocks, ifs, loops, or declarations
  - Statement_1, Statement_2
    - Executes multiple statements in sequence
    - Often used for multiple assignment statements
    - No blocks, ifs, loops, or declarations
    - Comma-delimited, no semicolons

- Perform no initializing action
- Perform no update action
Loops and Strings

Example

String myString = "Hello World!";
for (int i = 0; i < myString.length(); i++)
{
    System.out.println("Char at " + i + ". is " + myString.charAt(i));
}

Note that the loop starts at zero and uses the strictly less than comparison
Nested Loops Examples

- Shaking Hands (including with self, and duplicates)
  ```java
  for (int i = 0; i < studentCount; i++)
  {
      for (int j = 0; j < studentCount; j++)
      {
          System.out.println("Student " + i + ", shook Student " + j + ",\'s hand");
      }
  }
  ```

- Shaking Hands (only others, no duplicates)
  ```java
  for (int i = 0; i < studentCount; i++)
  {
      for (int j = i + 1; j < studentCount; j++)
      {
          System.out.println("Student " + i + ", shook Student " + j + ",\'s hand");
      }
  }
  ```
For Each Loops

- A syntactic sugar for specialized for loops that iterate over a set of items

Syntax

- \[ \text{for (Type variable : Set)} \]
  
  \[ \text{Body Statement} \]

- \[ \text{for (Type variable : Set)} \]
  
  \[ \{ \text{Body Statement}_1 \text{Body Statement}_2 \text{Body Statement}_n \} \]

- The possibilities for \text{Set} will be covered in a later class
Loop Use Case Recommendations

- **While Loop**
  - General looping, but the body need not be executed at least once
  - Purposeful infinite (or near-infinite) loops

- **Do...While Loop**
  - General looping, but the body must be executed at least once

- **For Loop**
  - Looping over a known iteration count

- **For Each Loop**
  - Looping over a set of values
Method Variable Scope

- Variable Scope: the parts of a program in which a variable can be accessed

- A method variable declared within a block
  - Is accessible anywhere in that block after it is declared
  - Is accessible in any child-block located after it is declared
  - Is NOT accessible outside of the block in which it is declared
  - Is NOT the same variable as an identically named variable of a sibling-block
public static void main(String[] args) {
    int myVar1 = 2;
    if (myVar1 >= 2) {
        int myVar3 = 5;
        System.out.println(myVar1);
        System.out.println(myVar3);
    }
    System.out.println(myVar1);
    System.out.println(myVar3); // ERROR
}
public static void main(String[] args)
{
    int myVar1;
    for (myVar1 = 0; myVar1 < 10; myVar1++)
    {
        int myVar3 = 5;
        System.out.println(myVar1);
        System.out.println(myVar3);
    }
    System.out.println(myVar1);
    System.out.println(myVar3); // ERROR
}
Method Variable Scope Example

```java
public static void main(String[] args) {
    for (int myVar1 = 0; myVar1 < 10; myVar1++) {
        int myVar3 = 5;
        System.out.println(myVar1);
        System.out.println(myVar3);
    }
    System.out.println(myVar1); // ERROR
    System.out.println(myVar3); // ERROR
}
```
public static void main(String[] args) {
    for (int i = 0; i < 10; i++) {
        int myInt = 5;
        System.out.println(i);
        System.out.println(myInt);
    }

    for (int i = 0; i < 100; i++) {
        System.out.println(i);
        System.out.println(myInt); // ERROR
    }
}
public static void main(String[] args) {
    for (int i = 0; i < 10; i++) {
        int myInt = 5;
        System.out.println(i);
        System.out.println(myInt);
    }

    for (int i = 0; i < 100; i++) {
        int myInt = 5; // Is different myInt, no error
        System.out.println(i);
        System.out.println(myInt);
    }
}
public static void main(String[] args) {
    for (int i = 0; i < 10; i++) {
        int myInt = 5;
        System.out.println(i);
        System.out.println(myInt);
    }

    for (int i = 0; i < 100; i++) {
        int myInt = 7;  // Is different myInt, no error
        System.out.println(i);
        System.out.println(myInt);
    }
}
Boolean Loop Control

- The controlling expression can be a boolean variable

Scanner keyboard = new Scanner(System.in);

int currentValue = 0;
int sum = 0;

while (currentValue >= 0)
{
    sum += currentValue;
    currentValue = keyboard.nextInt();
}

System.out.println(sum);

Scanner keyboard = new Scanner(System.in);

int sum = 0;
boolean areMore = true;

while (areMore)
{
    int currentValue = keyboard.nextInt();
    if (currentValue >= 0)
    {
        sum += currentValue;
    }
    else
    {
        areMore = false;
    }
}
The controlling expression can be a boolean variable

```java
Scanner keyboard = new Scanner(System.in);
int currentValue = 0;
int sum = 0;

while (currentValue >= 0) {
    sum += currentValue;
    currentValue = keyboard.nextInt();
}

System.out.println(sum);
```

```java
Scanner keyboard = new Scanner(System.in);
int sum = 0;
boolean areMore = true;

do {
    int currentValue = keyboard.nextInt();
    if (currentValue >= 0) {
        sum += currentValue;
        areMore = true;
    } else
        areMore = false;
} while (areMore)
```
Break Statement

- Returns control to the statements that follow the current loop
  - Aborts the loop

Syntax:

- break;
while (itemNumber <= MAX_ITEMS) {
    // ...
    if (itemCost <= leftToSpend) {
        leftToSpend -= itemCost;
        itemNumber++;
        if (leftToSpend <= 0) {
            // Out of money
            break;
        }
    }
}
Continue Statement

- Advances to the next iteration of the current loop
  - Skips the rest of the statements in this iteration
  - Still performs the `Update_Statement` in for loops
  - Still tests the controlling `Boolean_Expression` in all loops

- Syntax:
  - `continue;`
Continue Statement Example

- Shaking Hands (not self, but duplicates)
  - for (int i = 0; i < studentCount; i++)
    {
      for (int j = 0; j < studentCount; j++)
      {
        if (i == j)
          continue;

        System.out.println("Student " + i + " shook Student " + j + ")");
      }
    }
The break & continue statements are not required, but are conveniences
- It is possible to rewrite a loop to eliminate these statements

Shaking Hands (not self, but duplicates)
- for (int i = 0; i < studentCount; i++)
  {
    for (int j = 0; j < studentCount; j++)
    {
      if (i != j)
      {
        System.out.println("Student " + i + 
                          " shook Student " + j + "," + "'s hand");
      }
    }
  }
Advanced Break/Continue

- Syntax
  - Declaring the Loop
    - `labelName: Loop_Statement`
  - Within the Loop
    - `break labelName;`
    - `continue labelName;`

- Performs the break or continue action on the labeled loop instead of the current loop

- `labelName` uses the same naming style as variable names
Advanced Break/Continue Example

String myStr1 = "weight";
String myStr2 = "effect";
char match = '\0'; // The NULL character

OuterLoop: for (int i1 = 0; i1 < myStr1.length(); i1++)
{
    for (int i2 = 0; i2 < myStr2.length(); i2++)
    {
        if (myStr1.charAt(i1) == myStr2.charAt(i2))
        {
            match = myStr1.charAt(i1);
            break OuterLoop;
        }
    }
}

if (match != '\0')
{
    System.out.println(match);
}
Advanced Break/Continue Example

String myStr1 = "weight";
String myStr2 = "effect";

OuterLoop: for (int i1 = 0; i1 < myStr1.length(); i1++)
{
    for (int i2 = 0; i2 < myStr2.length(); i2++)
    {
        if (myStr1.charAt(i1) == myStr2.charAt(i2))
        {
            System.out.println(myStr1.charAt(i1));
            continue OuterLoop;
        }
    }
}
Questions?
Logistics

- **Next:**
  - Lab 3: Smiley

- **Tomorrow:**
  - Even More Loops
  - Debugging Techniques

- **Assignments**
  - Program 1 is Due Tomorrow
  - Lab 2 is Due Tomorrow
  - Lab 3 is Due on Tuesday
  - Program 2 is Available