Repetition Statements
Recitation – 02/20/2009

CS 180
Department of Computer Science,
Purdue University
Announcement

- Grades are out:
  - Project 2, 3
  - Lab 2, 3, 4, 5

- Project 4 is due on Feb 25th
Syntax for Three Forms of Loops

- **while loop**

```
while ( <boolean expression> )
  <statement block>
```

- **do-while loop**

```
do
  <statement>
while ( <boolean expression> )
```

- **for loop**

```
for ( <initialization>; <boolean expression>; <increment> )
  <statement>
```
Control Flow of **while**

```java
while (number <= 100) {
    sum    = sum + number;
    number = number + 1;
}
```

- **previous statement;**
- **int sum = 0, number = 1**
- **number <= 100 ?**
  - **true**
    - `sum = sum + number;`
    - `number = number + 1;`
  - **false**
    - next statement;
Control Flow of **do-while**

```java
int sum = 0, number = 1

// Statement (loop body)
do {
    sum += number;
    number++;
} while (number <= 100);
```

**Boolean Expression**

- `true` and `false` are used for the do-while loop.

**Diagram:**

- **previous statement;**
- `int sum = 0, number = 1` (start of loop)
- `sum += number; number++;`
- `number <= 100 ?` (Boolean expression)
- `true` branch:
  - `sum += number; number++;`
- `false` branch:
  - `next statement;`
Control flow of for

Initialization: number=1, sum=0

Boolean Expression: number<=100

Increment: number++

Statement (loop body): sum += number;

previou statement;

number=1, sum=0;

number<=100?

false

next statement;

true

sum += number;

number++;

next statement;
Choosing a Loop

- If you know how many times the loop will be iterated, use a **for loop**.
- If you don’t know how many times the loop will be iterated, but
  - it could be zero, use a **while loop**
  - it will be at least once, use a **do-while loop**.
Example 1

- The `charAt( int )` method of the `String` class returns the character at the given index of an instance.

```java
public String someMeth1( String str )
{
    String result = "";
    for ( int index = str.length() - 1; index >= 0; --index )
    {
        result += str.charAt( index );
    }
    return result;
}
```

What is the result of:

```java
someMeth1( "somewhere i have never traveled" )
```
Example 2

```java
public void someMeth2()
{
    String name = "";
    Scanner keyboard = new Scanner( System.in );
    do
    {
        System.out.print( "Please enter a name or "quit" to exit: " );
        name = keyboard.next();
        System.out.println( "Howdy "+name+"!" );
    } while ( !name.equals( "quit" ) );
}
```

How does the above compare to using a `while` loop? Is a `for` loop appropriate?
Example 3

```java
public void someMeth3()
{
    String name = "";
    Scanner keyboard = new Scanner( System.in );
    while ( true )
    {
        System.out.print( 
            "Please enter a name or "quit" to exit: " );
        name = keyboard.next();
        if ( !name.equals( "quit" ) )
            System.out.println( "Howdy "+name+"!" );
        else
            break;
    }
}
```

Does the above loop terminate?
How does this compare to `someMeth2`?
Example 4

```java
public void someMeth4() {
    String name = "";
    Scanner keyboard = new Scanner( System.in );
    S.o.p( "Please enter a name or "quit" to exit: " );
    name = keyboard.next();
    while ( !name.equals("quit") ) {
        S.o.pln( "Howdy "+name+"!" );
        S.o.p( "Please enter a name or "quit" to exit: " );
        name = keyboard.next();
    }
}
```

Notice that this has a cleaner flow than example 3. Priming the loop can avoid extra cases and bugs.
Example 5

```java
public void someMeth5()
{
    Scanner keyboard = new Scanner( System.in );
    while ( true )
    {
        switch( keyboard.nextInt() )
        {
        case 0: case 1: case 2:
            System.out.println( "In here" );
            break;
        case 3: case 4: case 5:
            System.out.println( "Somewhere else" );
            continue;
        }
        System.out.println( "After Switch" );
    }
}
```

How does this loop behave when 7 is entered? 3? 0?
Example 6

```java
public void someMethod6()
{
    for ( int i = 0; i < 5; ++i )
    {
        for ( int j = 5-i; j > 0; --j )
            System.out.print( "*" );

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

Notice how loops within loops can interact. What does each loop correspond to?

Notice how using a table can help you execute this manually.
Example 7

```java
public void someMeth7()
{
    for ( int i = 0; i < 5; ++i )
    {
        for ( int j = 5-i; j > 0; --i )
        {
            System.out.print( "*" );
            System.out.println();
        }
    }
}
```

What happens here?
Where is the mistake?
Example 8

```java
public void someMethod8()
{
    for ( int i = 0; i < 5; ++i )
    {
        for ( int j = 5-i; j > 0; --j )
        {
            System.out.print( "*" );
            --i;
            System.out.println();
        }
    }
}
```

How does this behave?
Example 9

```java
public void someMeth9()
{
    int sum = 0;
    for (int count=1; count<5; count++)
        System.out.println("Count:" + count);
    sum = sum + count;
    System.out.println("Sum: " + sum);
}
```

How does this behave?
What have we forgotten?
public void someMeth10()
{
    int sum = 0;
    for (int count=1; count<5; count++)
    {
        System.out.println("Count:" + count);
        sum = sum + count;
    }
    System.out.println("Sum: " + sum);
}
Quiz

- What is the difference between `break` and `continue`?
- Are the different forms of loops equivalent?