

## Methods

- o A way to change the recipe-reading flow of execution through a program
  - o It's like having to stop at a certain point in a pie recipe because you have to make up whipped cream
  - o The method is like the recipe for doing whipped cream that the pie recipe needs
  - o When you're done with the whipped cream, you can go back to the pie recipe where you left off

IT Dept.

## Methods Can Call Methods

- o Example:

```
System.out.println( Math.max( 5, Math.min( Math.abs(-1), 6 ) ) );
```

IT Dept.

## Flow of Execution

- o Methods are just one way of changing the flow of execution in a program
- o You will see other ways that:
  - o Jump to another point in a method/program due to an error (Exceptions)
  - o Cycle through the same set of statements multiple times (Loops)

IT Dept.

## Parameters

- o Here is the first line of the method declaration:

```
public boolean equalsIgnoreCase(String anotherString)
```

- o Here is the way the method is called:

```
String str1 = "luigi", str2 = "LUIGI";  
str1.equalsIgnoreCase( str2 );
```

- o How does the string get into the method???

IT Dept.

## Pass-by-value

- o Basic types are passed into a method by their value
  - o This means that the method receives a *copy* of the variable.
- o Can the method change the value of the original variable?
  - o No! It can only change the copy.

IT Dept.

## Pass-by-value for references

- o Reference variables are passed by their reference or address
  - o This means that the method gets a copy of the address to the object
- o Can the address change within the method?
  - o No! Only the copy can change.

IT Dept.

## Example

```
public class testPassing {
    static int i = 0;
    static StringBuffer str = new StringBuffer("princess");

    public static void main( String[] args ) {
        System.out.println("Int: " + i + " String: " + str);
        testVars( i, str);
        System.out.println("Int: " + i + " String: " + str);
    }

    public static void testVars( int i, StringBuffer str ) {
        i = 5;
        str.append("mario");
    }
}
```

IT Dept.

## When Do You?

- o When can you use variables/methods and where?
  - o This is called "scope"

IT Dept.

## Scope Rules

- o Scope is the term used to describe the *visibility* of a method or variable.
- o Examples
  - o Private methods have class scope (they may only be used within the class they are created in)
  - o A private instance variable also has class scope and can be used only within the class it's declared in

IT Dept.

## Local Variable Scope

- o The scope of a local variable declared inside a method is the method.
  - o The variable does not exist outside of the method.

IT Dept.

## Shadowing

- o What will this code print?

```
class Test
{
    private int mold = 1;
    public Test() {
        int mold = 0;
        System.out.print("mold=" + mold);
        System.out.println(", Test.mold=" + this.mold);
    }

    public static void main(String[] args) {
        Test myTest = new Test();
    }
}
```

IT Dept.

## Forward Instance Variable Declaration

- o Bad code:

```
class Test
{
    public static void main(String[] args) {
        System.out.print("mold=" + mold);
        System.out.println(", Test.mold=" + Test.mold);
        int mold = 0;
    }
    static int mold = 1;
}
```

IT Dept.

## If Statement

### o Syntax:

```
if (booleanExpression) statement
```

or

```
if (booleanExpression)
    statement
else
    statement
```

IT Dept.

## Example

```
import java.util.*;

public class Morning {
    public static void main( String args[] ) {

        Calendar rightNow= Calendar.getInstance();

        if (rightNow.get( Calendar.AM_PM )==Calendar.AM )
            System.out.println( "Good morning..." );
        else
            System.out.println( "Good afternoon..." );
    }
}
```

IT Dept.

## It doesn't matter...

### o Since

- o semicolons mark the end of an expression
- o Curly brackets mark a compound statement

- o It doesn't matter to the compiler if you put parts of a statement on different lines!

IT Dept.

## ...but it does

- o It matters very much to humans how your code is formatted

IT Dept.

## Dangling Else

```
if ( booleanExpression )
    if ( booleanExpression )
        statement
    else
        statement
```

- o The `else` clause is always associated with the nearest `if`
- o Use `{ ... }` to change the association

IT Dept.

## There are two main kinds of conditional statements

- o if
- o switch
  - o Some people recommend you only use if for style reasons
- o Also a ternary operator: we will go over this next week

IT Dept.

## The switch statement

- o Syntax (The default case is optional):

```
switch ( expression ) {  
    case char/byte/short/int constant : statementSequence  
  
    default: statementSequence  
}
```

IT Dept.

## Example

```
int zzz=5;  
int i=2;  
switch ( i ) {  
    case 1: zzz = 1;  
           break;  
    case 2:  
    case 3: zzz = 3;  
           break;  
    default: zzz = 0;  
}
```

IT Dept.

## Breaking out

- o The `break` statement can occur anywhere within a `switch`, `for`, `while` or `do` statement and causes execution to break out of the current statement (which may be a loop) and jump to the next statement.

IT Dept.

## Self Quiz

- o Express the following switch statement using nested if statements:

```
int grade = 60;  
int a=0,b=0;  
switch (grade) {  
    case 60: a = 1;  
    case 70: a = 2;  
           b = 2;  
           break;  
    case 80: a = 3;  
           b = 4;  
           break;  
    default: a = 5;  
           break;  
}
```

## Ternary Operator

- o `boolean-expr ? value1 : value2;`

```
if ( characterType == MOOCOW)  
    defaultText = "Moooooo.";  
else defaultText = "Hello.";
```

```
defaultText = (characterType == MOOCOW) ? "Moooooo." : "Hello.";
```

IT Dept.

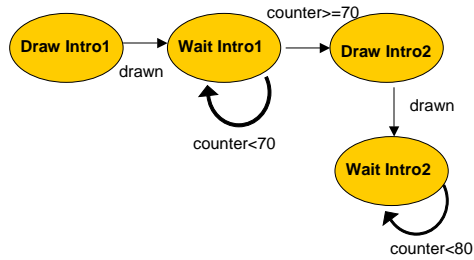
## State Machines

- o "An abstract machine consisting of a set of states (including the initial state), a set of input events, a set of output events, and a state transition function. The function takes the current state and an input event and returns the new set of output events and the next state. Some states may be designated as "terminal states". The state machine can also be viewed as a function which maps an ordered sequence of input events into a corresponding sequence of (sets of) output events."

– dictionary.com

IT Dept.

## Visualization of States



IT Dept.

## IceBlox State Machine

```

switch (gameState) {
  case 0: prepareField(); break;
  case 1: showField(); break;
  case 2: gameLoop(); break;
  case 3: happyPenguin(); break;
  case 4: clearField(); break;
  case 5: fixDeath(); break;
  case 6: gameOver(); break;
  case 7: drawIntro1(); break;
  case 8: waitIntro1(); break;
  case 9: drawIntro2(); break;
  case 10: waitIntro2(); break;
  case 11: drawIntro3(); break;
  case 12: waitIntro3(); break;
  default: break;
}
  
```

## Method for Waiting

```

public void waitIntro2() {
  offGraphics.setColor(Color.black);
  offGraphics.drawImage(
    small[1+(counter % 12)],
    140,110,this);
  offGraphics.fillRect(140,270,30,30);
  offGraphics.drawImage(small[animF[counter&7]],
    140,270,this);
  if (counter>80)
    gameState=11;
}
  
```

IT Dept.

## State Machines

- o Many games are just fancy state machines
  - o Many games use hierarchical state machines: state machines inside of state machines
  - o Some game-specific scripting languages support special syntax for state machines
    - o Unreal Tournament UnrealScript Example: <http://www.cs.rit.edu/~cs1/lectureNotes/BotConnection.uc>
    - o Note the similarity in most of the scripting language constructs to the Java language with the exception of the default properties and states at the bottom of the file.

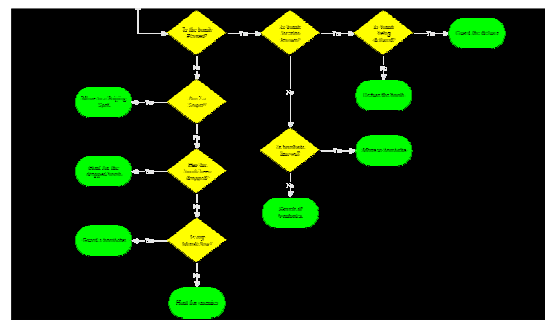
IT Dept.

## Other Ways of Using Conditionals?

- o Games can use conditionals in order to make decisions in the way characters behave
  - o Example on the next slide:
    - o CounterStrike: Condition Zero
    - o From a GDC talk on the "Official Counterstrike Bot" given by Michael Booth

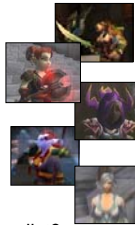
IT Dept.

## Deciding What to Do: Scenario Objectives (Simplified Counter-Terrorist Bomb Example)



## What's This?

- |         |            |
|---------|------------|
| 1. 3422 | 1. Hunter  |
| 2. 8945 | 2. Warrior |
| 3. 5643 | 3. Warlock |
| 4. 3527 | 4. Mage    |
| 5. 3472 | 5. Priest  |



A list of health values?      A character class list?

IT Dept.

## Arrays

- o A variable can hold exactly one value
- o Arrays do lists of items
- o Each location in the array is identified by its position/index in the array
  - o Array indices start at 0 and run to one less than the number of locations in the array

IT Dept.

## 1 Dimensional (1-D) Arrays



IT Dept.

## Arrays

- o What else would it be helpful to make into arrays?
- o Why are the indices helpful?

IT Dept.

## Arrays

- o Arrays are *objects* but there is no class that array objects are instances of
  - o This means that array variables are reference types
- o Like any reference type, before using an array you must do two things
  1. Declare the identifier (name) that will be used to refer to the array
  2. Create the array (give it space to put values in)
  3. Values inside an array must be of the same type

IT Dept.

## Arrays

- o Variables of array type are *declared* using bracket ([ ]) notation:

```
typename[] identifier;  
typename identifier[];
```

IT Dept.

## Array Creation

- o It is possible to directly initialize/create the values of the array elements using an initializer list:

```
int[] n = { 1, 2, 3, 4, 5 };  
float[] m = {1.3f, 2.4f, 3.5f};  
String w[] = {"Tom", "Scary", "Moo Cow"};
```

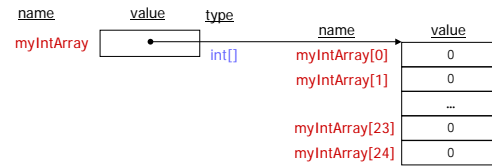
- o Or the array can be created with the new command:

```
int[] n = new int[5];  
float[] m = new float[5];  
String w[] = new String[3];
```

IT Dept.

## Creating an Array

```
int[] myIntArray = new int[ 25 ];
```



IT Dept.