

## IT218

- o Course web page:  
[http://games.rit.edu/intro\\_prog/games\\_prog\\_2/](http://games.rit.edu/intro_prog/games_prog_2/)
- o My web page: <http://www.it.rit.edu/~jdb/it2>
- o Email: jdbics [at] it [dot] rit [dot] edu
- o TA: Brian Murphy

## Goals

- o Software setup: use MSAA
- o Policies: course structure, grading, etc.
- o Java to C#
- o Visual Studio

## Software Setup

- o If you are able on your home machine:
  - o Please install Visual Studio .NET 2005 on your home machine
  - o You should also download the DirectX libraries (SDK)
- o Remember to go to the ET lab when you need help.

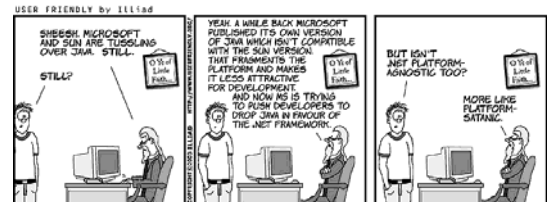
## Books and the Schedule Page

- o See web site:  
[http://games.rit.edu/intro\\_prog/games\\_prog\\_2/](http://games.rit.edu/intro_prog/games_prog_2/)
- o Note that the GD&D sections are quite a bit different than the regular IT218 sections.

## Projects

- o Projects show that you can develop a larger program outside of class than can be done in a lab
  - o Quarter-based project: Developing an Unreal Tournament Gamebots client bot from scratch
  - o The project will be done in "chunks" with the end result being deathmatch bots that will compete against each other at the end of the quarter

## Java vs. C#: a religious war?

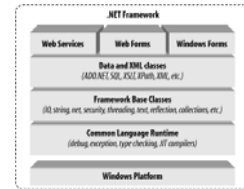


## Why?

- o Learning multiple languages is good for you
  - o Java and C# are not very different
  - o You will use many languages over the course of your career
- o It looks good on your resume
  - o The MS Explorer's program

## The .NET Framework

- o Common Language Infrastructure (CLI)
  - o An international standard
  - o Like the Java Byte Code
- o Common Language Runtime (CLR)
  - o Like the Java Virtual Machine
- o Framework Class Library (FCL)

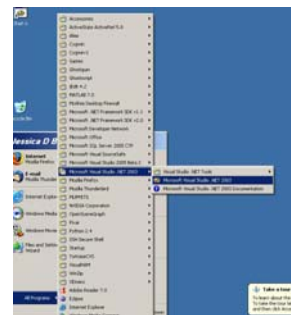


from Programming C#

## Integrated Development Environments

- o They are harder to learn as they are more complicated, but they increase your overall production in the long run
- o We will be using Visual Studio .NET 2005

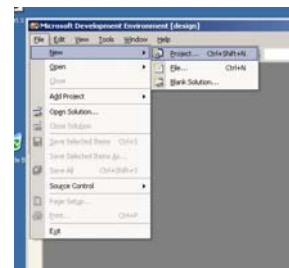
## Starting Visual Studio



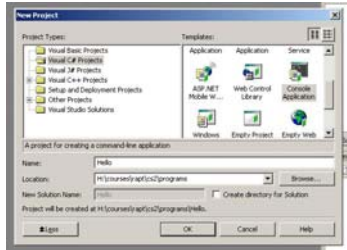
## Notes on Style

- o With Visual Studio .NET 2005 you pretty much end up having to use their style for indentation and brackets unless you change their templates
- o We want you to use RIT's style as you will have to use it for other courses
- o Changing brackets: Tools->Options->Text Editor-> C#

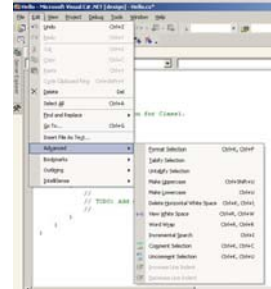
## Creating a New Project



## Creating a C# Program



## Helpful Commands



## Java

```
public class GoLevel {  
    public static void main( String[] args ) {  
        System.out.println("Go level n00b!");  
    }  
}
```

## C#

```
using System;  
public class GoLevel {  
    public static void Main(string[] args) {  
        System.Console.WriteLine("Go level n00b!");  
    }  
}
```

## Running a Program

- o Within Visual Studio
  - o Set the project to be the main build project
  - o Build
  - o Debug -> Start
- o At the command line prompt:
  - o csc GoLevel.cs
  - o GoLevel

## Comments

- o You can use XML (eXtended Markup Language) tags as comments
- o In order to build web page docs, we will use a separate tool (described in a later course)

## Namespaces

- o Useful for resolving possible conflicts between programs or components with large projects
- o Sometimes a company web address is reversed for the project namespace:
  - o edu.rit.cs for domain cs.rit.edu
- o You must include System if you want access to simple I/O methods
  - o using System;

## From Java to C#

- o It's really the same (Java == C#)
  - o java.lang.Object == System.Object
  - o final == sealed for unextendable classes
  - o A jar is like an assembly
  - o Packages are like namespaces
  - o Lots of keywords that are different with the same functionality
  - o Both do managed code
  - o No global methods
  - o Immutable Strings

## ... but it's Different

- o All basic types are objects
- o const is not entirely like final
  - o const/readonly
- o Java's byte is sbyte in C# and C# has a true unsigned byte type
- o float blah[]; must be written as float[] blah;
- o Some different constructs:
  - o foreach
  - o struct

## foreach

- o Similar to foreach statement found in languages such as Perl and PHP

```
string[] randomNames = {"Hitito", "Situamito", "Gito",  
                        "Jiritalossola"};  
foreach (string name in randomNames) {  
    System.Console.WriteLine( "You could be named " +  
                               name);  
}
```

## struct

- o Overhead is associated with making everything an object. Sometimes we want things to be simpler (like basic types for Java) and struct allows this to happen
- o Struct's are passed by value and always allocated on the stack. They do not need to be garbage collected.
- o C# basic types are defined as struct's
- o See example from pg. 48 of the Comparison paper online

## Copy/Paste the Point struct

- o Make a new console project or use an old one
- o Copy/paste the Point struct from the Comparison paper
- o Do a search for "struct Point"
  - o Turn the struct into a class
  - o Make the variables x and y private
  - o Do accessors (get) and mutators (set) for x and y

## Properties

- o Makes writing/accessing get/set methods easier

## Easier?

```
public class Square {
    private double length = 0.0;
    public Square( double length ) {
        this.length = length;
    }
    public double currentLength {
        get{ return length; }
        set{ length = value; }
    }
}

public static void Main(string[] args) {
    Square sq = new Square(4.0);
    Console.WriteLine(sq.currentLength);
    sq.currentLength = 5.0;
    Console.WriteLine(sq.currentLength);
} // end Square
```

## Change Point

- o Change the Point class to use properties
- o Please put an item on top of your monitor when you're done.

## Classes and Inheritance

- o You can have multiple classes in a source file and they don't need the same name as the source file in C#. This is not the case in Java.
- o Access modifiers
- o Properties
- o Syntax for inheritance is different
- o No "super" to call, so calling your superclass is different
- o Virtual methods
- o Overloading operators

## Inheritance Syntax

```
Java:
public class Player extends Entity {
    public Player() { this("Duh"); }
    public Player( String name ) { super(name); }
}
```

```
C#:
public class Player : Entity {
    public Player() : this("Duh") { }
    public Player( string name ) : base( name ) { }
}
```

## Virtual Methods

- o A virtual method is one that can be overridden by a subclass. This is what allows for polymorphism
- o All methods are automatically virtual in Java and therefore are not marked
- o This is not the case in C#
  - o To be overridden, a method can be declared "virtual"
  - o To override a virtual method in the same manner as Java, "override" is used
  - o See example on 29-31 of the Comparison Guide, search for "virtual methods (and final"

## Write a 3D Subclass of Point

- o Name: Point3
- o This class will have the same functionality as the Point class, but will add a third dimension (z)
- o The z dimension should have it's own get/set methods
- o People should give initial values for x, y, and z in the constructor

## Access Modifiers

- o Java: default, public, private, protected
  - o protected allows access by any class in the package and any subclass
  - o default allows access by any class in the package

## Similar, but different

- o C#: public, private, protected, internal, internal protected
  - o protected allows access by any subclass
  - o internal allows access within the assembly
  - o internal protected allows access within the assembly and by any subclasses

## switch

- o You can use string literals as labels
- o No fall through is allowed unless the label is empty
- o goto may be used if fall through is wanted

## switch example

```
string zug="zug";
string ick="ickier";
switch ( ick ) {
    case "icky": zug = "zug";
                break;
    case "ickier": zug = "more zugs";
                  goto case "ickiest";
    case "ickiest": zug = "zugly";
                  break;
    default: zug = "\0";
            break;
}
```

## enum

- o Exist in Java 1.5, but are weird:  
<http://java.sun.com/j2se/1.5.0/docs/guide/language/enums.html>
- o In C#:
  - o enum States { START, RUN, QUIT, ERROR=10};
  - o Equivalent to: const int START = 1; const int RUN = 2; etc.
  - o System.Enum is a class used for Enums
  - o Can set the type of an enum:
    - o enum States : byte { START, RUN, QUIT, ERROR=10};
  - o enum is a type and you can create items of that type:
    - o States currentState = States.START;
  - o enums are only defined at the class level

## Pass by Reference!

- o Java only does pass by value:
  - o Basic types copy their value
  - o Objects copy the address/reference of the object
  - o A true swap function that swaps the values of two integers is impossible to write
- o C# uses this as the normal method of calling, but can also do true pass by reference
  - o See "Pass by Reference" section in the handout
  - o A true swap function is possible to write

## References: understanding

- o Experiment with the pass by reference vs. pass by value example in the handout
  - o Be ready to explain how it works
- o Write a swap function with and without pass by reference that swaps the values of two items inside a method and effects their values outside of the method as well