Welcome to CS3505/5020
Software Practice II

Peter Jensen
Clinical Assistant Professor
Welcome to CS3505! (No 5020 students?)

History and future of this course
- Was Bob’s course, now mine (evil-sounding laugh)

Syllabus key areas:
- Overview – topics
- Teams
- Coursework
- Late days
- Books
- Cheating
Course Overview

We will be focusing on three major areas:

– Small Team Development:
  » Using and developing software requirements, architecture and design
  » Evolving and maintaining code for other teams

– Methodology:
  » Structuring and maintaining the development process

– Programming Interaction:
  » Developing games and/or simulations on top of the XNA framework
  » Emphasis will be on user interaction
**Objectives of the Class**

- **Build on cs3500 skills**
  - Transition from individual to team
  - Independent problem solving
  - Fourth semester of programming practice

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Architecture</th>
<th>Design</th>
<th>Implementation</th>
<th>Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeling - UML</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process – Traditional/Disciplined and Agile</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Language: C#, XNA, ...
Small Team Development

- Most of the homework and projects are team based in this course
  - There are a few individual assignments as well
- We will be working with other teams in development
  - Teams will be assigned different functional areas of at least one project, and will work with other teams to create large, interesting applications.
Teams

- First couple assignments will be individual/pairs
  - Goal is for each of you to learn the technology
- Then we will switch to teams
  - Pairs for a few assignments, 4 person teams later
- Teams will practice agile implementation:
  - Will have weekly sessions together in the lab
  - Will get points for meeting together
  - YOU HAVE TO FIND A TEAM WHERE YOUR SCHEDULES MATCH
  - An ideal meeting time will be the discussion lab times, so consider a team where you can have at least one discussion lab time together
- Start thinking about your teams
- We will have a wiki for helping set this up
Coursework

- **Weekly assignments**
  - Anticipate significant coding and other work

- **Discussion sections**
  - Only for the first few weeks, and as needed throughout the rest of the semester

- **Quizzes**
  - Lighter weight than midterms
  - Focus will be on software engineering methods

- **Final exam**
  - Only if needed
Late days

- **For individual work:**
  - Students have two late days to spend

- **For pair work**
  - Pairs have two late days to spend

- **For teamwork**
  - Teams have two late days to spend

- **Usage of late days**
  - Automatically applied and deducted by timestamp
  - Penalty for exceeding late days – 0 points
  - These are meant for extraordinary circumstances – funerals, illness, travel, winning lottery, etc. No additional late days.
Books

● **UML Distilled, Third Edition**
  - Used for the middle third of the course
  - Not on the shelf at bookstore – order or locate a book

● **Online books**
  - Learning C#
  - Microsoft XNA Unleashed

● **Use online books to save a buck or two**
  - Available through Marriott Library
  - May require campus internet address?
Cheating

- Is there such a thing in a team-based project course?
  - Yes! Some assignments are individual, respect this
  - Teams should not share code with other teams
  - Submitting solutions or work from a previous year (or from a worksite or internet site) counts as cheating

- Why do I care?
  - I gave three E’s last semester to premajors for cheating, I won’t hesitate to help kick out majors who cheat
  - This stuff is too much fun to be lazy about it – do the work and enjoy your results
Survival Guide

● As in 3500, we will start out of the gate with new ideas, technologies and more
  – Doing the labs/tutorials will help
  – Asking questions and seeking clarification is always encouraged
  – Seek out your own learning resources (web sites, books, etc.) that fit your needs
    » This is a key!!!
    » Keep developing your skills to find helpful learning resources, sample codes, etc.
  – Start early (as usual 😊)

● Discussions – we will use a few early for the technology parts and then stop
  – This is only a 3 credit course!
Experienced students

- Speak to me immediately after class if:
  - You have been programming in C# for a while,
  - You have used some form of software methodology in industry, and
  - You want to help out!
Course Technology

History and future of this course
  - Was Bob’s course, now mine (evil-sounding laugh)

Syllabus key areas:
  - Overview – topics
  - Teams
  - Coursework
  - Late days
  - Books
  - Cheating
In this course, we will be using C# and XNA technologies to develop applications

- New Xbox Live Community

XNA allows you to develop Windows or Xbox 360 games

We have 10 Xbox 360 consoles in EMCB 130

You may (should?) buy an Xbox 360 game pad (wired) to help in programming game interactions

- Works on both Xbox 360 and Windows
- Or get wireless to USB controller for your own wireless pad
Software Requirements

- Continue to use MSDNAA for resources
- Visual Studio .NET 2008
  - Have to upgrade to 2008 because of XNA 3.0
  - Note – prefer the Team Suite Edition
  - 2005 and 2008 can coexist if you have the disk space
- Visio 2003 + Phruby UML 2.0 templates
- Microsoft XNA 3.0
- Drivers for game pad
C# and .NET Technology

- While C# and .NET do borrow ideas from Java, C# and .NET are distinct technologies
  - There is a learning curve, even for experienced Java programmers
    » Hopefully, the curve is less steep than learning C++
  - It often helpful to view C# as its own language, and not just a “Microsoft Java”
    » Even though many things will seem very similar to Java
  - Simple language features are 95% (or more) of your programs, don’t worry about the advanced features until you need them or want them (which may be never).
Why Use C# and .NET?

- **To give you a chance to use this growing development technology**
  - This, combined with your Java and C++ experience, will give you a very balanced programming language background.
  - You can compare and contrast the advantages and disadvantages of each platform for various projects.

- **Plus XNA is currently only available with C#**
Why do we focus on games?

● **Games and simulations are non-trivial**
  – Skills acquired in game development are highly portable
  – Games require a diverse skill set: algorithms, OS principles, networking knowledge, UI design skills, etc.
  – Games often make use of cutting-edge technologies
  – Games offer a quick visual response to your programming decisions – big mistakes cannot hide easily
  – Games can be fun

● **I have worked in this industry and I like teaching in this environment**
  – Call it “Instructor Prerogative”
So, What is “.NET”? 

“.NET” is a marketing term for a family of products
- Tools for developing applications
- Runtime platform for executing applications
- Server products for managing applications
- Value-added services for application enhancement

Examples
- Visual Studio .NET
- .NET Framework Common Language Runtime
- .NET Compact Framework
- Application Center, BizTalk Server, SQL Server, …
Where Did .NET Come From?

- .NET is the next evolutionary step for the Microsoft platform
  - Biggest platform shift since move from DOS to Windows NT
  - Departure from Win32, classic C++, and MSVBVM60.DLL
  - Departure from COM and DCOM
  - Internet standards used for cross-machine integration
  - Common Language Runtime (CLR) used from same-machine integration
  - CLR raises programming model to new level of abstraction
What Does .NET Architecture Look Like?

- Console
- Win Forms
- ASP.NET Web Forms
- ASP.NET Web Service
- Components

.NET Framework Class Libraries
(XML parsing, collections, ...)

Common Language Runtime
(JIT compilation, CAS – Code Access Security, GC, ...)

Native Operating System

Hardware

Programmer Creates
The Common Language Runtime

- The CLR is a type-centric managed execution runtime
  - Types are described using new languages (C#, VB.NET, ...)
  - Compilers emit assemblies (still named DLL or EXE)
  - Assemblies contain intermediate language (IL)
  - IL is just-in-time (JIT) compiled at runtime
    » Note – ALWAYS COMPILED!!
  - Runtime services influence execution
  - Many services automated based on type information
Compilation to Assemblies

- IL-based assemblies enable cross-language development
  - Assemblies are the binary unit of deployment for types
  - Referenced during compilation for syntax checking
  - Loaded at runtime for execution
Just-in-Time (JIT) compilation

- IL is compiled into processor-specific code at runtime
  - Portable and less brittle
  - Optimized for target machine, not development machine
  - Never interpreted!!

```csharp
Point pt;
pt = new Point();
pt.x = 200;
pt.y = 300;
```

IL
```
// Point pt = new Point();
.locals init ([0] class Point pt)
newobj instance void Point::.ctor()
stloc.0

// pt.x = 200;
ldloc.0
ldc.i4 0xc8
stfld int32 Point::x

// pt.y = 300;
ldloc.0
ldc.i4 0x12c
stfld int32 Point::y
```

Intel x86
```
call FD5B0AD8 ; allocate
mov ecx, eax ; ecx == pt
call dword ptr ds:[003E5144h] ; pt ctor
mov dword ptr [ecx+4], 0C8h ; pt.x = 200
mov dword ptr [ecx+8], 12Ch ; pt.y = 300
```
Managed execution is characterized by runtime services
- assembly resolution
- JIT compilation
- garbage collection
- evidence based security
- serialization
- remoting
- web services
- ...

Many services are fully automated by runtime based on type information
Garbage collection

- CLR provides garbage collection for types that desire it
  - not all types are managed this way (more on this later)

- Mitigates most common issues in “manual” memory mgmt
  - ‘forgetting’ to free memory no longer a possibility
  - no danger of double deletes
  - cyclic references no longer imply a leak
  - ...

- GC performance is a key goal
  - optimized to minimize pause times
Building code for the CLR

- Building CLR-based code requires a new compiler
  - Language selection largely a personal choice
  - Programming model, libraries, type system shared by all
  - Visual Studio.NET provides RAD environment
    » C#, VB.NET, MC++, J#
  - Command-line compilers provide batch-oriented build support
    » csc.exe, vbc.exe, cl.exe, ilasm.exe
  - 3rd party compilers for many other languages
    » FORTRAN, COBOL, Scheme, Haskell, Mondrian, Eiffel, Pascal, Java, Mercury, Oberon, Perl, Python, ML...
Example: hello world in C#

class AnyNameYouWant {
    static void Main() {
        System.Console.WriteLine("Hello, World");
    }
}

hello.cs

c:\> csc /t:exe hello.cs
Microsoft (R) Visual C# 2005 Compiler version 8.00.50727.42
for Microsoft (R) Windows (R) 2005 Framework version 2.0.50727
Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.

c:\> hello
Hello, World
What is Up With C#?

- Microsoft describes C# as:
  - “…modeled on C++ syntax, immediately familiar to C++ and Java developers, as well as intuitive new language constructs that greatly simplify development tasks.”

- Other may see it as Microsoft’s version of Java
  - Certainly, it does have many things in common with Java
C# Major Features

- Like Java, C# has:
  - C++ Like Syntax
  - Garbage Collection
  - Targets a Virtual Machine
    » Does not compile directly to native code (like C++)
  - Contains a large set of base libraries for development.
    » Some are part of the .NET standard library, others are provided by Microsoft.
C# and Java Differences

- Some major differences
  - Different libraries (similar functionality, but a different design)
  - Namespaces versus packages
    » Similar, but conventions of library organization is different.
  - Different OO features
    » Java 1.5 added many features that C# 2.0 had, but again, they are different (and aren’t in Java 1.4).
C# Features of Note

● Generics
  - You can think of this as a simpler version of C++ templates
    » Java 1.5 has its own version of this.

● Attributes
  - A powerful feature that allows you to add additional metadata to classes
    » Java 1.5 has a similar feature, called annotations.
C# Features of Note

- **Native Events and Delegates**
  - Events are a first class mechanism in C#
  - To support this (and more), delegates are used

- **Delegates are:**
  - A type safe way to refer to a method.
  - Think back to function pointers in C++
  - Allows methods to be passed to other methods as parameters
  - Delegates can have generic type.
C# Wackiness

- All of this allows stuff like:

```csharp
newList = aList->map(
    int delegate(int val)
    {return val+1;});
```

- Meditate upon this
C# and .NET

● So, C# and .NET is a powerful language and platform
  – C# 3.0 will add even more features for working with data and objects.

● It contains notable differences with Java
  – As such, it is worth learning both.
Homework 1

- Ok, I can’t believe that I’m assigning a homework the very first day
  - What were you going to do with all your free time? 😊

- Preview
  - Tomorrow’s discussion – introduction to C# - write some simple code
  - Homework 1 – write a simple pool table game in C# WITHOUT XNA (posted Wednesday)

- Examples
  - A quick demo in Visual Studio