What this course is about

Not algorithms, languages, operating systems or databases— you already studied that in other classes.

Not business/finance issues— you have or will study that elsewhere.

So what is it? building software faster, better, cheaper using teams of people.

It is concerned with using technologies: processes, models, and tools to make teams produce better code, in less time, with less money.

You will learn about techniques for figuring out what to build, how to build it, and evaluating whether you’ve done what the customer wanted.
The Bad News

You should have taken ECS 140A, (and other prerequisites).

You should really understand data abstractions, static typing, object-oriented programming, etc. C++ (or Java) knowledge is very important. The material presented in this course assume this knowledge.

You will have to complete a significant project in this course.

Unlike other coursework (but like in industry) the work product is not just a program, but a set of documents describing what the system does and how it was built, and how it works.

Your work will be judged by industrial standards: thoroughly tested, bugs documented, etc.

Motto: "Learn through experience"
The Good News
About learning through experience

This course will probably help you get up to speed quickly, and survive on your first job. Specially with: Requirements, Design, Inspection, Testing, etc.

As part of the course, you will produce a system complete with documents and a substantial implementation. If you excel at this course, you can show your work to your employers, who will probably be impressed.

You will learn about really cool, powerful tools for writing web apps.

You will get more done in one quarter (in terms of implementation) than you may have thought possible.

Apart from the project, the course should not be too difficult.

You will become a much more object-oriented, understand software process, architecture, design and coding secrets. Good for interviews.
Logistics

Most stuff should be active now.

Post questions to newsgroup if you find broken links or have questions. Class home page URL has been emailed to you.

Newsgroups:
ucd.class.ecs160 ucd.class.ecs160.d

(Posting policy is linked from class home page)

Most important: Don’t email questions to me or the TA—post to newsgroup.

Course Home Page:
http://smf.cs.ucdavis.edu/ erdem/teaching/160

All details (office hours, TA’s, assignments, answers, readings, hints, etc etc) will be posted there.

Watch the web page and the newsgroup frequently! We will assume everyone is reading it!
Project Details

Form teams of 4 people—by next Tuesday.

Build a web site for repair shop management.

Resist "Eye Candy" temptations.

Follow a defined process: project assignments are intermediate work products.

Assignments due noon on specified date.

20% credit demerit for each day delay; no exceptions.

You are allowed 1 week after grades are returned for any assignment or quiz request a regrade. No Exceptions.

You can run your own web servers—Apache-Tomcat, on CSIF machines. Details will be made available

Either way, must be submitted, installed and demo-ed on CSIF machines!!!

Non ECS-160 use of web site reported to SJA.
More about Project

• Requirements will be vague! Customer (me) may change his mind partway, and ask for something different. (The Real World, (TRW!!))

• "Big Bang" due date, with checking of intermediate milestones (TRW strikes again!). However, you will
  – Be awarded credit for steady progress through the course . . .
  – Penalized less for defects if you are meeting deadlines.

• We will use Java Servlets
  – Downside: learning curve
  – Upside: useful, current technology.
  – Real world experience: Application specific platforms are de regeur.
  – Current technology: server-side scripting.
Teams

Form Teams of 4. Try and get one person who has taken 165A on your team.

Report Team membership to (demir@cs.ucdavis.edu) by Tuesday, Jan 10, noon. If you haven’t, we will assign you as we please! Feel free to use the newsgroup to find team mates.

Either way, you will have to make it work. Welcome to the real world.

Each team member will be held responsible for demonstrating good working knowledge of all aspects of the product.

We are ready and willing to help with team process. Bring problems to us, and we are willing to facilitate discussions. No negative effect on grade (and perhaps even a positive one) if brought to attention in the first 4-5 weeks.
Outline

• Java

• CVS

• Servlets and Tomcat Web Server
Java

- Object oriented

- Syntax similar to C++

```java
public class Hello{
    public static void main(String[] args)
    {
        System.out.println("Hello world.");
    }
}
```

- Hello.java (source file) – javac -> Hello.class (byte code)

- A virtual machine interprets byte code.
  - Byte Code
    Virtual Machine
    Operating System
Java

- Portable
- Strongly typed
- Garbage collection
Unlike C++

- No header files
- No macros
- No pointers — everything is a reference.
- No unions
- No multiple inheritance
public class Hello{
    public static void main(String[] args)
    {
        System.out.println("Hello world.");
    }
}

Save in Hello.java (file name and the class name should be the same.)

Compile: javac Hello.java Output is in Hello.class

Run: java Hello (Note no extension, only the class name is passed.)
Arrays

public class HelloArrays{
    public static void main(String[] args)
    {
        String arr[] = new String[4];
        arr[0] = "Hello";
        arr[1] = " ";
        arr[2] = "world";
        arr[3] = ".\n"
        for (int i = 0; i < arr.length; i++)
        {
            System.out.print(arr[i]);
        }
    }
}
public class HelloArraysRE{
    public static void main(String[] args)
    {
        String arr[] = new String[4];
        arr[0] = "Hello";
        arr[1] = " ";
        arr[2] = "world";
        arr[3] = "\n";
        for (int i = 0; i < arr.length+1; i++)
        {
            try{
                System.out.print(arr[i]);
            }
            catch (ArrayIndexOutOfBoundsException e){
                System.err.println("Caught! " + e);
            }
        }
    }
}
public class HelloException{
    public static void main(String[] args)
    {
        java.io.FileReader f =
            new java.io.FileReader("testfile");
    }
}

Compile: javac HelloException.java

HelloException.java:4: unreported exception ...
Corrected

public class HelloExceptionC{
public static void main(String[] args)
{
    try{
        java.io.FileReader f =
            new java.io.FileReader("testfile");
    } catch(java.io.FileNotFoundException exception){
            System.err.println(exception.getMessage());
    }
}
}
Defining methods

`this`: The implicit parameter available to instance methods.

```java
public class HelloMethods {
    public String message = "";
    public void sayMessage()
    {
        System.out.println(this.message);
    }
}

public class Caller{
    public static void main(String args[])
    {
        h.message="Hello world";
        HelloMethods h = new HelloMethods();
        h.sayMessage();
    }
}
```
Inheritance

- A class (subclass) can extend another class (superclass) through inheritance

- Subclass inherits
  - the methods
  - the variables

- Subclass can
  - Add new methods
  - Add new variables
  - Override methods

- Since subclass is superclass + new definitions, it can be used in place of the superclass.
Example

```java
public class Shape {
    public void draw()
    {
        return;
    }
}

public class Point extends Shape{
    private int x;
    private int y;
    public Point(int x, int y){
        this.x=x;
        this.y=y;
    }
    public void draw()
    {
        System.out.println("Drawing a point at " + x + " " + y);
    }
}

public class Line extends Shape{
    private Point start;
    private Point end;
    public Line(Point start, Point end){
        this.start=start;
        this.end=end;
    }
    public void draw()
    {
        System.out.println("Drawing a line");
    }
}
```

public class ShapeCaller{

    public static void main(String[] args)
    {
        Shape[] shapes=new Shape[2];
        shapes[0]=new Point(1,2);
        shapes[1]= new Line(new Point(1,3),
                            new Point(1,4));
        for (int i=0; i < shapes.length;i++)
        {
            shapes[i].draw();
        }
    }
}
Create your own exception

```java
public class MyException extends Exception {
    public MyException (){
        super();
    }
    public MyException (String s){
        super(s);
    }
}

Caller:

public class ExceptionCaller {
    public static void main (String args[])
    throws MyException {
        throw new MyException("no particular reason");
    }
}
```
Abstract classes

For instance:

```java
public abstract class Shape {
    public abstract void draw();
}
```

Try:

```java
public class ShapeAbstractCaller {
    public static void main(String args[])
    {
        ShapeAbstract s = new ShapeAbstract();
    }
}
```

Compile: javac ShapeAbstractCaller.java

```
ShapeAbstractCaller.java:5: ShapeAbstract is abstract; cannot be instantiated . . .
```

CVS

Concurrent Versions System (aka Concurrent Versioning System)

- Keeps track of changes made to a set of files.

- Allows several developers to concurrently work on a software project.

The CVS server keeps track of the changes.

- Client (software developer) checks-out files from the server.

- Modifies files, add new files etc.

- Checks-in changes
CVS

Start a new project:

1. Set CVSROOT environment variable to root directory (for the project the path of this directory will probably be given to you.)
   e.g.
   (for bash) export CVSROOT=/home/erdem/cvsroot
   (for tcsh) setenv CVSROOT "/home/erdem/cvsroot"

2. run cvs init

3. make a directory for your project, cd into that directory and import it to cvs
   e.g.
   • mkdir 160prj
   • cd 160prj
   • cvs import -m’Initial import’ ecs160prj vendor start
   • Rename the project dir:
     cd ..; mv 160prj 160prj.bak
Working with CVS

Assuming project name ecs160prj

Checkout: `cvs checkout ecs160prj`
This makes a directory named ecs160prj

Update: `cvs update`

Commit: `cvs commit filename` Each commit increases version number by 1.
1.1 — commit → 1.2

The cycle:

1. Checkout the project if you haven’t done before
2. `cd` into project directory (same with project name)
3. run update command
   If you get a message like “conflicts found in filename”
then you should take care of these conflicts. Open the file and fix the conflicts. Your original file is saved to 
.filename.revision_number.

4. If you want to add a file then create the file and run:
cvs add newfilename

5. After you edit your files run the update command and fix conflicts if any

6. Run commit command
CVS tips

You can update to a previous version of a file.

- **cvs update -D "1 hour ago" filename** updates the file to the latest version of the last hour.

- **cvs update -r version_number** updates the file to the given version number.

You can tag your project when you reach a milestone.

For instance, when you reach milestone 1 run:

```
cvs update
cvs commit
cvs rtag milestone_1 ecs160prj
```

You can now continue to work on your project w/o affecting the previous release. When you need to demo your code you can run: **cvs checkout -r milestone_1 ecs160prj** and you will get the version of milestone 1.
You can see the history of file by running `cvs log filename`.

And, you can compare your changes to the repository version by running `cvs diff -u filename`.
Apache Tomcat

Apache Tomcat is a web server.

- Accepts HTTP requests.
- Serves HTML pages.

Apache Tomcat can run Java in response to requests to produce the HTML pages.
http://tomcat.apache.org

Servlets are Java programs that generate responses to the HTTP requests.
A good online book: http://uclibs.org/PID/36703 (only from campus or through library proxy.)