Reading

Mercurial: The Definitive Guide, Bryan O'Sullivan,
http://hgbook.red-bean.com/read/index.html
   Chapters 2, 3, 5.

Unit Testing
   Writing Unit Test section of JUnit FAQ, http://junit.sourceforge.net/doc/faq/faq.htm
Crashing

Last Day to Drop: Feb 2

Last Day to Add: Feb 4
Course Web Site

http://www.eli.sdsu.edu/index.html

CS 580 Spring 10

Lecture Notes
Assignments
Wiki
Mailing List
Course Portal
Syllabus
Reading Assignments
Languages

Java
Smalltalk
Ruby
C#
Scala
Knowing a Language

Basic syntax of the language

Core API
   Good grasp of the common or core API
   Collections, Files, Exceptions, Streams

Language culture - Ways of doing things in each language
   Java Doc
   Searching the API
   Compiling/running code
   Using Smalltalk browsers
   Naming conventions

Object-oriented programming
Client-Server

Client
Initiates peer-to-peer communication
Translate user requests into requests for data from server via protocol
GUI often used to interact with user

Server
Program that waits for incoming communication requests from a client
Extracts requested information from data and return to client
Common Issues

• Authentication
• Authorization
• Data Security
• Privacy
• Protection
• Concurrency
Required of a Programmer

Designing robust protocols
Network programming
Designing usable computer-human interfaces
Good documentation skills
Good debugging skills
Understand the information flow of the company/customer
Mastery of concurrency
Multi-platform development
Database programming
Security
Scale Changes Everything
# Names

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>Smalltalk</th>
<th>C#</th>
<th>Ruby</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>PascalCase</td>
<td>PascalCase</td>
<td>PascalCase</td>
<td>PascalCase</td>
</tr>
<tr>
<td>Method</td>
<td>camelCase</td>
<td>camelCase</td>
<td>PascalCase</td>
<td>foo_bar</td>
</tr>
<tr>
<td>Field</td>
<td>camelCase</td>
<td>camelCase</td>
<td>camelCase</td>
<td>@foo_bar</td>
</tr>
<tr>
<td>Parameter</td>
<td>camelCase</td>
<td>camelCase</td>
<td>camelCase</td>
<td>foo_bar</td>
</tr>
<tr>
<td>Local Variable</td>
<td>camelCase</td>
<td>camelCase</td>
<td>camelCase</td>
<td>foo_bar</td>
</tr>
</tbody>
</table>
x = x + 1 //Add one to x
What does this do?

for i := 1 to Num do
    MeetsCriteria[ i ] := True;
for i := 1 to Num / 2 do begin
    j := i + i;
    while ( j <= Num ) do begin
        MeetsCriteria[ j ] := False;
        j := j + i;
    end;
for i := 1 to Num do
    if MeetsCriteria[ i ] then
        writeln( i, ' meets criteria ' );
What does this do?

for PrimeCandidate:= 1 to Num do
    IsPrime[ PrimeCandidate ] := True;

for  Factor:= 1 to Num / 2  do begin
    FactorableNumber := Factor + Factor ;
    while ( FactorableNumber <= Num ) do begin
        IsPrime[ FactorableNumber ] := False;
        FactorableNumber := FactorableNumber + Factor ;
    end;
end;

for PrimeCandidate:= 1 to Num do
    if IsPrime[ PrimeCandidate ] then
        writeln( PrimeCandidate, ' is Prime ' );
A verses B

public class A {
    public int x;
    public int y;
    public int z;
}

public class B {
    private int x;
    private int y;
    private int z;

    public int getX() { return x; }
    public int getY() { return y; }
    public int getZ() { return z; }

    public void setX(int value) { x = value; }
    public void setY(int value) { y = value; }
    public void setZ(int value) { z = value; }
}