

CS 580 Client-Server Programming  
Spring Semester, 2007  
Doc 1 Introduction  
Jan 18, 2007

Copyright ©, All rights reserved. 2007 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (<http://www.opencontent.org/opl.shtml>) license defines the copyright on this document.

# Course Web Site

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

on-line courses

CS 580 Spring 07

Lecture Notes

Lecture Notes with Ink

Assignments

Wiki

Mailing List

Syllabus

Reading Assignments

# Languages

Java

Smalltalk

Ruby

C#

# 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

and brvtns

	Java	Smalltalk	C#	Ruby
Class	PascalCase	PascalCase	PascalCase	PascalCase
Method	camelCase	camelCase	PascalCase	foo_bar
Field	camelCase	camelCase	camelCase	@foo_bar
Parameter	camelCase	camelCase	camelCase	foo_bar
Local Variable	camelCase	camelCase	camelCase	foo_bar

`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 Mun 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;}  
}
```