ECE 462
Object-Oriented Programming using C++ and Java

Draw Game

Yung-Hsiang Lu
yunglu@purdue.edu
## Package

**com.brackeen.javagamebook.tilegame.sprites**

## Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Creature</strong></td>
<td>A Creature is a Sprite that is affected by gravity and can die.</td>
</tr>
<tr>
<td><strong>Fly</strong></td>
<td>A Fly is a Creature that fly slowly in the air.</td>
</tr>
<tr>
<td><strong>Grub</strong></td>
<td>A Grub is a Creature that moves slowly on the ground.</td>
</tr>
<tr>
<td><strong>Player</strong></td>
<td>The Player.</td>
</tr>
<tr>
<td><strong>PowerUp</strong></td>
<td>A PowerUp class is a Sprite that the player can pick up.</td>
</tr>
</tbody>
</table>
com.brackeen.javagamebook.tilegame.sprites

Class PowerUp

java.lang.Object
   \| sprite
      \| com.brackeen.javagamebook.tilegame.sprites.PowerUp

Direct Known Subclasses:

public abstract class PowerUp
extends Sprite

A PowerUp class is a Sprite that the player can pick up.
Class Creature

public abstract class Creature
extends Sprite

A Creature is a Sprite that is affected by gravity and can die. It has four Animations: moving left, moving right, dying on the left, and dying on the right.
public abstract class Creature extends Sprite {

/**
   * Amount of time to go from STATE_DYING to STATE_DEAD.
*/
private static final int DIE_TIME = 1000;

public static final int STATE_NORMAL = 0;
public static final int STATE_DYING = 1;
public static final int STATE_DEAD = 2;

private Animation left;
private Animation right;
private Animation deadLeft;
private Animation deadRight;
private int state;
private long stateTime;

/**
   * Creates a new Creature with the specified Animations.
*/
public Creature(Animation left, Animation right,
    Animation deadLeft, Animation deadRight)
```java
public void loadCreatureSprites() {

    Image[][] images = new Image[4][];

    // load left-facing images
    images[0] = new Image[] {
        loadImage("player1.png"),
        loadImage("player2.png"),
        loadImage("player3.png"),
        loadImage("fly1.png"),
        loadImage("fly2.png"),
        loadImage("fly3.png"),
        loadImage("grub1.png"),
        loadImage("grub2.png"),
    };

    images[1] = new Image[images[0].length];
    images[2] = new Image[images[0].length];
    images[3] = new Image[images[0].length];
    for (int i=0; i<images[0].length; i++) {
        // right-facing images
        images[1][i] = getMirrorImage(images[0][i]);
        // left-facing "dead" images
```
private Image getScaledImage(Image image, float x, float y) {

    // set up the transform
    AffineTransform transform = new AffineTransform();
    transform.scale(x, y);
    transform.translate(
            (x-1) * image.getWidth(null) / 2,
            (y-1) * image.getHeight(null) / 2);

    // create a transparent (not translucent) image
    Image newImage = gc.createCompatibleImage(
            image.getWidth(null),
            image.getHeight(null),
            Transparency.BITMASK);

    // draw the transformed image
    Graphics2D g = (Graphics2D)newImage.getGraphics();
    g.drawImage(image, transform, null);
    g.dispose();

    return newImage;
}
java.awt.geom

Class AffineTransform

java.lang.Object
   java.awt.geom.AffineTransform

All Implemented Interfaces:
   Serializable, Cloneable

public class AffineTransform
   extends Object
   implements Cloneable, Serializable

The AffineTransform class represents a 2D affine transform that performs a linear mapping from 2D coordinates to other 2D coordinates that preserves the "straightness" and "parallelness" of lines. Affine transformations can be constructed using sequences of translations, scales, flips, rotations, and shears.
public void draw(Graphics2D g, TileMap map,
        int screenWidth, int screenHeight)
{
    Sprite player = map.getPlayer();
    int mapWidth = tilesToPixels(map.getWidth());

    // get the scrolling position of the map
    // based on player's position
    int offsetX = screenWidth / 2 -
        Math.round(player.getX()) - TILE_SIZE;
    offsetX = Math.min(offsetX, 0);
    offsetX = Math.max(offsetX, screenWidth - mapWidth);

    // get the y offset to draw all sprites and tiles
    int offsetY = screenHeight -
        tilesToPixels(map.getHeight());

    // draw black background, if needed
    if (background == null ||
        screenHeight > background.getHeight(null))
    {
        g.setColor(Color.black);
        g.fillRect(0, 0, screenWidth, screenHeight);
    }
```java
// draw the visible tiles
int firstTileX = pixelsToTiles(-offsetX);
int lastTileX = firstTileX +
pixelsToTiles(screenWidth) + 1;
for (int y=0; y<map.getHeight(); y++) {
    for (int x=firstTileX; x <= lastTileX; x++) {
        Image image = map.getTile(x, y);
        if (image != null) {
            g.drawImage(image,
                        tilesToPixels(x) + offsetX,
                        tilesToPixels(y) + offsetY,
                        null);
        }
    }
}

// draw player
    g.drawImage(player.getImage(),
                Math.round(player.getX()) + offsetX,
                Math.round(player.getY()) + offsetY,
                null);

// draw sprites
```
Large Background

background + tile map

visible on screen
Manage Visible Objects

- An object is visible if the location is between offsetX and offsetX + xresolution
- In a game with few moving objects, it is acceptable to draw every object. If it is outside the visible region, it cannot be seen.
- If the background moves slowly, it gives the impression of distance. This allows a smaller image for the background and saves memory.
ECE 462
Object-Oriented Programming using C++ and Java

Network Clients

Yung-Hsiang Lu
yunglu@purdue.edu
Networking, Client-Server Model

server
- google.com
- cnn.com
- amazon.com
- purdue.edu

client
- your laptop

open a socket
request a port
listen to the port
create a thread
serve the client

client connect to the port at the machine

allow other clients to connect
ping www.google.com
PING www.1.google.com (64.233.167.99) 56(84) bytes of data.
64 bytes from py-in-f99.google.com (64.233.167.99): icmp p_seq=0 ttl=243 time=12.7 ms
64 bytes from py-in-f99.google.com (64.233.167.99): icmp p_seq=1 ttl=243 time=12.6 ms
64 bytes from py-in-f99.google.com (64.233.167.99): icmp p_seq=2 ttl=243 time=12.9 ms
64 bytes from py-in-f99.google.com (64.233.167.99): icmp p_seq=3 ttl=243 time=12.9 ms

--- www.1.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 12.686/12.832/12.977/0.162 ms, pipe 2
[(qstruct04) ~/] traceroute www.mit.edu
traceroute to www.mit.edu (18.7.22.83), 30 hops max, 46 byte packets
  1  msee-b003-c6506-01.ecn-101.ecn.purdue.edu (128.46.1 01.1)  0.370 ms  0.235 ms  0.227 ms
  2  172.19.124.1 (172.19.124.1)  0.635 ms  0.587 ms  0.281 ms
  3  192.31.0.5 (192.31.0.5)  0.369 ms  0.311 ms  0.317 ms
  4  192.31.0.101 (192.31.0.101)  0.472 ms  5.833 ms  0.343 ms
  5  192.31.0.13 (192.31.0.13)  0.418 ms  0.373 ms  0.475 ms
  6  lynn-b168-m10i-01-campus.tcom.purdue.edu (192.5.40. 58)  3.727 ms  0.471 ms  0.523 ms
  7  gigapop-ctc-re-t640.tcom.purdue.edu (192.5.40.134)  1.989 ms  1.603 ms  1.553 ms
  8  149.165.254.226 (149.165.254.226)  7.608 ms  7.506 ms
  9  newy-chic-100.layer3.nlrl.net (216.24.186.33)  30.29 ms
  2  ms  29.759 ms  29.598 ms
List of TCP and UDP port numbers

From Wikipedia, the free encyclopedia

Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are transport layer protocols used for communication between computers. The Internet Assigned Numbers Authority (IANA) is responsible for assigning port numbers for specific uses.

Contents

1 Ranges
2 Port status colors
3 Well Known Ports: 0–1023
4 Registered Ports: 1024–49151
5 Dynamic and/or Private Ports: 49152–65535
6 References
7 External links

Ranges

The port numbers are divided into three ranges:[1]:

- Well Known Ports are those in the range 0–1023. On Unix-like operating systems, opening a port in this range requires invocing operating system-specific administrative privileges or processes running as root.
- Registered Ports are those in the range 1024–49151. These ports require only normal user privileges for opening.
- Dynamic and Private Ports are those in the range 49152–65535. These ports are available for any user, although they are not guaranteed to be persistent across system restarts.
<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Officiality</th>
</tr>
</thead>
<tbody>
<tr>
<td>53/TCP,UDP</td>
<td>Domain Name System (DNS)</td>
<td></td>
</tr>
<tr>
<td>54/TCP,UDP</td>
<td>XNS (Xerox Network Services) Clearinghouse</td>
<td></td>
</tr>
<tr>
<td>55/TCP,UDP</td>
<td>XNS (Xerox Network Services) Authentication</td>
<td></td>
</tr>
<tr>
<td>56/TCP,UDP</td>
<td>RAP (Route Access Protocol)</td>
<td></td>
</tr>
<tr>
<td>57/TCP</td>
<td>MTP, Mail Transfer Protocol</td>
<td></td>
</tr>
<tr>
<td>58/TCP,UDP</td>
<td>XNS (Xerox Network Services) Mail</td>
<td></td>
</tr>
<tr>
<td>67/UDP</td>
<td>Bootstrap Protocol (BOOTP) Server, also used by Dynamic Host Configuration Protocol (DHCP)</td>
<td></td>
</tr>
<tr>
<td>68/UDP</td>
<td>Bootstrap Protocol (BOOTP) Client, also used by Dynamic Host Configuration Protocol (DHCP)</td>
<td></td>
</tr>
<tr>
<td>69/UDP</td>
<td>Trivial File Transfer Protocol (TFTP)</td>
<td></td>
</tr>
<tr>
<td>70/TCP</td>
<td>Gopher protocol</td>
<td></td>
</tr>
<tr>
<td>79/TCP</td>
<td>Finger protocol</td>
<td></td>
</tr>
<tr>
<td>80/TCP</td>
<td>Hypertext Transfer Protocol (HTTP)</td>
<td></td>
</tr>
<tr>
<td>81/TCP</td>
<td>Torpark—Onion routing</td>
<td></td>
</tr>
<tr>
<td>82/UDP</td>
<td>Torpark—Control</td>
<td></td>
</tr>
<tr>
<td>83/TCP</td>
<td>MIT ML Device</td>
<td></td>
</tr>
<tr>
<td>88/TCP</td>
<td>Kerberos—authentication system</td>
<td></td>
</tr>
</tbody>
</table>
Client States

• Does a later request from the same client depend on an earlier request?
  No ⇒ stateless
  Yes ⇒ stateful

• For stateful requests, the client, the server, or both have to store information based on earlier requests.
Description:

**Protocol suite:** TCP/IP.

**Type:** Application layer file transfer protocol.

**Ports:** HTTP: 80, 8008, 8080 (TCP) server.
S-HTTP: 80 (TCP) server.
HTTPS: 443 (TCP) server over SSL/TLS.

**Related protocols:** webDAV, Web Distributed Authoring and Versioning.

**URI:** http:, https:

**MIME subtype:** application/http, message/http, message/s-http.

**Working groups:** http, HyperText Transfer Protocol.
httpbis, Hypertext Transfer Protocol Bis.
webdav, WWW Distributed Authoring and Versioning.

**Links:** HTTP, Object Header links.
HTTP status codes.
HTTP message:

Methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE</td>
<td>RFC 1945</td>
</tr>
<tr>
<td>GET</td>
<td>RFC 1945</td>
</tr>
<tr>
<td>HEAD</td>
<td>RFC 1945</td>
</tr>
<tr>
<td>LINK</td>
<td>RFC 1945</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>RFC 2068</td>
</tr>
<tr>
<td>PATCH</td>
<td>RFC 2068</td>
</tr>
<tr>
<td>POST</td>
<td>RFC 1945</td>
</tr>
<tr>
<td>PUT</td>
<td>RFC 1945</td>
</tr>
<tr>
<td>TRACE</td>
<td>RFC 2068</td>
</tr>
<tr>
<td>UNLINK</td>
<td>RFC 1945</td>
</tr>
</tbody>
</table>

Header fields:
Java HTTP Client
```java
//ClientSocket.java
import java.io.*;
import java.net.*;
class ClientSocket {
    public static void main(String[] args) {
        try {
            String webAddress = args[0];
            String hostHeader = "Host: " + webAddress;
            Socket socket = new Socket(webAddress, 80);
            OutputStream os = socket.getOutputStream();
            PrintStream ps = new PrintStream(os, true);
            InputStream in = socket.getInputStream();
            InputStreamReader in_reader = new InputStreamReader(in);
            BufferedReader b_reader = new BufferedReader(in_reader);
            ps.println("GET / HTTP/1.0\r\n" + hostHeader + "\r\n" + "\r\n");
            boolean more = true;
            while (more) {
                String str = b_reader.readLine();
                if (str == null)
                    more = false;
                System.out.println(str);
            }
        } catch (IOException e) {
            System.out.println("Error: " + e);
        }
    }
}
```
[(qstruct04) ~/lecturecode/1105/java/ ] java clientSocket et www.google.com
HTTP/1.0 200 OK
Cache-Control: private, max-age=0
Date: Wed, 02 Jul 2008 14:48:20 GMT
Expires: -1
Content-Type: text/html; charset=ISO-8859-1
Set-Cookie: PREF=ID=49176524815f28e0:TM=1215010100:LM=1215010100:S=_kSS5YgXK0Jm875y; expires=Fri, 02-Jul-2010 14:48:20 GMT; path=/; domain=.google.com
Server: gws
Connection: close

<html><head><meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
<title>Google</title>
</head><body>
Network Protocols

• Protocols specify what commands / formats can be sent and the responses / formats.
• Protocols also specify the responses when commands cannot execute.
C++ Qt HTTP Client
Qt Examples

This is the list of examples in Qt’s examples directory. The examples demonstrate Qt features in small, self-contained programs. They are not all designed to be impressive when you run them, but their source code is carefully written to show good Qt programming practices. You can launch any of these programs from the Examples and Demos Launcher application.

If you are new to Qt, you should probably start by going through the Tutorials before you have a look at the Application example.

In addition to the examples and the tutorial, Qt includes a selection of demos that deliberately show off Qt’s features. You might want to look at these as well.

One more valuable source for examples and explanations of Qt features is the archive of the Qt Quarterly.

In the list below, examples marked with an asterisk (*) are fully documented. Eventually, all the examples will be fully documented, but sometimes we include an example before we have time to write
Network

- Blocking Fortune Client*
- Broadcast Receiver
- Broadcast Sender
- Fortune Client*
- Fortune Server*
- FTP
- HTTP
- Loopback
- Threaded Fortune Server*
- Torrent

OpenGL

- 2D Painting*
- Framebuffer Object
- Framebuffer Object 2
- Grabber
- Hello GL*
after qmake -project
before qmake
#include <QtGui>
#include <QtNetwork>
#include <QTcpServer>
#include <string>
using namespace std;

class ClientSocket : public QObject {
    Q_OBJECT
    string wwwName;
    QTcpSocket* socket;
    string constructHttpRequest();

public:
    ClientSocket( string name );
    ~ClientSocket();

public slots:
    void reportConnected();
    void reportHostFound();
    void getWebPage();
    void socketConnectionClosed();
};

#endif
ClientSocket::ClientSocket( string siteName )
{
    wwwName = siteName;
    socket = new QTcpSocket();
    connect( socket, SIGNAL( connected() ),
             this, SLOT( reportConnected() ) );
    connect( socket, SIGNAL( hostFound() ),
             this, SLOT( reportHostFound() ) );
    connect( socket, SIGNAL( readyRead() ),
             this, SLOT( getWebPage() ) );
    connect( socket, SIGNAL( connectionClosed() ),
             this, SLOT( socketConnectionClosed() ) );
    QString qstr( wwwName.c_str() );
    socket->connectToHost( qstr, 80 );
}

ClientSocket::~ClientSocket() {
    if (socket) {
        delete socket;
    }
}

string ClientSocket::constructHttpRequest() {

}
string ClientSocket::constructHttpRequest() {
    string hostHeader = "Host: " + wwwName;
    string urlString( hostHeader );
    string httpRequestString = "GET / HTTP/1.0\r\n" +
                               urlString + "\r\n" + "\r\n";
    return httpRequestString;
}

void ClientSocket::reportHostFound() {
    // cout << "host found" << endl;
}

void ClientSocket::reportConnected() {
    // cout << "connection established" << endl;
    string httpRequest = constructHttpRequest();
    socket->write( httpRequest.c_str() );
}

void ClientSocket::getWebResponse() {
    // cout << "socket ready to read" << endl;
    int howManyBytes = socket->bytesAvailable();
    // cout << "bytes available: " << howManyBytes << endl;
    char data[howManyBytes];
void ClientSocket::getWebPage() {
    // cout << "socket ready to read" << endl;
    int howManyBytes = socket->bytesAvailable();
    // cout << "bytes available: " << howManyBytes << endl;
    char data[howManyBytes];
    socket->read( data, howManyBytes );
    cout << data;
    cout.flush();
}

void ClientSocket::socketConnectionClosed() {
    // cout << "socketConnectionClosed" << endl;
    exit(0);
}

int main( int argc, char* argv[] )
{
    if (argc > 1)
    {
        QApplication app( argc, argv );
        ClientSocket sock( argv[1] );
        return app.exec();
    }
ECE 462
Object-Oriented Programming using C++ and Java

Network Server

Yung-Hsiang Lu
yunglu@purdue.edu
C++ Qt Chat Server
#include <QNetwork>
#include <QString>
#include <QThread>
#include <QApplication>
#include <vector>
using namespace std;

class ChatServer;

class ClientHandler : public QObject
{
  Q_OBJECT
private:
  QString ch_name;
  QTcpSocket* ch_socket;
  QTextStream* ch_os;
public:
  ClientHandler( QTcpSocket* sock);
  virtual ~ClientHandler();
  static ChatServer * ch_server;
private slots:
  void readFromClient();
  friend class ChatServer;
};
void readFromClient();
friend class ChatServer;
};

class ChatServer: public QObject
{
  Q_OBJECT
private:
  QTcpServer * cs_server;
  QList<ClientHandler *> cs_clientList;
public:
  ChatServer();
  virtual ~ChatServer();
  void broadcastClient(QString);
public slots:
  void connectNewClient();
};
#endif
// ChatServer.cc
#include "ChatServer.h"
#include <iostream>
using namespace std;

ChatServer * ClientHandler::ch_server = 0;
ChatServer::ChatServer()
{
    cs_server = new QTcpServer();
    if (! cs_server->listen())
    {
        qWarning( "Failed to register the server port" );
        exit( 1 );
    }
    cout << "Server port " << cs_server->serverPort() << endl;
    connect(cs_server, SIGNAL(newConnection()),
            this, SLOT(connectNewClient()));
    ClientHandler::ch_server = this;
}

void ChatServer::connectNewClient()
{
    QTcpSocket* socket = cs_server->nextPendingConnection();
}
```cpp
void ChatServer::connectNewClient()
{
    QTcpSocket* socket = cs_server->nextPendingConnection();
    ClientHandler* clh = new ClientHandler( socket );
    cs_clientList.push_back(clh);
    cout << "A new client connected" << endl;
}

void ChatServer::broadcastClient(QString message)
{
    ClientHandler * handler;
    for (int index = 0; index < cs_clientList.size(); index++)
    {
        handler = cs_clientList[index];
        *(handler->ch_os) << message;
        (handler->ch_os) -> flush();
    }

    // A chatter's terminal always shows name at beginning
    // of a new line.
    for (int index = 0; index < cs_clientList.size(); index++)
    {
        ClientHandler * handler = cs_clientList[index];
```
for (int index = 0; index < cs_clientList.size(); index ++)
{
    ClientHandler * handler = cs_clientList[index];
    *(handler->ch_os) << (handler->ch_name + "": ");
    (handler->ch_os) -> flush();
}

ChatServer::~ChatServer()
{
    if (cs_server)
    {
        delete cs_server;
    }
    ClientHandler* clh = cs_clientList.takeFirst();
    while (clh != 0)
    {
        delete clh;
        clh = cs_clientList.takeFirst();
    }
}

ClientHandler::ClientHandler( QTcpSocket* socket)
    : ch_name(""),
    ch_socket( socket )

--(Unix)-- chatserver.cc  (C++ Abbrev)--L53--39%---------------------
ClientHandler::ClientHandler( QTcpSocket* socket)
    : ch_name(""'),  
    ch_socket( socket )
{
    ch_os = new QTextStream( ch_socket );

    (*ch_os) << "Welcome to a chat room powered by C++\n";
    (*ch_os) << ">>>>> Enter 'bye' to exit <<<\n";
    (*ch_os) << "Enter chat name: ";
    ch_os -> flush();
    connect( ch_socket, SIGNAL( readyRead() ),
             this, SLOT( readFromClient() ) );
}

ClientHandler::~ClientHandler()
{
    if (ch_os)
        { delete ch_os; }
}

void ClientHandler::readFromClient() { 
    QTcpSocket* sock = (QTcpSocket*) sender(); 
    while ( sock->canReadLine() ) {
```cpp
void ClientHandler::readFromClient() {
    QTcpSocket* sock = (QTcpSocket*) sender();
    while (sock->canReadLine()) {
        QString qstr = sock->readLine();
        qstr = qstr.trimmed(); // remove white space
        if (ch_name == "")
            { ch_name = qstr;
              QString outgoing = "\nMessage from chat server: " +
                                ch_name + " signed in\n";
              ch_server -> broadcastClient(outgoing);
            }
        else if (qstr == "bye")
            { QString outgoing = "\nMessage from the chat server: " +
                          ch_name + " signed off\n";
              ch_server -> broadcastClient(outgoing);
              ch_socket->close();
              ch_socket = 0;
            }
        else
            { QString outgoing = "\n" + ch_name + ": " + qstr + "\n";
```

else if ( qstr == "bye" )
{
    QString outgoing = "\nMessage from the chat server: " +
    ch_name + " signed off\n";
    ch_server -> broadcastClient(outgoing);
    ch_socket->close();
    ch_socket = 0;
}
else
{
    QString outgoing = "\n" + ch_name + ": " + qstr + "\n";
    ch_server -> broadcastClient(outgoing);
}

int main( int argc, char* argv[] )
{
    QApplication app( argc, argv );
    ChatServer server;
    return app.exec();
}
QObject * QObject::sender () const [protected]

Returns a pointer to the object that sent the signal, if called in a slot activated by a signal; otherwise it returns 0. The pointer is valid only during the execution of the slot that calls this function.

The pointer returned by this function becomes invalid if the sender is destroyed, or if the slot is disconnected from the sender's signal.

**Warning:** This function violates the object-oriented principle of modularity. However, getting access to the sender might be useful when many signals are connected to a single slot.

See also [QSignalMapper](http://doc.trolltech.com/4.4/qobject.html#sender).

```cpp
void QObject::setParent ( QObject * parent )
```
C++ Qt Server with Long Latency
bool IsPrime(int num)
{
    int factor = 2;
    // a very slow way to compute
    do
    {
        if ((num % factor) == 0)
        {
            return false;
        }
        factor ++;
    } while (factor < num);
    return true;
}

long TotalPrime(long maxValue)
{
    long sum = 0;
    if (maxValue <= 1) { return 0; } // no prime number
    if (maxValue == 2) { return 1; } // 2 is prime
    if (maxValue >= 3) { sum = 2; } // 2 and 3 are prime
    for (int curNum = 4; curNum <= maxValue; curNum ++)
    {
        if (IsPrime(curNum) == true)
        {
            sum ++;
        }
    }
    return sum;
}
```cpp
void ClientHandler::readFromClient() {
    QTcpSocket* sock = (QTcpSocket*) sender();
    while (sock->canReadLine()) {
        QString qstr = sock->readLine();
        qstr = qstr.trimmed();
        if (qstr == "bye") {
            (*ch_os) << "Bye
"
            ch_os -> flush();
            ch_socket->close();
            ch_socket = 0;
        } else {
            long maxValue = qstr.toLong();
            long total = TotalPrime(maxValue);
            QString outgoing = "\n There are " +
            QString::number(total) +
            " prime numbers between 0 and " +
            QString::number(maxValue) +
            "\n";
            (*ch_os) << outgoing;
            (*ch_os) << "Give a number or say \"bye\": ";
            ch_os -> flush();
        }
    }
```
Server with Threads
```cpp
#ifndef PRIMESERVER_H
#define PRIMESERVER_H

#include <QtNetwork>
#include <QString>
#include <QThread>
#include <QApplication>
#include <vector>

using namespace std;

class PrimeThread : public QThread
{
public:
    PrimeThread(long v, QTextStream* s);
    void run();
    long total;
    QTextStream* os;
    long maxValue;
};

class PrimeServer;

class ClientHandler : public QObject
{
    Q_OBJECT
private:

```
```cpp
class PrimeServer;
class ClientHandler : public QObject
{
    Q_OBJECT
private:
    QTcpSocket* ch_socket;
    QTextStream* ch_os;
public:
    ClientHandler( QTcpSocket* sock);
    virtual ~ClientHandler();
public slots:
    void answerReady();
private slots:
    void readFromClient();
    friend class PrimeServer;
};

class PrimeServer: public QObject
{
    Q_OBJECT
private:
    QTcpServer * cs_server;
    QList<ClientHandler *> cs_clientList;
public:
```
```
```
PrimeThread::PrimeThread(long v, QTextStream* s)
{
    maxValue = v;
    os = s;
}

void PrimeThread::run()
{
    total = TotalPrime(maxValue);
}

PrimeServer::PrimeServer()
{
    cs_server = new QTcpServer();
    if (!cs_server->listen())
    {
        qWarning("Failed to register the server port");
        exit(1);
    }
    cout << "Server port " << cs_server->serverPort() << endl;
    connect(cs_server, SIGNAL(newConnection()),
            this, SLOT(connectNewClient()));
}

void PrimeServer::connectNewClient()
void ClientHandler::readFromClient()
{
    QTcpSocket* sock = (QTcpSocket*) sender();
    while (sock->canReadLine()) {
        QString qstr = sock->readLine();
        qstr = qstr.trimmed();
        if (qstr == "bye")
        {
            (*ch_os) << "Bye\n\n";
            ch_os->flush();
            ch_socket->close();
            ch_socket = 0;
        }
        else
        {
            long maxValue = qstr.toLong();
            PrimeThread *pt = new PrimeThread(maxValue, ch_os);
            pt->start();
            connect(pt, SIGNAL(finished()),
                    this, SLOT(answerReady()));
        }
    }
}

void ClientHandler::answerReady()
{
else
{
    long maxValue = qstr.toLong();
    PrimeThread * pt = new PrimeThread(maxValue, ch_os);
    pt -> start();
    connect(pt, SIGNAL( finished() ),
            this, SLOT( answerReady() ) );
}

void ClientHandler::answerReady()
{
    PrimeThread* pt = (PrimeThread*) sender();
    QString outgoing = "\n There are " +
    QString::number(pt -> total) +
    " prime numbers between 0 and " +
    QString::number(pt -> maxValue)
    + "\n";
    (* (pt -> os)) << outgoing;
    (* (pt -> os)) << "Give a number or say \"bye\": ";
    (pt -> os) -> flush();
    delete pt;
}
Further Performance Improvement

• implement an efficient algorithm to count prime numbers
• create a thread only when the input number is large
• reuse threads
ECE 462
Object-Oriented Programming using C++ and Java

Test Coverage

Yung-Hsiang Lu
yunglu@purdue.edu
**Code Coverage Plugin**

**NetBeans Beta Plugin**

**Version:** 2.2.1  
**Author:** Mikhail Kondratyev (mikk@netbeans.org)  
**Date:** 12/14/07  
**Source:** NetBeans Beta  
**Homepage:** http://codecoverage.netbeans.org/

**Plugin Description**

The Code Coverage Plugin is a NetBeans 6.0 plugin that enhances the existing NetBeans functionality with new code coverage features. The plugin works as a transparent additional service that colors all Java files according to the unit test coverage information. With code Coverage plugin enabled user can continue to work with his/her project in the usual way but can easily view the test coverage of the project classes. The code coverage plugin will update the code coverage data and refresh editors markup every time a unit test is executed for the project. Current the Java Application, Java Library, Java Project with Existing Sources and NetBeans module projects are supported.
public class coverage {

c static void main(String[] args) {
    double a = Math.random();
    double b = Math.random();
    double c = Math.random();
    double d = Math.random();
    if (a > b) {
        System.out.println("a > b " + a + " " + b);
    } else {
        System.out.println("a <= b " + a + " " + b);
    }
    if (c > d) {
        System.out.println("c > d " + c + " " + d);
    } else {
        System.out.println("c <= d " + c + " " + d);
    }
    if ((a + c) < b) {
        if (a > b) {
            System.out.println("(a + c) < b and a > b ");
        } else {
            System.out.println("(a + c) < b and a <= b ");
        }
    }
}
yHL Test Coverage 5
public class coverage {
    public static void main(String[] args) {
        double a = Math.random();
        double b = Math.random();
        double c = Math.random();
        double d = Math.random();
        if (a > b) {
            System.out.println("a > b");
        } else {
            System.out.println("a < b");
        }
        if (c > d) {
            System.out.println("c > d");
        } else {
            System.out.println("c < d");
        }
        if ((a + c) < (b + d)) {
            if (a > b) {
                System.out.println("a > b and a < b");
            } else {
                System.out.println("a < b and a > b");
            }
        } else {
            System.out.println("c + d > a + b");
        }
    }
}
Select JUnit Version

Select JUnit version for which the created test skeletons should be created:

- JUnit 3.x
- JUnit 4.x

[Select] [Cancel] [Help]
Create Tests

Class to Test: coverage
Class Name: coverageTest
Location: Test Packages

Code Generation

Method Access Levels
- Public
- Protected
- Package Private

Generated Code
- Test Initializer
- Test Finalizer
- Default Method bodies

Generated Comments
- Javadoc Comments
- Source Code Hints

OK  Cancel  Help
```java
public class coverage {

    public static void main(String[] args) {
        double a = Math.random();
        double b = Math.random();
        double c = Math.random();
        double d = Math.random();
        if (a > b) {
            System.out.println("a > b " + a + " + b");
        } else {
            System.out.println("a <= b " + a + " + b");
        }
        if (c > d) {
            System.out.println("c > d " + c + " + d");
        } else {
            System.out.println("c <= d " + c + " + d");
        }
        if ((a + c) < b) {
            if (a > b) {
                System.out.println("(a + c) < b and a > b ");
            } else {
                System.out.println("(a + c) < b and a <= b ");
            }
        }
    }
}
```
public class coverage {

    public static void main(String[] args) {
        double a = Math.random();
        double b = Math.random();
        double c = Math.random();
        double d = Math.random();
        if (a > b) {
            System.out.println("a > b " + a + " " + b);
        } else {
            System.out.println("a <= b " + a + " " + b);
        }
        if (c > d) {
            System.out.println("c > d " + c + " " + d);
        } else {
            System.out.println("c <= d " + c + " " + d);
        }
        if ((a + c) < b) {
            if (a > b) {
                System.out.println("(a + c) < b and a > b ");
            } else {
                System.out.println("(a + c) < b and a <= b ");
            }
        }
    }
}
public class coverage {
    public static void main(String[] args) {
        double a = Math.random();
        double b = Math.random();
        double c = Math.random();
        double d = Math.random();
        if (a > b) {
            System.out.println("a > b " + a + "+ " + b);
        } else {
            System.out.println("a <= b " + a + "+ " + b);
        }
        if (c > d) {
            System.out.println("c > d " + c + "+ " + d);
        } else {
            System.out.println("c <= d " + c + "+ " + d);
        }
        if ((a + c) < b) {
            if (a > b) {
                System.out.println("(a + c) < b and a > b ");
            } else {
                System.out.println("(a + c) < b and a <= b ");
            }
        }
    }
}
Unreachable Code

If a, b, and c are zeros or positive numbers
\[(a + c) < b \implies a > b \text{ is impossible}\]
\(\implies\) problem in the logic?

if ((x <= 0) && (x >= width)) // hit left or right wall
{
    // width > 0
    vx = -vx; // change direction
}