A Game Development Process: With Java, Git, and Tic-Tac-Toe

Version 1.0 by Brandon Froberg

July 31, 2013

Introduction

The following document will guide you through a step-by-step "game development process". NO JAVA EXPERIENCE IS NEEDED! However, several prerequisite programs will be needed for completion of this lab. The goal of this lab is to 1) show an example Java game, 2) re-enforce utilizing a local and remote Version Control System (VCS), and 3) renforcing the use of a VCS during active game development. Throughout this lab you will be expected to go from a code editor, to command-line/terminal, to a web browser and back. Fear not and read all directions. May the Force be with you!

Prerequisite "Programs"

1. A Text Editor (I recommend Notepad++).
2. Java Development Kit 7
3. Git Version Control System for a VCS
4. A GitHub account

Please see Appendices A, B, and C for the install/setup/configuration of programs two through four.

THE LAB

The following section assumes that you have all the prerequisite programs installed, configured, and ready to use. Additionally, this lab will only use the programs that are installed to the PATH. Thus all the commands do NOT have full the path of execution (example: `"C:\Program Files (x86)\Notepad++\notepad++.exe"` is simply `notepad++`).

Please note this lab tries to show by example and with pictures on the process of coding. You are NOT expected to fully understand all commands and actions taken. The primary objective is muscle memory in regards to coding practices.

Start your Command Prompt!

The whole execution in the lab will occur on the command line (terminal for you Unix-like folks). Please start the “Run Prompt”: Press (and hold) the Windows button while pressing R. Now type cmd.exe and press enter.

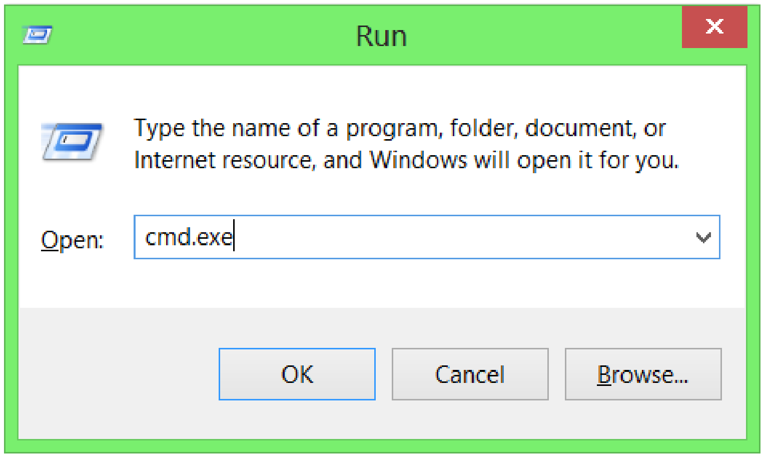
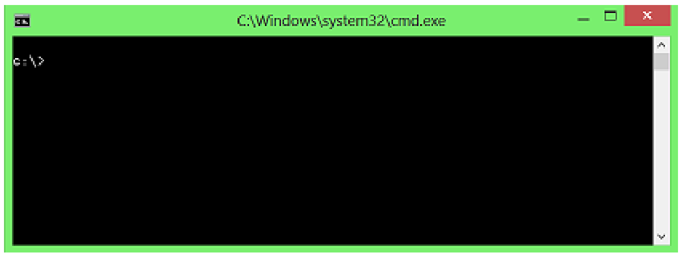

Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
cmd.exe
OK
Cancel
Browse...

Figure 1: Run Prompt with cmd.exe typed

On Linux terminal may be started by pressing Ctrl+Alt+T, and Mac OSX can start terminal by searching for “terminal”.

Once started the prompt you should see something similar to the following:



```
c:\>
```

Figure 2: Windows command shell started

Having this window open type the following command: `mkdir TicTacToe`. This will create a folder having the same name, and we must change directory to it by typing: `cd TicTacTie`. Also type: `explorer .` to open a folder directly where the command prompt is located.

“Git”ting Started

The first step in creating a coding repository with git is to type: `git init`. Git will create a repository in a hidden directory called `.git`. This is where all the code updates and objects are stored. You can see a current status (in our case nothing should be there!) with the command: `git status`.

Create a simple Java file

In this new folder create a new text file name TicTacToe. Java will blow up if you do not match the main method’s name to the `<filename>.java` source file. You **must** change the .txt to .java in order to compile if you just made an ordinary text file. In command prompt type: `move TicTacToe.txt TicTacToe.java` (Linux and Mac Users will need to use *mv* vs *move*). Within the file place the following code:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
```

Save the file and return to the command prompt.

Saving files to the Git Repository

Now type `git status`. Do you see a difference? It should list the new file as “untracked”, so let’s track it! Simply type: `git add *`. This will add ALL files in folder to be tracked! Once again type `git status` and note the differences!

Now we can finally `commit` these tracked changes to the Git repository. You can accomplish this by: `git commit -m "<enter your own message!>"`. At this point Git should list these changes. ADVANCED: you can see the GUI version of git by executing: `gitk`.

Creating a Remote Git Repository

One of the best ways to save your data is by backing up the data on a cloud server. GitHub is one such service that works flawlessly with Git. Open up the website and log into your account. Please note that in the top right of the web page you can find the “create repository” button:

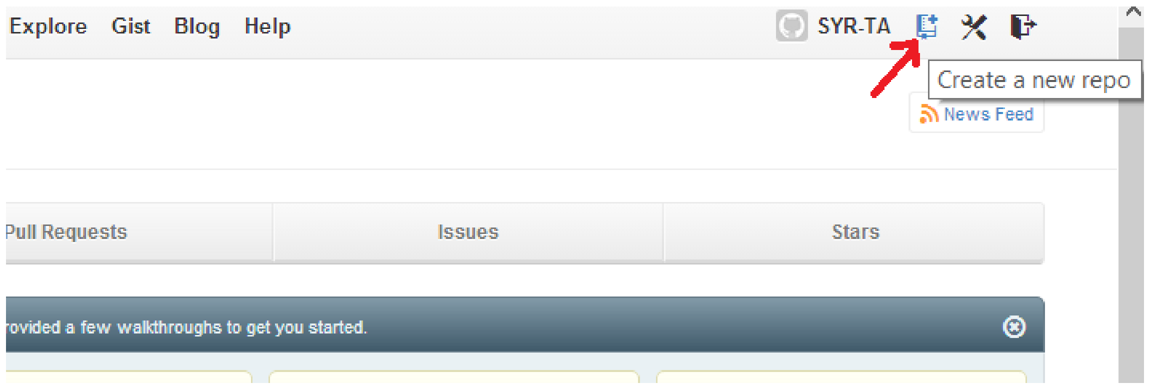
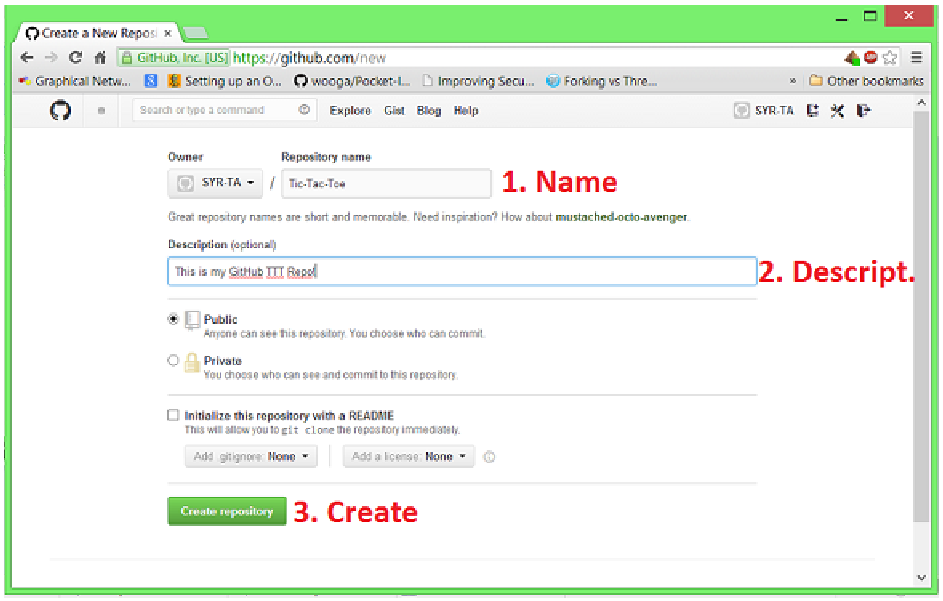


Figure 3: GitHub Create Repository Button

This button will bring you to a new screen, which allows you (with three simple steps) to create an on-line Git repo:



Owner: SYR-TA / Repository name: Tic-Tac-Toe

Great repository names are short and memorable. Need inspiration? How about mustached-octo-avenger.

Description (optional): This is my GitHub TTT Repo!

[✓ Public] Anyone can see this repository. You choose who can commit.

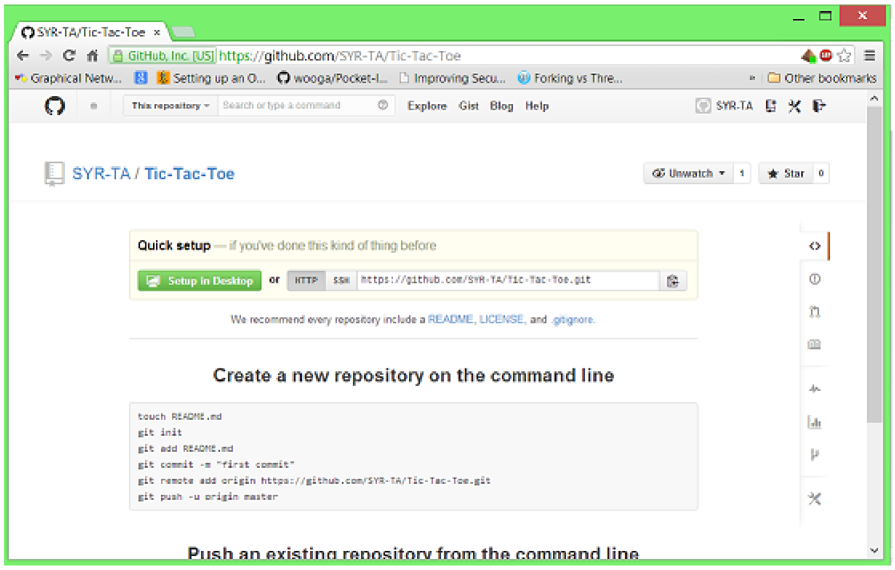
Private You choose who can see and commit to this repository.

Initialize this repository with a README
This will allow you to git clone the repository immediately.

Add .gitignore: None | Add a license: None

Figure 4: GitHub Repository Creation in Three Steps

Feel free to modify the first two steps, BUT remember your changes! After you press the create button you should see the following screen:



SYR-TA / Tic-Tac-Toe

Quick setup — if you've done this kind of thing before

Setup in Desktop or HTTP SSH https://github.com/SYR-TA/Tic-Tac-Toe.git

We recommend every repository include a README, LICENSE, and .gitignore.

Create a new repository on the command line

```
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/SYR-TA/Tic-Tac-Toe.git
git push -u origin master
```

Push an existing repository from the command line

Figure 5: GitHub New Repository

GitHub has created a simple repo with no contents. This is the exact same process as we used in `git init`, however it is executed on the GitHub servers. At this point we can add this remote repository to our Git project; go back to the command prompt!

Adding a Remote GitHub Repository

GitHub does a fantastic job at pre-building commands for Git repositories, however some of the commands are a bit hard to understand. Please refer to Figure 5 (or your GitHub web-page), and notice the section "Create a new repository on the command line". All the commands you need to add GitHub are here, yet we will be changing a specific command to make it more human readable.

The fifth command `"git remote add ..."` only needs one change. Remove the word `origin` and replace it with `GitHub`. On the command prompt type: `git remote add GitHub https://github.com/<username>/<repo name>.git`. This will now *link* your local Git repository to that of your GitHub account's repository.

To actually send your repository to GitHub you will need one final command: `git push GitHub master`. By default all repositories save your code to the *master branch* in Git, therefore the above command tells Git to take your **master** branch and **push** it to the **remote** repository of **GitHub**

Take a Breather

No really. Get up walk around, or go watch a funny YouTube video.

Recap of Work Thus Far

We made a Git repository to save our code, we added a quick snippet of Java, created a remote repository, committed our changes to be saved in Git, and finally pushed the repo to GitHub. Now let's finish out the rest of Tic Tac Toe.

The Rest of the Tic Tac Toe Code:

Place the following the code into the same Java source file from above. **¡WARNING!** If you try to copy and paste the code as a single blob you will get the page numbers from the bottom of each page! This will BREAK the compilation. Please give mad props to Blmaster[3] for making this very slick version of Tic-Tac-Toe.

```
public class TicTacToe implements ActionListener {
    final String VERSION = "1.0";
    //Setting up ALL the variables

    JFrame window = new JFrame("Tic-Tac-Toe " + VERSION);

    JMenuBar mnuMain = new JMenuBar();
    JMenuItem mnuNewGame = new JMenuItem("New Game"),
    mnuInstruction = new JMenuItem("Instructions"),
    mnuExit = new JMenuItem("Exit"),
    mnuAbout = new JMenuItem("About");

    JButton btn1v1 = new JButton("Player vs Player"),
    btn1vCPU = new JButton("Player vs CPU"),
    btnBack = new JButton("<--back");
    JButton btnEmpty[] = new JButton[10];

    JPanel pnlNewGame = new JPanel(),
    pnlNorth = new JPanel(),
    pnlSouth = new JPanel(),
    pnlTop = new JPanel(),
    pnlBottom = new JPanel(),
    pnlPlayingField = new JPanel();
    JLabel lblTitle = new JLabel("Tic-Tac-Toe");
    JTextArea txtMessage = new JTextArea();

    final int winCombo[][] = new int[][] {
        {1, 2, 3}, {1, 4, 7}, {1, 5, 9},
        {4, 5, 6}, {2, 5, 8}, {3, 5, 7},
        {7, 8, 9}, {3, 6, 9}
        /*Horizontal Wins*/ /*Vertical Wins*/ /*Diagonal Wins*/
    };
    final int X = 412, Y = 268, color = 190;
    boolean inGame = false;
    boolean win = false;
    boolean btnEmptyClicked = false;
    String message;
    int turn = 1;
    int wonNumber1 = 1, wonNumber2 = 1, wonNumber3 = 1;

    public TicTacToe(){ //Setting game properties and layout and sytle...
        //Setting window properties:
        window.setSize(X, Y);
        window.setLocation(450, 260);
        window.setResizable(false);
        window.setLayout(new BorderLayout());
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //Setting Panel layouts and properties
        pnlNewGame.setLayout(new GridLayout(2, 1, 2, 10));
        pnlNorth.setLayout(new FlowLayout(FlowLayout.CENTER));
        pnlSouth.setLayout(new FlowLayout(FlowLayout.CENTER));

        pnlNorth.setBackground(new Color(color-20, color-20, color-20));
        pnlSouth.setBackground(new Color(color, color, color));

        pnlTop.setBackground(new Color(color, color, color));
        pnlNewGame.setBackground(Color.blue);

        //Adding menu items to menu bar
        mnuMain.add(mnuNewGame);
        mnuMain.add(mnuInstruction);
        mnuMain.add(mnuAbout);
        mnuMain.add(mnuExit);//---->Menu Bar Complete

        //Adding buttons to NewGame panel
        pnlNewGame.add(btn1v1);
        pnlNewGame.add(btn1vCPU);

        //Adding Action Listener to all the Buttons and Menu Items
        mnuNewGame.addActionListener(this);
        mnuExit.addActionListener(this);
        mnuInstruction.addActionListener(this);
        mnuAbout.addActionListener(this);
        btn1v1.addActionListener(this);
        btn1vCPU.addActionListener(this);
        btnBack.addActionListener(this);

        //Setting up the playing field
        pnlPlayingField.setLayout(new GridLayout(3, 3, 2, 2));
        pnlPlayingField.setBackground(Color.black);
        for(int i=1; i<=9; i++) {
            btnEmpty[i] = new JButton();
            btnEmpty[i].setBackground(new Color(220, 220, 220));
            btnEmpty[i].addActionListener(this);
            pnlPlayingField.add(btnEmpty[i]);
        }
        //Adding everything needed to pnlNorth and pnlSouth
        pnlNorth.add(mnuMain);
        pnlSouth.add(lblTitle);

        //Adding to window and Showing window
        window.add(pnlNorth, BorderLayout.NORTH);
        window.add(pnlSouth, BorderLayout.CENTER);
        window.setVisible(true);
    }

    //-------------------START OF ACTION PERFORMED CLASS-------------------------//
    public void actionPerformed(ActionEvent click){
        Object source = click.getSource();
        for(int i=1; i<=9; i++){
            if(source == btnEmpty[i] && turn < 10){
                btnEmptyClicked = true;
                if(!(turn % 2 == 0))
                    btnEmpty[i].setText("X");
                else
                    btnEmpty[i].setText("O");
                btnEmpty[i].setEnabled(false);
                pnlPlayingField.requestFocus();
                turn++;
            }
        }
        if(btnEmptyClicked){
            checkWin();
            btnEmptyClicked = false;
        }
        if(source == mnuNewGame) {
            clearPanelSouth();
            pnlSouth.setLayout(new GridLayout(2, 1, 2, 5));
            pnlTop.add(pnlNewGame);
            pnlBottom.add(btnBack);
            pnlSouth.add(pnlTop);
            pnlSouth.add(pnlBottom);

        }
        else if(source == btn1v1){
            if(inGame){
                int option = JOptionPane.showConfirmDialog(null,
                "If you start a new game," + "your current game will be lost..." +
                "\n" + "Are you sure you want to continue?",
                "Quit Game?" ,JOptionPane.YES_NO_OPTION);
                if(option == JOptionPane.YES_OPTION){
                    inGame = false;
                }
            }
            if(!inGame){
                btnEmpty[wonNumber1].setBackground(new Color(220, 220, 220));
                btnEmpty[wonNumber2].setBackground(new Color(220, 220, 220));
                btnEmpty[wonNumber3].setBackground(new Color(220, 220, 220));
                turn = 1;
                for(int i=1; i<10; i++){
                    btnEmpty[i].setText("");
                    btnEmpty[i].setEnabled(true);
                }
            win = false;
            showGame();

            }
        }
        else if(source == btn1vCPU) {
            JOptionPane.showMessageDialog(null, "Coming Soon!!");
        }
        else if(source == mnuExit){
            int option = JOptionPane.showConfirmDialog(null,
            "Are you sure you want to exit?",
            "Exit Game" ,JOptionPane.YES_NO_OPTION);
            if(option == JOptionPane.YES_OPTION)
                System.exit(0);
        }
        else if(source == mnuInstruction || source == mnuAbout){
            clearPanelSouth();
            String message = "";
            txtMessage.setBackground(new Color(color, color, color));
            if(source == mnuInstruction){
                message = "Instructions:\n\n" +
                "Your goal is to be the first player to get 3 X's or O's in a\n" +
                "row. (horizontally, diagonally, or vertically)";
            } else {
                message = "About:\n\n" +
                "Title: Tic-Tac-Toe\n" +
                "Author: Blmaster\n" +
                "Version: " + VERSION + "\n";
            }
            txtMessage.setEditable(false);
            txtMessage.setText(message);
            pnlSouth.setLayout(new GridLayout(2, 1, 2, 5));
            pnlTop.add(txtMessage);
            pnlBottom.add(btnBack);
            pnlSouth.add(pnlTop);
            pnlSouth.add(pnlBottom);
        }
        else if(source == btnBack){
            if(inGame)
```

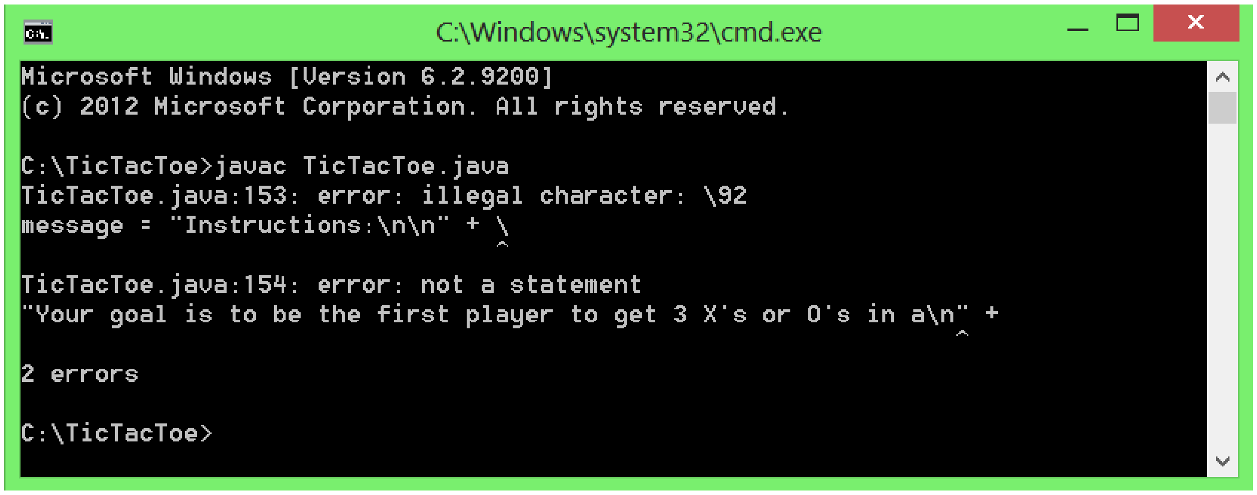
```java
            showGame();
        else {
            clearPanelSouth();
            pnlSouth.setLayout(new FlowLayout(FlowLayout.CENTER));
            pnlNorth.setVisible(true);
            pnlSouth.add(lblTitle);
        }
    }
    pnlSouth.setVisible(false);
    pnlSouth.setVisible(true);
}
//-------------------END OF ACTION PERFORMED CLASS-------------------------//

/*
----------------------------------
Start of all the other methods. |
----------------------------------
*/
public void showGame(){ // Shows the Playing Field
    // *IMPORTANT*- Does not start out brand new
    //             (meaning just shows what it had before)
    clearPanelSouth();
    inGame = true;
    pnlSouth.setLayout(new BorderLayout());
    pnlSouth.add(pnlPlayingField, BorderLayout.CENTER);
    pnlPlayingField.requestFocus();
}

public void checkWin(){
    // checks if there are 3 symbols in a row vertically,
    // diagonally, or horizontally. then shows a message
    // and disables buttons.

    for(int i=0; i<7; i++){
        if(
        !btnEmpty[winCombo[i][0]].getText().equals("") &&
        btnEmpty[winCombo[i][0]].getText().equals(btnEmpty[winCombo[i][1]].getText()) &&
        btnEmpty[winCombo[i][1]].getText().equals(btnEmpty[winCombo[i][2]].getText())
            /*
            The way this checks the if someone won is:
            First: it checks if the btnEmpty[x] is not equal to an empty string
            x being the array number inside the multi-dimensional
            array winCombo[checks inside each of the 7 sets][the first number].

            Second: it checks if btnEmpty[x] is equal to btnEmpty[y]
            x being winCombo[each set][the first number]
            y being winCombo[each set the same as x][the second number]

            (So basically checks if the first and second number in each
            set is equal to each other)

            Third: it checks if btnEmtpy[y] is equal to btnEmpty[z]
            y being the same y as last time
            z being winCombo[each set as y][the third number]
```

```
                Conclusion: So basically it checks if it is equal to
                the btnEmpty is equal to each set of numbers
                */
            ){
                win = true;
                wonNumber1 = winCombo[i][0];
                wonNumber2 = winCombo[i][1];
                wonNumber3 = winCombo[i][2];
                btnEmpty[wonNumber1].setBackground(Color.white);
                btnEmpty[wonNumber2].setBackground(Color.white);
                btnEmpty[wonNumber3].setBackground(Color.white);
                break;
            }
        }
        if(win || (!win && turn>9)){
            if(win){
                if(turn % 2 == 0)
                    message = "X has won!";
                else
                    message = "O has won!";
                win = false;
            } else if(!win && turn>9){
                message = "Both players have tied!\nBetter luck next time.";
            }
            JOptionPane.showMessageDialog(null, message);
            for(int i=1; i<=9; i++){
                btnEmpty[i].setEnabled(false);
            }
        }
    }

    public void clearPanelSouth(){ //Removes all the possible panels
        //that pnlSouth, pnlTop, pnlBottom
        //could have.
        pnlSouth.remove(lblTitle);
        pnlSouth.remove(pnlTop);
        pnlSouth.remove(pnlBottom);
        pnlSouth.remove(pnlPlayingField);
        pnlTop.remove(pnlNewGame);
        pnlTop.remove(txtMessage);
        pnlBottom.remove(btnBack);
    }

    public static void main(String[] args){
        new TicTacToe();// Calling the class construtor.
    }
}
```

The Moment of Truth: Compiling TicTacToe

Finally, we can try to actually compile this Java source with the javac compiler. Simply type: `javac TicTacToe.java`. There is an old adage with coding and compiling: *"No news is good news!"*.

If something like the following figure shows up you are in trouble:



```
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\TicTacToe>javac TicTacToe.java
TicTacToe.java:153: error: illegal character: \92
message = "Instructions:\n\n" + \
                                ^
TicTacToe.java:154: error: not a statement
"Your goal is to be the first player to get 3 X's or O's in a\n" +
                                                              ^
2 errors

C:\TicTacToe>
```

Figure 6: Javac Compile Errors

This shows that there are compiler issues regarding the syntax and improper code. Welcome to the world, and pain, that are syntax errors. Look at the line numbers given and try to determine what went wrong. As good or bad as Alice is you never have to deal with such errors!

Running the Game

I'm sure you REALLY want to stop coding and start playing Tic-Tac-Toe. Let's make sure the game was compiled, thus type: `dir` (`ls` for you Linux/Mac guys). Do you see a TicTacToe.class file? That is the bytecode generated from the source code, which can be executed with the Java Runtime Engine. So fire it up! Type the following command: `java TicTacToe`. If the game was properly compiled you should see this:

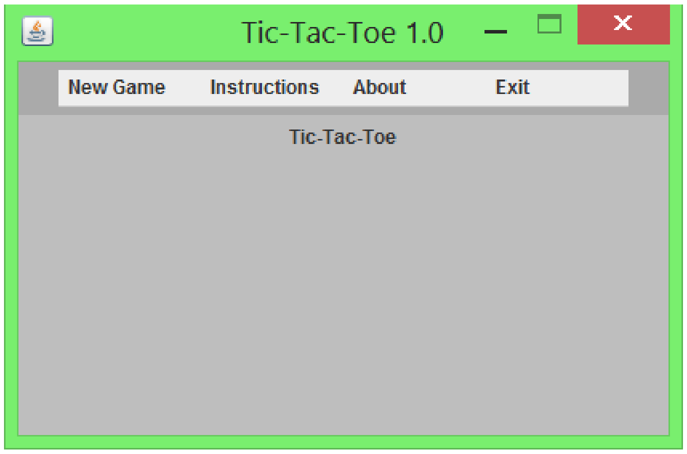
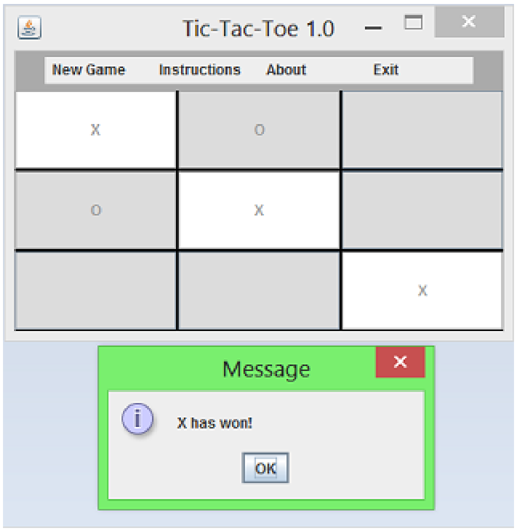


Figure 7: Tic-Tac-Toe First Started

Play a game or ten and see how it executes:



X has won!

Figure 8: Tic-Tac-Toe X Wins!

Let's "Git" Out of Here!

Before the lab completes we need to make sure we back up the files to the local and remote Git repository! Yet again type: `git add *` (verify with `git status`), and then commit the changes: `git commit -m "<more epic message than last time>"`. As a fun distraction see how the commit history looks like with the Git GUI: execute `gitk` on the command line. Notice both commits? The changes in green?
Before finishing the lab send your files to your GitHub repo: `git push GitHub master`. Verify the files were sent to GitHub. If all files have been committed you are set! Congratulations!

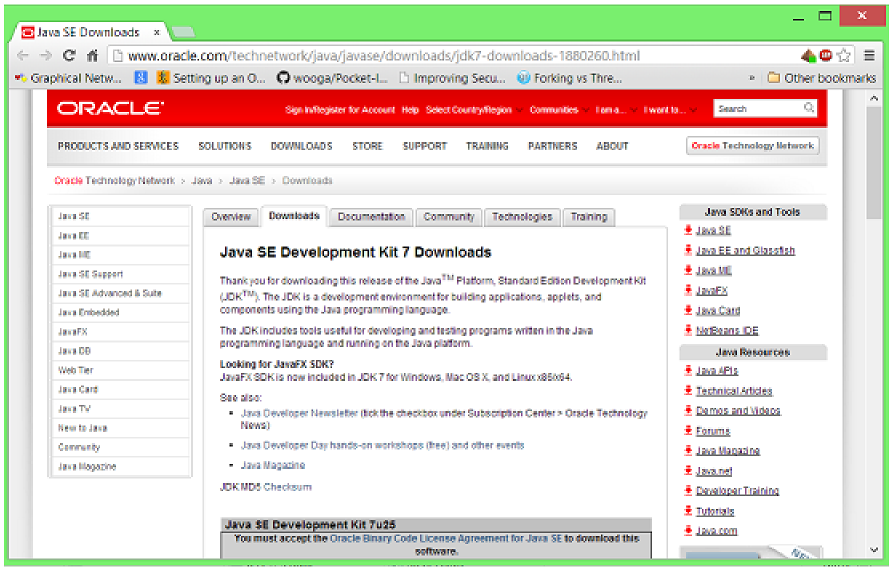
Before you run away...

As some food for thought before you run away screaming please take a moment to review this very brief look at a "game development process". Coding is NOT easy. You probably notices many issues with the installation, configuration, and execution of some, if not all, of the tools. I must state that to get better at coding you need to practice. A lot. Like 10,000 hours. Then you will become an expert. However, you should have seen very simple ways to help you code: 1) Command Line (after proper set up) is lightning fast with javac and git, 2) GitHub makes remote backups and sharing code with friends a snap, and 3) PATH can be greatly leveraged to navigate your computer and start programs on the fly. I hope you learned at least ONE thing, and I greatly encourage you to continue down this track of game development. Have fun and cheers to work well done!

Appendix A: Installing the Java Development Kit (JDK)

The JDK Website

The JDK is available for download from Oracle's website here: Java Development Kit 7. It should look like the following figure:



Java SE Downloads
www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html
PRODUCTS AND SERVICES SOLUTIONS DOWNLOADS STORE SUPPORT TRAINING PARTNERS ABOUT
Oracle Technology Network > Java > Java SE > Downloads
Overview Downloads Documentation Community Technologies Training
Java SDKs and Tools
Java Resources

Java SE Development Kit 7 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

Looking for JavaFX SDK?
JavaFX SDK is now included in JDK 7 for Windows, Mac OS X, and Linux x86/x64.

See also:

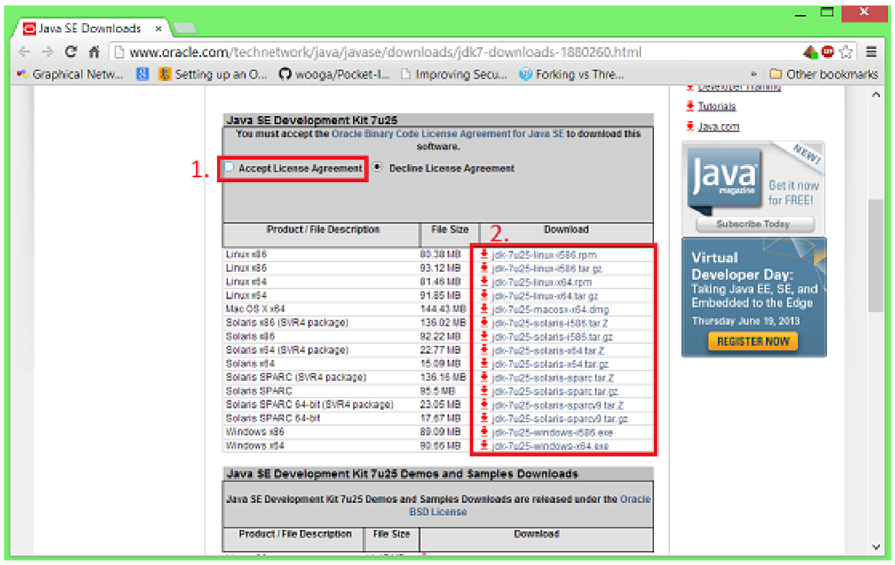
- Java Developer Newsletter (tick the checkbox under Subscription Center > Oracle Technology News)
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK MD5 Checksum

Java SE Development Kit 7u25
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Figure 9: Java Website

Please note that you need to accept the license agreement (1.) before you are able to download the file (2.):



Java SE Development Kit 7u25
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Product / File Description	File Size	Download
Linux x86	80.38 MB	jdk-7u25-linux-i586.rpm
Linux x86	93.12 MB	jdk-7u25-linux-i586.tar.gz
Linux x64	81.46 MB	jdk-7u25-linux-x64.rpm
Linux x64	91.85 MB	jdk-7u25-linux-x64.tar.gz
Mac OS X x64	144.43 MB	jdk-7u25-macosx-x64.dmg
Solaris x86 (SVR4 package)	136.02 MB	jdk-7u25-solaris-i586.tar.Z
Solaris x86	92.22 MB	jdk-7u25-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	22.77 MB	jdk-7u25-solaris-x64.tar.Z
Solaris x64	15.09 MB	jdk-7u25-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	136.16 MB	jdk-7u25-solaris-sparc.tar.Z
Solaris SPARC	95.5 MB	jdk-7u25-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	23.05 MB	jdk-7u25-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	17.67 MB	jdk-7u25-solaris-sparcv9.tar.gz
Windows x86	89.09 MB	jdk-7u25-windows-i586.exe
Windows x64	90.66 MB	jdk-7u25-windows-x64.exe

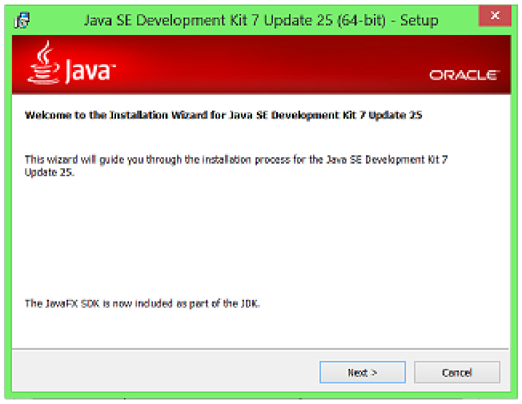
Java SE Development Kit 7u25 Demos and Samples Downloads
Java SE Development Kit 7u25 Demos and Samples Downloads are released under the Oracle BSD License

Product / File Description	File Size	Download

Figure 10: Download Agreement

The JDK Installer

The JDK installer is semi-automated and will ask for occasion user input (install directory, shortcuts, etc). Follow the directions, and leave the default settings.



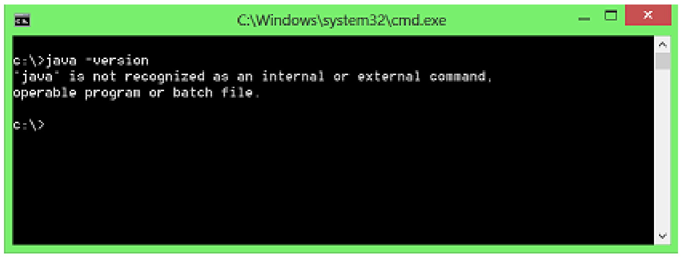
Welcome to the Installation Wizard for Java SE Development Kit 7 Update 25

This wizard will guide you through the installation process for the Java SE Development Kit 7 Update 25.

The JavaFX SDK is now included as part of the JDK.

Figure 11: JDK Installer

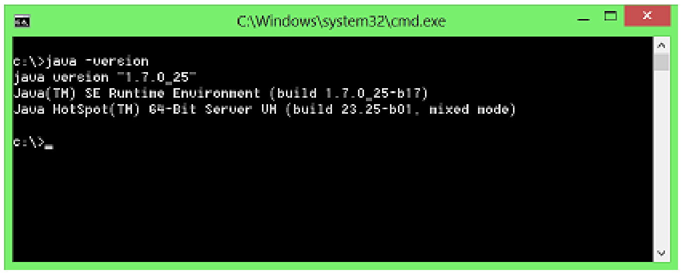
If you try to bring up commandline/terminal and execute `java -version` and see the following error please reinstall:



```
c:\>java -version
'java' is not recognized as an internal or external command,
operable program or batch file.

c:\>
```

Figure 12: JDK Installation Error

Otherwise you should have the "version" information in front of you:



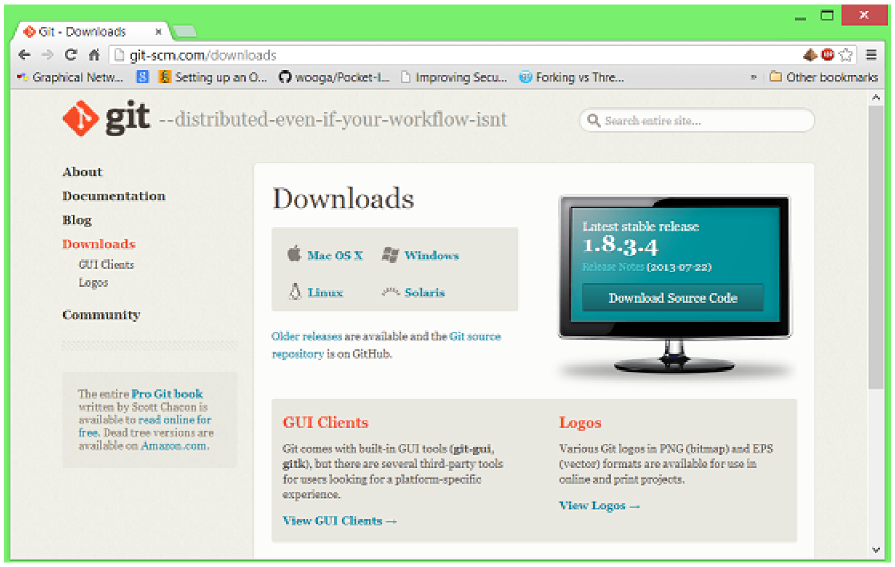
```
c:\>java -version
java version "1.7.0_25"
Java(TM) SE Runtime Environment (build 1.7.0_25-b17)
Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode)

c:\>_
```

Figure 13: JDK Properly Installed

Appendix B: Installing Git

The Git Website

The JDK is available for download from Oracle's website here: Git Version Control System. It should look like the following figure:



Downloads

Latest stable release
1.8.3.4
Release Notes (2013-07-22)

Older releases are available and the Git source repository is on GitHub.

The entire Pro Git book written by Scott Chacon is available to read online for free. Dead tree versions are available on Amazon.com.

GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

View GUI Clients →

Logos

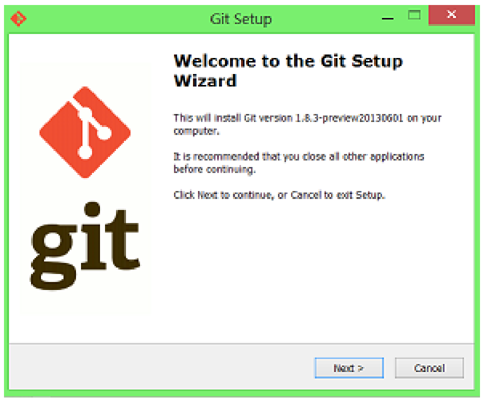
Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

View Logos →

Figure 14: Git Website

The Git Installer

The Git installer is semi-automated and will ask for occasion user input (install directory, shortcuts, etc). Follow the directions, and please note the two figures for specialized options.



Welcome to the Git Setup Wizard

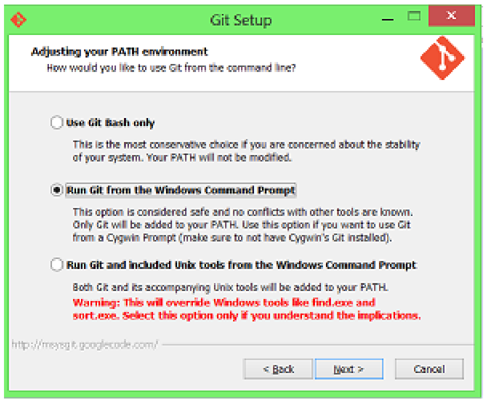
This will install Git version 1.8.3-preview20130601 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

Figure 15: Git Installer

The Git Installer Options

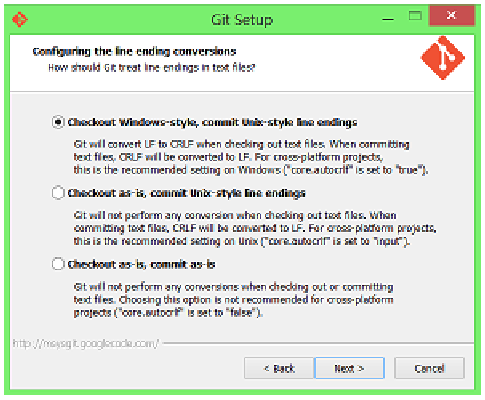


Adjusting your PATH environment
How would you like to use Git from the command line?

- ○ **Use Git Bash only**
  This is the most conservative choice if you are concerned about the stability of your system. Your PATH will not be modified.
- ● **Run Git from the Windows Command Prompt**
  This option is considered safe and no conflicts with other tools are known. Only Git will be added to your PATH. Use this option if you want to use Git from a Cygwin Prompt (make sure to not have Cygwin's Git installed).
- ○ **Run Git and included Unix tools from the Windows Command Prompt**
  Both Git and its accompanying Unix tools will be added to your PATH.
  Warning: This will override Windows tools like find.exe and sort.exe. Select this option only if you understand the implications.

http://msysgit.googlecode.com/

Figure 16: Git Option Setting 1



Configuring the line ending conversions
How should Git treat line endings in text files?

- ● **Checkout Windows-style, commit Unix-style line endings**
  Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").
- ○ **Checkout as-is, commit Unix-style line endings**
  Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").
- ○ **Checkout as-is, commit as-is**
  Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").

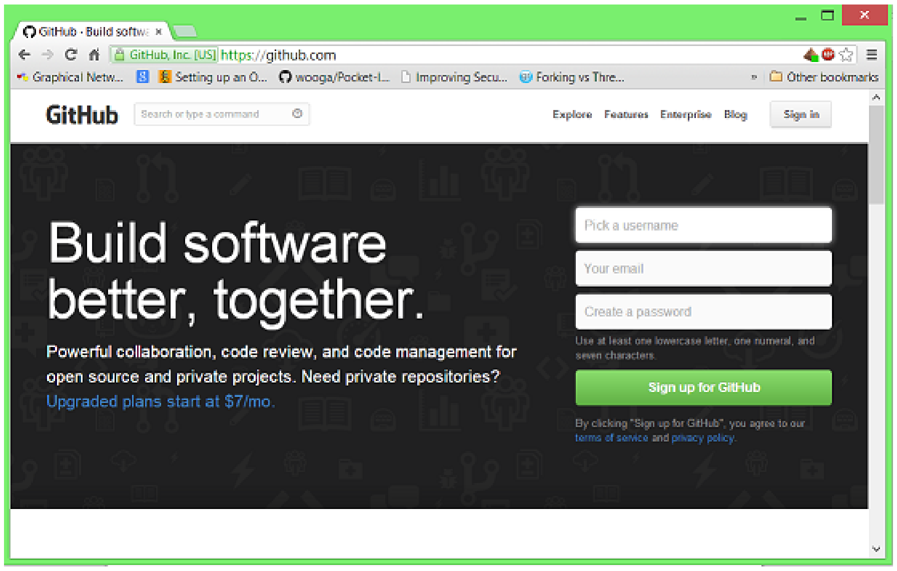
http://msysgit.googlecode.com/

Figure 17: Git Option Setting 2

Appendix C: GitHub

The GitHub Website

GitHub is a free online storage for Git repositories. Please go to the website and sign up for an account



Build software better, together.

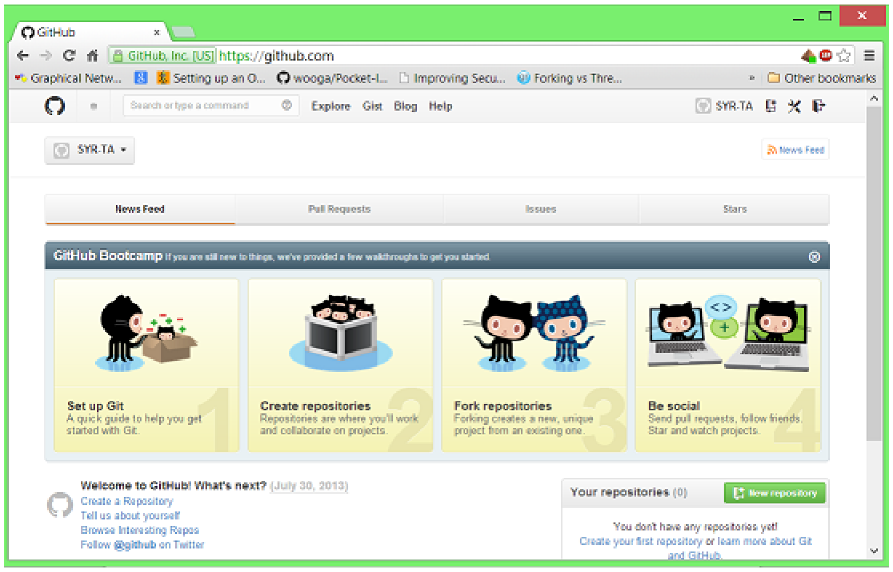
Powerful collaboration, code review, and code management for open source and private projects. Need private repositories? Upgraded plans start at $7/mo.

Use at least one lowercase letter, one numeral, and seven characters.

By clicking "Sign up for GitHub", you agree to our terms of service and privacy policy.

Figure 18: GitHub Registration Page

When you access the website and sign in it should look like the following figure:



GitHub Bootcamp If you are still new to things, we've provided a few walkthroughs to get you started.

Set up Git
A quick guide to help you get started with Git.

Create repositories
Repositories are where you'll work and collaborate on projects.

Fork repositories
Forking creates a new, unique project from an existing one.

Be social
Send pull requests, follow friends. Star and watch projects.

Welcome to GitHub! What's next? (July 30, 2013)
Create a Repository
Tell us about yourself
Browse Interesting Repos
Follow @github on Twitter

Your repositories (0)

You don't have any repositories yet!
Create your first repository or learn more about Git and GitHub.

Figure 19: GitHub when signed in

Appendix D: The PATH Environment Variable

Operating System Environment Variables

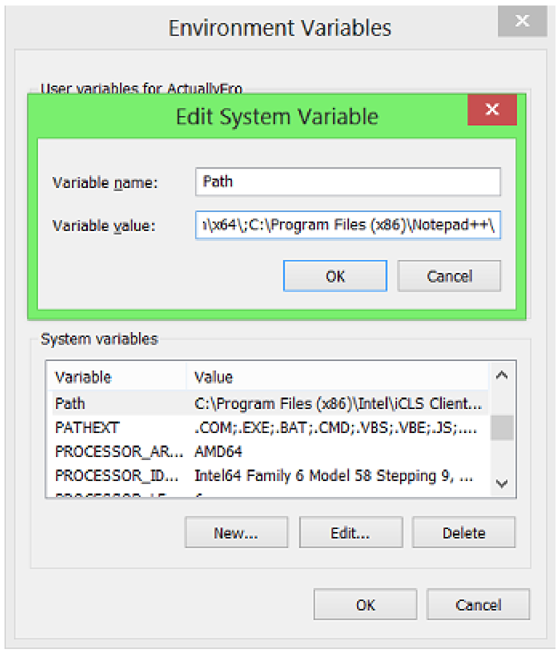
> "Environment variables are a set of dynamic named values that can affect the way running processes will behave on a computer. They are part of the operating environment in which a process runs. They were introduced in their modern form in 1979 with Version 7 Unix, so are included in all Unix operating system flavors and variants from that point onward including Linux and OS X. From PC DOS 2.0 in 1982, all succeeding Microsoft operating systems including Microsoft Windows, and OS/2 also have included them as a feature, although with somewhat different syntax, usage and standard variable names." – Wikipedia 1

The "PATH" Environment Variables

> "PATH is an environment variable on Unix-like operating systems, DOS, OS/2, and Microsoft Windows, specifying a set of directories where executable programs are located. In general, each executing process or user session has its own PATH setting." – Wikipedia 2

Why Use PATH?

The PATH system variable is the FIRST place an Operating System will look for executable files. In turn this means that you can simply add desired programs that can be "directly" excuted. For instance: Notepad++ is installed, by default, to `"C:\Program Files (x86)\Notepad++\notepad++.exe"` on 64 bit Windows. When placed on the PATH you can simply start the program from commandline:



Variable name: Path

Variable value: ı\x64\;C:\Program Files (x86)\Notepad++\

Variable	Value
Path	C:\Program Files (x86)\Intel\iCLS Client...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_AR...	AMD64
PROCESSOR_ID...	Intel64 Family 6 Model 58 Stepping 9, ...

Figure 20: Notepad++ Added to Windows PATH Variable

Appendix E: Storing Java programs in a Jar

Ever seen a Java program get passed around as a .class file? Liar! If you have seen a Java program chances are really good you saw a "Jar" file (.jar extension). One of the most common jar files out there for video games is Minecraft.jar home to the wonderful game "Minecraft".

What is a Jar?

A glorified zip file. Don't believe me? Find a jar file and change the extension from .jar to .zip. Attempt to extract the file. Did it work?

Why would we Jar Java code? That .class file worked!

With this lab we made a simple output of a class file that had no input files/graphics/sounds. Image if you would the THOUSANDS of files needed in a standard video game. Have you ever had to download all of these individual files? I didn't think so!

A quick "How To"

To "jar" a file you haveto use the jar tool with the following format: Make a jar: `jar <flags> <output name>.jar <default classpath> <files to shove in jar ... >`. In our case we only have one file, which is the TicTacToe.class. This file can be very quickly added with: `jar cvfe TicTacToe.jar TicTacToe TicTacToe.class`. Compare and contrast the former and aft commands and see if you can identify each parts.

Why doesn't java work!?

When you placed the file(s) into the jar it changed the way java must interact with the file. In order to start a jar file you must now use the format: `java -jar <name of file>.jar`. This informs java that it must look at the file as a jar.