Assignments

- Assignment #1a
- Assignment #1b
- Assignment #1c
- A Brief Linux Reference Guide
- Assignment #2a
- Assignment #2b
- Assignment #3
- Assignment #4
- Assignment #5
- Assignment #6
- Assignment #7
- Assignment #8a
- Assignment #8b
- Assignment #9
- Assignment #10
- Assignment #11
- Assignment #12
- Assignment #13
- Assignment #xx (not this one)
- Assignment #14

Assignment #1a

misweb registration

Our class uses misweb for the class's course technology. We do not use MyCourses, or any other course technology system. You can use either a desktop computer or a mobile device (with a web browser) to access our class's online resources:

1. In your web browser, go to the main misweb page: misweb.business.msstate.edu

This will take you to the main misweb page, which displays the College of Business (COB) academic areas.

2. In a desktop browser, you will see a list of all of the COB faculty. A mobile browser (with its much smaller screen) will display only the academic areas – on a mobile device, you will need to tap on your teacher's academic area to open a list of faculty in that area.

3. Tap on your teacher's name to go to his or her faculty profile page.

4. In a desktop browser, you will see your teacher's complete misweb page; scroll down to the Courses Taught section. In a mobile browser, tap on the "Courses taught" button; this will open a list of courses that your teacher teaches.

5. Find your class in the list of your teacher's courses taught, and tap on the "online gradebook" button for that class.

6. You will see a login page similar to the one shown to the right. When the login page loads in your browser, you can save a bookmark to that page for fast access to this page in the future.

If you have not previously registered for misweb, you will need to register. Upon trying to log in, you will see a form that has several input fields. Enter any information that is needed to complete all fields. Please try to type correctly!

Click the Continue button. That’s all there is to it! You are now registered for the misweb system. You can now access any of your class's course technology.

If you have any questions about this system, please email:

rodney.pearson@msstate.edu

Checking your misweb online grades

To check your grades, go to the main misweb page, at:

http://misweb.business.msstate.edu

Go to your teacher's faculty profile page. Scroll down to the Courses Taught section. Click on any link for Online Grades.

On the next screen, select your class from the list of Online Gradebooks, enter your net id and password in the textfields, and click the Log in button.

Assuming that you log in correctly, you will see one or more buttons for your class. These buttons (if available) link to:

- your grades
- the private course web site
- an online class calendar
- the class email archive
- the class message board

The buttons page also has a link at the very top that you can use to log out of misweb. For (your own) security reasons, you should always log out rather than just moving to a different web page, since logging out physically terminates your misweb session.

For your homework assignment, you must do the following:

1. Register for misweb.

2. Enter your basic information. You must enter your name and email address; everything else is optional.

3. Check your grades in this class.

4. While you are checking your grades, familiarize yourself with the available buttons, such as Private Course Web Site, EMail Archive, and Class Message Board.

5. Post a message on the Class Message Board under the Topic:

Characteristics of a Good Software Developer

If the Topic has not been created yet, create it. If it has been created, add your message to the Topic. Try to add something meaningful.

Pay attention to what you are doing when you post a message on the Class Message Board. If the Topic already exists, DO NOT CREATE A NEW TOPIC. Instead, add a message to the existing Topic.

To add a message to an existing topic, you must actually read at least one existing message on the topic. When you read a message, you will see a button to Post a New Message.

6. When you access the class message board, you will see an option at the bottom to set your “Message Board notification” level. Select “detail”, so you get an automatic email as soon as anyone posts a message to the class message board. This is a required part of this assignment.

The messages posted on the class message board could be an invaluable tool when you are working on your homework assignments.

You do not need to turn in any type of paper or report for this assignment (or any other assignment in this class). Everything will be graded electronically.

misweb Online Gradebook System mobile access

Student Instructions

A new feature in misweb provides you with convenient access to selected course technology through your mobile web browser (smartphone, laptop, or other mobile device). Once set up, you will be able to access your online grades, course web site, and more with just a few simple taps on your mobile device.

To set your device up for quick access in the future, follow these one-time steps:

1. In your device's web browser, go to the main misweb page: misweb.business.msstate.edu

This will take you to the main misweb page, which displays the College of Business academic areas.

2. Tap on your teacher's academic area to open a list of faculty in that area.

3. Tap on your teacher's name to go to his or her faculty profile page.

4. On your teacher's misweb page, tap on the "Courses taught" button. This will open a list of courses that your teacher teaches.

5. Find your class in the list of courses, and tap on the "online gradebook" button for that class.

6. You will see a login page similar to the one shown to the right. When the login page loads in your browser, you can save a bookmark to that page for fast access in the future. [On an iPhone, iPod Touch, or iPad, tap the middle icon at the bottom of the screen (a rectangle, with an arrow pointing to the right), and select the “Add to Home Screen” option. This will create an icon on your Home Screen that you can use in the future to quickly check your grades in this class.]

On this login page, you can select your class from your teacher's list of online gradebooks, enter your net id, and save that information on your device for future accesses. With that information saved, all you will need to do in future accesses is tap your home screen icon, type in your net password, and tap the Log in button.

You can save your net password on your device as well, although for privacy and security reasons, this is not recommended. If you do save your net password on your device, on future accesses you will simply tap your home screen icon, then the Log in button. Two taps, and you're logged in.

Upon logging in, you will see a concise version of the Online Gradebook System, with larger buttons, larger text, and no superfluous text (as shown to the left).

Caching

Be sure to read this page. It describes a problem that confuses many web users.

If you ever access a web page, and it seems like you are getting an old copy of the page, your web browser may be giving you a copy of that page from its *cache*. In general, a *cache* is a temporary holding place. When you access a web page, your browser can store the page in its cache, on your local hard drive. Then, if you ask for the page again, the browser can give you the copy from its cache rather than going back to the internet for the page. If the page has not changed, the copy from cache is just fine. But if the page HAS changed, you have a problem.

You can normally force the browser to give you a new copy by holding down the shift key, and clicking on the Refresh (or Reload) button.

Past versions of the major browsers had good options that you could use to configure the browser cache settings. The current browsers, however, do not give you many cache configuration options. Always remember to suspect that you might be getting a cached page, and use shift-reload to force your browser to load a new page.

As a web professional, always remember that browsers may be configured differently, so something that works one way on one computer may work a different way on a different computer. Caching is one of those things that can be configured differently, and it can really confuse the inexperienced user.

Assignment #1b

Programming Environment Setup

In assignment #1b, you will download and install several pieces of software. In assignment #1c, you will write your first Java application. Be sure to do assignment #1c!

The software that we will use during the semester will be made available to you on our course web site. This software is free, public domain software. The software is also available for download from various world wide web sites.

1. You will need Sun Microsystem's Java Development Kit (JDK). The JDK is available on our misweb course web site, in both Windows (both 32-bit and 64-bit) and Mac versions. You can accept all default options during this installation. Java may already be installed on your computer, so you may want to delay this step to see if you encounter a problem in step 3. If step 3 (Eclipse installation) works, you already have the JDK installed on your computer.

2. (Steps 2 through 6 are optional. You do not need to install an IDE, but you may if you want to.) Some people like an Integrated Development Environment (IDE), in which they type their programs (probably getting type-ahead hints, color coding, delimiter matching, etc) and run the programs, all within that environment. If you want to do this, you can intall Eclipse. You can download a free copy of Eclipse from http://www.eclipse.org/downloads

Download the "Eclipse IDE for Java Developers". You do not need the "EE" (Enterprise Edition) version.

3. Once downloaded, install Eclipse as you would any other Windows or Mac program. (Eclipse requires that you have a Java environment installed on your computer, which you likely already do. If you don't, you can download the Java Development Kit (JDK) from our private course web site.)

4. Run Eclipse. The first time you run it, it will ask for your desired location for your projects that you will create. You can use the default location, or customize your own location. Just be sure that you know where you are storing your files, because you'll need to find them later on. Every Eclipse project that you create will be in a separate folder, underneath the location that you create in this step.

5. If you see a Welcome window, close it.

6. By default, Eclipse will format blocks of code with the opening curly brace at the right end of a line, such as:

```
if(i==0) {


}
```

Some people prefer that the opening curly brace be on the next line:

```
if(i==0)
{


}
```

If you want to configure this, in Eclipse click on Preferences (in Windows Eclipse, Preferences is under the Window menu option; in Mac Eclipse, it's under the Eclipse menu option), Java, Code Style, Formatter.

Click the "New..." button to create a new Profile. Name your new profile anything you want.

The "Edit Dialog" will open automatically. Under the Braces tab, change every setting to "Next Line".

7. An account will be created for you on the mislab server. Your mislab account name will be based on your Banner net id, such as abc123.

You can telnet to mislab.business.msstate.edu to log onto your mislab account. [Technically, telnet is disabled on the mislab server, because it is considered insecure. An alternate protocol, SSH (Secure Shell), is enabled. These instructions use the generic term telnet, but you must actually use SSH on mislab.] PuTTY, an excellent, free Windows telnet/SSH program, is available on our course web site. Installing PuTTY on your computer should be straightforward (simply copy putty.exe to your desired location). On a Mac computer, you can use the built-in ssh program.

Install PuTTY on your computer. When you open the program (after installation), you will see a screen similar to the one shown to the right. Type mislab.business.msstate.edu in the Host Name textfield.

You may want to do one more thing here, changing the default colors to black text on a white background (the default is white text on a black background). If you do want to do this, click on Colours (on the left), then change the Red, Green, and Blue values for the Foreground and Background settings (for white, change the Red, Green, and Blue values to 255, 255, and 255; for black, change them to 0, 0, and 0).

Click the Save button, which will save mislab.business.msstate.edu as your default Host Name, and save your color scheme. Then click the Open button. You will be prompted to login. Your login id is equal to your Banner net id.

Type your login id, and press Enter. You will be prompted for your mislab password. **Your initial password is equal to the last four digits of your MSU student id number.**

> Telnet note: Most telnet programs will not *echo* your password back onto the screen when you type it. That is, when you type your password, you will not see ANY CHARACTERS from your password displayed on the screen. Carefully type your password, and press *Enter*.

Hopefully, you will successfully log in. If you do, you will see a screen similar to the one shown below.

After you log in, use the *passwd* command to change your password. Your new password should be at least 6 characters long; it should not be based on a dictionary word. You might want to consider using digits and/or special characters as well. The Linux passwd program is very picky about the password that you select!

To change your password, after you telnet to mislab.business.msstate.edu and log in, type:

`passwd`

You will be asked for your current password (to authenticate yourself –- don't want to let just anyone change your password!), and then asked to type your new desired password –- twice.

Use the *chfn* command to specify your full name.

`chfn`

Type *logout* to log out of your mislab account.

`logout`

8. You need to set up your mislab account for web programming. If your user name is *abc123*, your web address will be:

http://mislab.business.msstate.edu/~*abc123*

Throughout the remainder of these instructions, references to *abc123* represent your actual login name.

When you Telnet to mislab and login to your account, you will be in your home directory.

> **public_html**
>
> In general, subdirectories are a great way to organize your files in a meaningful structure. In this class, however, you will put all of your files in your public_html directory. This makes it much easier for your teacher to find your files.

Make sure that you are currently in your home directory (that's where you will be placed automatically when you telnet and log in to your account, so you should be there right now). Look

at the prompt on the command line. Your prompt should look something like:

```
[abc123@mislab abc123]$ _
```

This prompt shows (in order) your login id, the name of the server, and the *current* directory (the _ is the cursor).

Get a directory listing: (type the letter l, not the digit 1).

`ls -l` get a directory listing

The directory listing will show (for example):

```
drwxrwxr-x 10 abc123 abc123 5120 Oct 4 13:41 public_html
```

This directory listing shows several things. First, the first character (d) indicates that this directory entry is a *directory*. If the entry related to a file, a dash would appear as the first character. The next nine characters show the *permissions* that apply to this directory. You need to understand Unix directory and file permissions.

Unix directories and files have three sets of permissions: one set for the owner of the file (sometimes called *user* permissions); a second set for accounts which are in a Unix group (an individual group will be created for you and your teacher; use the middle set of permissions to give your teacher permission to access your files); and a final set, called *other*, or *world*, for everyone else. (The acronym *UGO* is sometimes used to refer to User, Group, and Other.) Look at the directory listing again:

```
drwxrwxr-x 10 abc123 abc123 5120 Oct 4 13:41 public_html
```

The nine characters "rwxrwxr-x" indicate the user, group, and other permissions for this directory. The first three characters relate to the user permissions, the next three to the group permissions, and the final three to other permissions. Each entity (user, group, other) can have any combination of three permissions: read, write, and execute. The directory listing above shows:

user permissions: read, write, execute
group permissions: read, write, execute
other permissions: read, execute

When you create a directory or a file, you need to think about what permissions users will need to that directory or file. In general, you want to give users the minimal permissions necessary to do what they need to do. If users need to read your file, give them read permission. If they need to execute the file, give them execute permission. If they need to write onto your file (high risk! be careful!), give them write permission.

Each set of Unix directory permissions consists of three possible rights; each right has a numeric equivalent:

Read = 4 Write = 2 Execute = 1

The permissions for a file (or directory) are set with the *chmod* command, specifying a 3-digit number which relates to:

digit

#1 -- user permissions

#2 -- group permissions

#3 -- other permissions

Each of the three digits is the sum of the permission rights desired for that entity. If you want to set the permissions on your public_html directory to (these are common permissions for web-based files):

user permissions: read, write, execute

group permissions: read, execute

other permissions: read, execute

The numeric equivalents to these permissions are:

user permissions: 4+2+1 7

group permissions: 4+1 5

other permissions: 4+1 5

Use the chmod command to set the correct desired permissions on your public_html directory:

`chmod 755 public_html` change permissions

Now get a new directory listing:

`ls -l` get a directory listing

The directory listing will show (for example):

```
drwxr-xr-x 10 abc123 abc123 5120 Oct 4 13:41 public_html
```

his shows that public_html is a directory. It also shows the three sets of user/group/other permissions. Finally, it shows the group id (GID) and user id (UID) of the owners of the directory. The GID indicates the mislab group who owns this file (*abc123*); the UID indicates the user who

owns the file (*abc123*).

9. You will need to create an HTML file in your public_html subdirectory before you will actually be able to access your web site from a web browser.

As a quick exercise, you can use nano (a Linux text editor) to create an HTML file.

> Important: Put your HTML files in your public_html directory.

First, be sure to move into your public_html directory.

`cd public_html`

Your prompt should look something like:

`[abc123@mislab public_html]$`

This prompt shows your login id, the name of the server, and the current directory. Now, to create your new file, type:

`nano index.html`

Type the following HTML code:

```
<html>
<body>
Hello world
</body>
</html>
```

Press <Ctrl>-x to exit nano. Answer "y" to "Save modified buffer?", and press <enter> when prompted for the file name.

Get a directory listing.

`ls -l`

Review the permissions on your hello.html file. You may see something like:

`-rw-rw-r-- 10 abc123 abc123 5120 Oct 4 13:41 index.html`

You should now be able to get into your web browser and access your sample web page at (for example):

http://mislab.business.msstate.edu/~*abc123*

10. As soon as possible, a list of all Advanced Languages students will be available at:

http://mislab.business.msstate.edu

As soon as you create your index.html file, be sure to test your page from the Advanced Languages student list, since that is the page from which grading will be performed.

> Step 7 tells you how to install FileZilla on your own computer. If you are using a COBI Lab computer, FileZilla is already installed. You can skip to step 10.
>
> On COBI Lab computers, FileZilla is available in the *Programming Tools* icon, right there on the desktop. Double-click the Programming Tools icon to open it.

11. You will need an FTP (File Transfer Protocol) program to upload your work to your mislab account. If you do not already have such a program, you can download **FileZilla** (Windows) from our private course web site. On a Mac computer, YummyFTP is a very good FTP program. At this time, it costs $4.99.

If you are using a COBI Lab computer, you should store your files on a flash drive.

Be sure to upload your files into your public_html directory on your mislab account.

12. Download the Java API (Application Programmer's Interface) documentation from the course web site and unzip it on your computer.

You should create a bookmark link in your web browser to the JDK online documentation (the provided documentation gives detailed information about the JDK API -- Application Programmer’s Interface -- all of the classes, methods, and properties available in the JDK).

Get into your web browser and select "File", "Open", then “Choose File”. Browse your JDK directory structure to find the file a file named java\API\index.html (there are several of them, read on to make sure you open the correct one).

This page is an alphabetical list of Java classes, properties, and methods.

Save a bookmark to this page, then get used to using this online documentation when you program in Java.

13. The programs and files that are included in section 3 of your course packet are available in the /home/bis3523/resources/coursePacket directory on mislab. If you ever want a copy of a program that we go over in class, you’ll know where to find it.

14. Finally... an unfortunate fact of life for web developers is that the various web browsers are somewhat incompatible. In this class, the standard is set at the current version of Firefox. That is, your assignments and programming exams will be graded with the Firefox browser. It is highly recommended that you test your programs in this browser so that you can be sure that they will

work when they are graded.

15. (Windows only) If you want to use the JDK from the command-line (we will on mislab, and you can on your computer if you want to, but this is totally optional), you need to configure your environment. We will probably not need to do anything from the command line on your computer, because we will probably be able to do everything from within Eclipse, so these remaining steps are totally optional.

Modify the value of your computer's Path environment variable. The Path environment variable tells your operating system locations in which to look for executables (such as javac.exe). For this class, you should add c:\program files\java\jdk1.8.0_20\bin to your Path. You can do this with the following steps:

1. open Control Panel

2. open System

3. click on the Advanced tab

4. click on the Environment Variables button

5. Under System Variables, edit the value of Path

The value of the Path environment variable is a list of directories, separated by semi-colons. Simply add your program files\java\jdk1.8.0_20\bin directory to the end of this value, after a semi-colon. The next time you open a DOS window, the path will be updated.

When you open a new DOS window, you can type:

```
path
```

to see the value of your Path environment variable. Your java\bin directory should be part of that value, and the operating system will look there for executables.

16. Test your Java installation. Get to your command line and type:

```
javac –version
```

This should display the version of the JDK that you just installed. More importantly, it will show that the javac command works. This is the command that you will type in the future to compile your Java programs.

You should see something like:

```
javac 1.8.0_20
```

17. The Java interpreter (JVM) uses an environment variable named Classpath to specify "a list of directories to be searched for .class files". That is, when you try to run a program, if your program

uses any outside .class files, the interpreter will search for those .class files – but it will search only in the directories specified by the value of your Classpath environment variable.

You can check the value of your Classpath environment variable by going to the DOS prompt and typing:

```
set classpath
```

This will display something like:

```
CLASSPATH=C:\Program Files\Java\jre1.8.0_20\lib\ext\QTJava.zip
```

This Classpath specifies only one place that the interpreter should look for .class files – in the one listed location. You probably also want the interpreter to ALWAYS look in the current directory first, then elsewhere. To set this up, you need to edit the value of your Classpath environment variable, using steps similar to those followed to modify the value of your Path environment variable.

In directory parlance, . indicates “the current directory”. You want to insert this in your list before the other specified location, so you should change the value to something like:

```
.;C:\Program Files\Java\jre1.8.0_20\lib\ext\QTJava.zip
```

This tells the interpreter, first, search the current directory for .class files; if you don’t find them there, look in the second location.

Assignment #1c

Hello World

filenames: HelloWorld.java, HelloWorld.class

(Many of the assignments provide instructions for writing your programs in Eclipse. Using Eclipse is not required. You can type your programs with a standard text editor, and compile from the command-line.)

Write a simple text-based application to test your installed Eclipse environment. Write an application that will display the phrase "Hello world, from Eclipse" to your screen.

First, open Eclipse.

1. Using the Eclipse menu, select File, New, and Java Project.

2. This will open the New Java Project window. In the Project name: textfield, enter the name of your project:

```
HelloWorld
```

(case-sensitive, no spaces)

Click the Finish button. This will create your project.

3. The Eclipse toolbar has a number of icons. One of these is a green circle with a white C (and a small plus sign in its top right corner). In the screen capture, it's just below the rap10 in the project path. This is the New Class icon. Click it to create a new class for your project.

4. There are many options available when you create a new class. The first thing we want to do is name the class. In the Name: textfield, type HelloWorld

(case-sensitive, no spaces)

Check one of the checkboxes near the bottom:

```
public static void main(String[] args)
```

This is the standard (and required) method header for your main method. This is the first method that will be called when you run your program. We could type the method header ourselves, but by checking this checkbox, Eclipse will type if for us.

Click the Finish button. Eclipse will create the file HelloWorld.java, and put it in your project's src folder.

5. The screen capture to the right shows your initial project, complete with:

- class header

- main method header

- matching curly braces

6. Type one line of code, inside your main method:

```
System.out.println("Hello world, from Eclipse.\n\n");
```

7. Click the Run icon, the green circle with a white arrow pointing to the right. Your output will appear on the Console, the wide area toward the bottom of the Eclipse window.

Hopefully, your first program will run without errors!

8. Outside Eclipse, find your HelloWorld.class file, in the workspace that you specified. In your "workspace" folder, you will find a separate folder for each project that you create, such as one named HelloWorld. The project folders will contain at least two subfolders: src (which contains .java source code files), and bin (which contains compiled byte code .class files). Look in the bin folder for HelloWorld.class.

Upload HelloWorld.class to your mislab account, into your public_html directory. Telnet to mislab.business.msstate.edu, log in to your account, and run your byte code in the mislab environment.

```
java HelloWorld
```

This demonstrates that your byte code is platform-independent. The same byte code will run in your Windows or Mac environment and in mislab's Linux environment. All you need is an appropriate (platform-dependent) JVM in the environment. The byte code is platform-independent; the JVM is platform-dependent. The program named "java" is the JVM on mislab.

9. Upload your HelloWorld.java file to your mislab account, into your public_html directory. (It will be in your project's src folder on your local computer.)

Compile it.

```
javac HelloWorld.java
```

Run your new byte code.

```
java HelloWorld
```

This demonstrates that your source code is platform-independent. You can create your source code in a Windows, Mac, or Linux environment, then compile it into byte code in a different environment. All you need is an appropriate (platform-dependent) Java compiler in the environment. Your source code is platform-independent; the compiler itself is platform-dependent. "javac" is the compiler on mislab.

A Brief Linux Reference Guide

Working with directories

- `mkdir <name>`
  - make subdirectory with specified name
- `rmdir <name>`
  - remove specified subdirectory
- `cd <name>`
  - move down to specified subdirectory
- `cd ..`
  - move up one directory level
- `ls`
  - get a directory listing
- `ls -l`
  - directory listing, long format (letter l)
- `ls -al`
  - directory listing, including hidden files

Working with files

- `more <filename>`
  - display the contents of the file, a page at a time (press <space> to go to next page, or <b> to back up to the previous page)
- `chmod <permissions> <file>`
  - set the permissions on <file>
- `chown <UID>.<GID> <file>`
  - change the user.group owner of <file>
- `cp <source> .`
  - copy specified file to current directory
- `cp <source> <dest>`
  - copy file <source> to the destination
- `mv <old> <new>`
  - rename the file from <old> to <new>
- `mv <old> <dest>`
  - move the file <old> to the destination
- `rm <file>`
  - remove the specified file
- `rm *`

    - remove all files in current directory
- `head <file>`
    - display the first 10 lines of <file>
- `head -20 <file>`
    - display the first 20 lines of <file>
- `tail <file>`
    - display the last 10 lines of <file>
- `tail -20 <file>`
    - display the last 20 lines of <file>

Administration

- `man <command>`
    - display the manual for <command>
- `passwd`
    - change your account's password
- `chfn`
    - change your account's full name
- `top`
    - display all running processes
- `kill <pid>`
    - kill one of your processes
- `ps -u <user>`
    - display all running processes for <user>

Special files

- `public_html`
    - directory in which all web pages should reside
- `.profile`
    - account setup file (similar to autoexec.bat)
- `.forward`
    - used to forward email to another email address

Assignment #2a

DateTime

filenames: DateTime.java, DateTime.class

Create the text-based application DateTime.java. This program should instantiate a Date object, then use that object's toString() method to display the current date and time to “Standard Out”. Be sure to upload your platform-independent source code to your mislab account public_html directory, and be sure to actually test your program on your mislab account.

1. Use Eclipse or a text editor to create a project named DateTime. (The instructions here assume that you are using Eclipse, but you can certainly type your program using a text editor.)

2. Within your project, create a class named DateTime. This class file should contain:

- a class header

- a main method header

- a statement to instantiate a Date object

Instantiating a Date object (an instance of the Date class) in Java is exactly like instantiating one in Javascript. Use the “new” operator. Since Java is a strongly typed language, you must define your variable first, such as:

```
Date myDate;
myDate=new Date();
```

This defines a variable named myDate, whose data type is Date (it is a Date object). It then uses the “new” operator to instantiate a Date object, and assigns that object to the variable myDate.

- a statement to call your Date object's toString() method to get a string representation of the Date object (and assign that string to a String variable)

```
String dateString;
dateString=myDate.toString();
```

- a statement to display the string

```
System.out.println(dateString);
```

4. You will notice that you have syntax errors – unresolved references – because the Date class is not in a standard package that is automatically available to your program.

Before your class header, import the classes from the java.util package:

```
import java.util.*;
```

5. Run your program. As a side-effect, this will also create your .class file. Hopefully it will display the correct date and time on the Console.

6. Upload your DateTime.java file from this project to your mislab account. The file will be in the src subdirectory, under your project folder in your designated workspace. In your ftp program, drag just DateTime.java from the left (your computer) to your mislab public_html directory.

7. ssh to mislab, log in, move to your public_html directory, compile your program, and run your program to make sure it works.

`cd public_html`

`javac DateTime.java`

`java DateTime`

Assignment #2b

Throughout the semester, be sure to use the filenames that the assignments tell you to use. On mislab, store all files in your public_html directory. Do not create any additional directories. Do everything you can to make the job of grading your work easier!

Home Page

You must correctly set up your account on the mislab server, as explained in Assignment #1b, before you can complete this assignment.

filename: index.html

Create a web page that will be used to grade your remaining assignments. This page will provide links to each of your remaining assignments. Be sure to name your file for this assignment index.html. Also, be sure to store the file in your public_html directory.

1. Create your index.html. index.html should contain several anchor tags, providing a link to each of your assignments for the semester (provide links for assignments 1-14). Also provide links for exams 1, 2, 3, and 5. Finally, create a working email link in your links.html, so a grader can email you easily while looking at your assignments throughout the semester. Your links may be either text-based or image-based.

2. Set your link for assignment #1 to display your HelloWorld.java source code file. To do this, ssh to your mislab account, move to the public_html directory, and copy lister.php to your account:

```
cd public_html
```

```
cp /home/bis3523/resources/code/lister.php .
```

Don't forget the period at the end, which specifies the "destination" of your cp command – the current directory.

Set the href of your anchor tag for assignment #1 to

```
href='lister.php?HelloWorld.java'
```

Follow this procedure any time that you need to display the contents of a file rather than having the browser "render" the file.

3. Set your link for assignment #2 to display your DateTime.java source code file that you will create later in this assignment.

```
href='lister.php?DateTime.java'
```

4. Provide some sort of "Not available yet" page for your assignments 1-14 and exams 1, 2, 3, and 5 so your user doesn't get an error page when following those links.

5. Upload all of your required files to the public_html directory on your mislab account. Test your page, using Firefox, from the Advanced Languages student page.

Be sure to test each of your assignments in the environment in which it will be graded. If your assignment does not work under Firefox, from the Advanced Languages page, it does not work. Period.

FOR EVERY ASSIGNMENT: Be sure to go to mislab.business.msstate.edu, to the BIS 3523 page, and to your page. Your Home Page should load automatically.

Reminder

Don’t forget that each programming assignment is an individual assignment. These are not group projects. Do your own work. Do not look at anyone else’s code without that person’s explicit permission. YOU should type every single character in your programs.

Assignment #3

Text-based Payroll Application

Filenames: Payroll.java, Employee.java, Utility.java

Create a project named Payroll. In this project, we will create two classes – Payroll and Employee – and use a third class that you can download from our course web site. Payroll will contain the logic of our payroll application. The Employee class will be a re-usable class that we can use to represent an Employee object. The Utility class contains several class methods that you will end up using throughout the semester.

1. Create your Employee class. Important note: Since this is not the class that represents the main functionality of our application, it will NOT have a main method. Don't check the checkbox for a main method.

2. Your Employee class needs one instance variable (property):

public double payRate;

3. You need a constructor method so your application program can instantiate an Employee object. Your constructor should receive a double, which is this employee's payrate. The constructor needs to assign that parameter to your instance variable for more persistence. (Remember that the name of a constructor method must match the name of its class: Employee. Also, it is not allowed to specify a "return data type", not even "void").

```
public Employee(double thisPayRate)
{
□□this.payRate=thisPayRate;
}
```

4. You need a method named calculatePay that receives an int (hoursWorked) and returns a double (grossPay). This method should calculate the employee's grossPay (hoursWorked times payRate).

```
public double calculatePay(int hoursWorked)
{
□double grossPay=hoursWorked * this.payRate;
□return grossPay;
}
```

5. Create a class that will contain the main logic of your application. Name this class Payroll.

6. Initially, inside your main method define a hard-coded hoursWorked and payRate:

```
int hoursWorked=40;
double payRate=8.75;
```

7. Inside your main method, instantiate an Employee object, passing payRate to the constructor method. Then call the calculatePay method of that Employee object (it's an instance method), passing hoursWorked as your argument, to calculate the gross pay. Display the gross pay to the screen.

```
Employee employee=new Employee(payRate);
double grossPay=employee.calculatePay(hoursWorked);
System.out.println(grossPay);
```

Now, that's a pretty good start, but there are a number of things that you can now do:

8. Modify the program to accept both hoursWorked and payRate from the keyboard. One of our early course packet DOS-based programs uses a readInt method, something like:

```
int hoursWorked=Utility.readInt();
```

Utility.java is on our course web site. Download a copy of Utility.java. Open it in a text editor.

Back in Eclipse, create a new class named Utility. Delete any automatic code that Eclipse puts in that file, and copy/paste the entire Utility.java code from your text editor into this new class in Eclipse.

9. You will, of course, want to display a prompt before reading the input.

```
System.out.println("Hours worked?");
int hoursWorked=Utility.readInt();
```

10. Use Utility.readDouble to accept the payRate from the keyboard.

11. One of the reasons for defining the Employee class is that it separates the functionality of the Employee object from the functionality of your application program. The person who writes the application program doesn't need to know anything about calculating pay; he or she just needs to call the Employee class's calculatePay method. The person who maintains the Employee class needs to know how to calculate grossPay.

Modify the Employee class's calculatePay method to pay time and a half for overtime. That is, if the employee works over 40 hours, pay 1.5 times the payRate for any hours over 40.

Pretend that you are two different people (this will be easy if you actually believe that you ARE two different people). One of your personas is working on the application program, and your other persona (or one of your other personas) maintains the Employee class. You have just demonstrated that the application programmer doesn't need to know the inner workings of calculating gross pay. The application programmer didn't change his (or her, or its) application program; the other programmer changed the Employee class. The application programmer didn't have to recompile his program. As a matter of fact, the application programmer may not even know that anything has changed. But his program is working!

This demonstrates encapsulation. All of the functionality of an Employee object is encapsulated in the Employee class. It also demonstrates the concept of the class's interface. All the application programmer needs to know is the interface with the Employee class: if there are any public properties, the application programmer may manipulate those directly. If there are any public methods, what do they receive? and what do they return? The application programmer does not need to know the inner workings of those methods.

12. You could add a flat tax rate as a property of the Employee class.

```
private double taxRate=.10;
```

Now, if that is a flat taxRate, applied to all Employee objects evenly, does EACH Employee object need its own taxRate, or is there one taxRate for all Employee objects? Although either would work, it seems wasteful to have a separate copy of the value .10 for each Employee object when one copy would do, so you should change that line to:

```
private static double taxRate=.10;
```

You could write a calculateTax method now. This method would receive a double (grossPay) and return a double (netPay). When you use taxRate in your calculations, use Employee.taxRate since taxRate is a class variable (non-static properties are called instance variables; static properties are called class variables).

If you want to prevent the value of taxRate from being changed during the program, you can declare taxRate as "final". By convention, final variables are written in all upper case characters, using the underscore character to separate "logical words":

```
private static final double TAX_RATE=.10;
```

13. Try to think of some other things that you could add to this example.

14. Upload your .java files from this project (including Utility.java) to your mislab account.

Your link to your assignment #3 (in your index.html) should display your Employee.java file. It's in the src subdirectory of your project. Be sure to test your page from http://mislab.business.msstate.edu to verify that your links work.

Assignment #4

Text-based Trivia Application

Filenames: TextTrivia.java

Write a text-based application which will list today's trivia events to the console. Ask the user for today's month and day (from the keyboard), then read the class's trivia file (two versions available from our course web site). Read the records, and if the month and day from the trivia event match the user's input, display the year and event to the console.

Check for the following data entry errors:

1. invalid month number (should be 1-12)

2. invalid day number (minimum day number is 1; if month==1, 3, 5, 7, 8, 10, or 12,
   maximum day number is 31; if month==4, 6, 9, or 11,
   maximum day number is 30; if month==2, maximum day number is 29)

Display the appropriate trivia events, plus a count of the number of events listed.

The trivia file:

Two versions of the trivia file are available on our private course web site:

1. Trivia.dat contains variable-length, delimited fields (using a tab as the delimiter character). When you read a record from this file, you can break it apart using the String class's split() method.

2. Trivia.fil contains fixed-length fields. You can extract individual fields from this record by using the String class's substring() method.

Assignment #5

There is no assignment #5

Assignment #6

GUI Payroll Application

Filenames: GUIPayroll.java, Employee.java, Utility.java

This assignment re-uses the Employee class that you created in the previous assignment. You may also want to use the padString method from Utility.java for formatting some of your output. Hopefully, this assignment will help demonstrate how you can re-use an existing class (such as your Employee class) to build a system.

Write a Java GUI application which performs payroll calculations, similar to your previous assignment. The layout of your application is up to you, but you can feel free to mimic the screen shown here. The captured screen shown there uses a 5-row, 2-column GridLayout which contains:

- three Label objects

- three TextField objects

- three Button objects

- one TextArea object

The operation of this application should be fairly obvious. Your user is supposed to enter an employee name, hours worked, and a pay rate. When the user clicks the Calculate button, your program should instantiate an Employee object, then use methods of the Employee class to calculate the employee's pay. The TextArea should be cleared, and the employee's information should be shown in the TextArea.

The Clear button clears all three TextFields, as well as the TextArea.

The Quit button terminates the program and closes the window.

1. In Eclipse, create a project named GUIPayroll.

2. Create a class named GUIPayroll. This class should be a subclass of java.awt.Frame. This class will contain your application program logic.

3. The GUIPayroll class should contain a main method.

4. The GUIPayroll class should also contain a constructor method.

```
public GUIPayroll()
{
}
```

5. Your main method should do only one thing: instantiate a GUIPayroll object.

```
new GUIPayroll();
```

6. The GUIPayroll constructor method should:

- set your frame's title

- set your frame's size

- set up your desired layout manager

- instantiate GUI objects and add them to the layout manager

- register your class as the event listener for your buttons

- make your frame visible

Notes:

- To respond to a button click, you need to:

```
import java.awt.event.*;
```

implements ActionListener (in the class header)

override the default actionPerformed method

```
public void actionPerformed(ActionEvent e)
{
}
```

- Your logic that actually responds to button clicks (your event responder logic) goes in the actionPerformed method

- To copy Employee.java from a previous project into this one, right-click or control-click on the old Employee.java file, select Copy, then Paste it into your new project.

Note: Because telnet is a text-based program, you will not be able to telnet to your mislab account and run your GUI application. For grading, your teacher will download your files to his computer, and run it locally. Be sure to upload all required .java files from your project to your mislab account.

Assignment #7

Improved GUI Payroll Application

Filenames: GUIPayroll.java, Employee.java, Utility.java

Modify your GUIPayroll project to handle invalid user input in the (expected) numeric fields. Use a try/catch block when you try to get a primitive numeric representation of your String values from the TextFields.

If you detect invalid input, change the text color in your TextArea to red.

Assignment #8a

Setting up your own tomcat Server on mislab

mislab runs the apache web server software. apache is open-source software, available for free from www.apache.org. Over 60% of the web servers on the internet run the apache web server software.

apache does not support Java servlets by itself. To provide that support, mislab runs the tomcat server software (this is also open-source, free software). For class purposes, each student will run his or her own tomcat server. The following instructions explain how to get your tomcat server up and running.

1. First, check your grades for this class, and look for your Assigned Tomcat Port Numbers. You will have two assigned port numbers: a primary port number, and a secondary port number. Write these numbers down for later use.

2. Next, you need to edit one of your system startup files on your mislab account. “Hidden” files are not shown when you get a normal directory listing, such as with the command “ls -l”. To show “all” files, use the command:

`ls -al`

Telnet to your account, and get a listing of all files in your root directory (ls -al). You should see several files whose filenames begin with a period. These are hidden files; they are not normally shown when you get a directory listing (unless you include the “a” switch, of course). You need to edit the file named .profile.

`pico .profile`

Your .profile file contains commands that you want to have executed as part of your login process. For tomcat programming, you need to set the values of four environment variables. You can do this with the export command.

You should see the following four lines in your .profile:

```
export JAVA_HOME=/usr/local/jdk
export CATALINA_HOME=/usr/local/tomcat
export CATALINA_BASE=/home/bis3523/abc123/tomcat
```

```
export CLASSPATH=.:
/usr/local/tomcat/common/lib/servlet.jar:
/usr/local/jdk/bin/mysql.jar
```

Modify the third of these lines to substitute your netid for abc123. Also make sure that your entire path specification is correct (make sure it is /home/bis3523, for instance).

The commands that you type into your .profile are exactly like commands that you would type at the command line. At the command line, you could not press <Enter> in the middle of a command. Similarly, you cannot split any of these commands across multiple lines.

Be sure to check your .profile for accuracy. Each command must be on a single line.

These environment variables specify (1) the location of mislab's Java installation, (2) mislab's tomcat (catalina) installation, (3) your tomcat directory, and (4) locations (directories or files) to be searched for Java classes.

3. You can create your own aliases for commands in your .profile file. For instance, if you want to type "dir" instead of "ls –l", you can include the following line in your .profile:

```
alias dir="ls –l"
```

For your convenience, an alias to start your tomcat server, one to shutdown your tomcat server, and one to show all running tomcat servers have been created in your .profile file.

```
alias starttom="/usr/local/tomcat/bin/startup.sh"
alias stoptom="/usr/local/tomcat/bin/shutdown.sh"
alias showtom="ps -f -C java"
```

You will be able to type starttom to start your tomcat server, stoptom to shut it down, and showtom to show all running tomcat servers.

4. Press Control-X to exit pico, saving your file.

5. Log out of your account, then log back in. The lines that you just inserted into your .profile file will be processed when you log back in (and every time you log in in the future).

Any time that you edit your .profile, you need to log out, then log back in. The commands in your .profile are processed as part of the login process.

6. Under your root directory, make a subdirectory named tomcat:

`mkdir tomcat`

7. Change to your tomcat subdirectory:

`cd tomcat`

8. Copy all of the tomcat files from mislab's tomcat directory to your own tomcat subdirectory:

`cp -R /usr/local/tomcat/* .`

(The -R switch indicates that all subdirectories should be copied, recursively. The period at the end specifies that the current directory is the destination to which files are to be copied.)

9. Get a directory listing. You should see several subdirectories, including: conf, logs, webapps, and work. Change to your conf subdirectory.

`cd conf`

10. The file server.xml is used to configure your tomcat server. You need to edit this file to specify your assigned port numbers.

`pico server.xml`

(If you do not want to use pico, you could ftp server.xml down to your local computer, edit the file, then ftp it back to your mislab account.)

On line 3, change `port=8085` to use your assigned secondary port number. (You can press Control-C to see the current line number in pico.)

On line 15, change `port=8080` to use your assigned primary port number.

11. Exit pico, saving your edited file.

12. Start your tomcat server:

`starttom`

You should get a message indicating the directories that your server is using. These should correspond to the environment variables that you set in your .profile.

13. Check to see if your tomcat server is running:

`showtom`

```
conf> showtom
UID PID PPID C STIME TTY TIME CMD
rpearson 24611 1 13 11:00 pts/2 00:00:04 /usr/local/jdk/bin/java -Djava.u
```

You should see something like:

(Note: There could be multiple tomcat servers running, since each student will run his or her own server.)

14. Get into your web browser and try to access your server. Go to the following URL:

http://mislab.business.msstate.edu:8080

(Replace the 8080 with your assigned primary port number.)

This should load the index.jsp file that is stored in your tomcat/webapps/ROOT directory. (You may have to try this more that once, since it takes a few moments for your tomcat server to actually get up and running. Its first response to a request may be slow, but subsequent responses will be fast.)

15. You should have the file counter.jsp in your tomcat/webapps/ROOT subdirectory. The .jsp filename extension indicates to the tomcat server that this is a java server page, meaning that it includes jsp code. Given your knowledge of Java, you can probably look at counter.jsp and determine to some extent what is going on in the page. When your tomcat server gets a request for this file, it (the server) translates the .jsp file into an actual java servlet. To see this in action, telnet to your account, and move to tomcat/work/Catalina/localhost/ROOT/org/apach/jsp. You will probably have two files in that directory: index_jsp.java and index_jsp.class. These two files were created when you accessed your index.jsp page.

16. In your browser, go to:

http://mislab.business.msstate.edu:8080/counter.jsp

(Replace the 8080 with your assigned primary port number.)

You should see a page which displays the current date and time, followed by an access counter.

17. Now go back to your telnet session and get a new directory listing. You will see that your tomcat server translated your .jsp file into a .java file (counter_jsp.java), then compiled that into a .class file. Your .class file generated the page that you see in the browser!

18. The first time you access counter.jsp with your browser, there will be a significant delay, because the .jsp file has to be translated into .java, then the .java file has to be compiled into byte code. On subsequent requests for the page, these steps can be skipped, so the desired page will load much more quickly.

Note that any time that you shut down your server, or any time that you edit the .jsp file, Java code generation and compilation will need to occur again. One noticeable effect of this is that the next request for the page will, once again, be slow. Another noticeable effect is that your counter will be reinitialized. This counter will count correctly, as long as you don’t shut down your tomcat server and don’t edit your .jsp file.

19. If you want to do some quick experimentation, you can put your own .html files in your tomcat/webapps/ROOT directory and access them via your own assigned tomcat primary port

number.

20. Now that you know that you can process a .jsp correctly, test a servlet. We want to make sure that you can compile a servlet, and that you can then run it from a web page.

You should have the following two files:

`tomcat/webapps/ROOT/snooperServlet.html`

`tomcat/webapps/ROOT/WEB-INF/classes/SnooperServlet.java`

Edit SnoooperServlet.html and change the default port number of 8080 to your primary tomcat port number (this is in the href attribute of an anchor tag).

Also change the abc123 that you find in that attribute to your net id.

Move to your "classes directory". If you are currently in your home directory, you can use one, or several, cd commands to move to your classes directory, such as:

`cd tomcat/webapps/ROOT/WEB-INF/classes`

or

`cd tomcat`

`cd webapps`

`cd ROOT`

`cd WEB-INF`

`cd classes`

21. Get a directory listing, to be sure that you really have SnooperServlet.java.

`ls -l`

22. Compile SnooperServlet.java.

`javac SnooperServlet.java`

If you get an error message such as:

`-bash: javac: command not found`

then you have an error in the Path environment variable in your .profile. Since your .profile was pre-configured for you, you should not get this error message. If you have accidentally messed up your path, however, you could get this error. Your .profile should contain the following line:

`PATH=$PATH:$HOME/bin:/usr/sbin:/usr/local/jdk/bin`

Since java is in /usr/local/jdk/bin on mislab, this value should point the operating system to the correct file.

If you get an error message such as:

```
SnooperServlet.java:10: package javax.servlet does not exist
```

or

```
SnooperServlet.java:13: cannot resolve symbol
```

then you probably have an error in your classpath (also set in your .profile, but you typed this one).

23. Once you have worked the bugs out of your environment, you will successfully compile SnooperServlet.java. Get a directory listing, and make sure that you have a SnooperServlet.class.

24. In your web browser, go to:

http://mislab.business.msstate.edu:8080/snooperServlet.html

Click the anchor link. If you see a display of a lot of information, formatted in a table, you are doing great!

25. When you are finished, you can shut down your tomcat server, or you can leave it running to wait for requests.

```
stoptom
```

Assignment #8b

DataEntry servlet

Filenames: DataEntry.html, DataEntry.java

Create an HTML form that requests data from the user. Submit that form data to a servlet. For output, display the user's submitted data back to the user. For extra experience, also save the data in a server-based text file.

1. Create your HTML data entry form, and store it in your tomcat/webapps/ROOT directory. You can access it at:

http://mislab.business.msstate.edu:8080/DataEntry.html

(Of course, substitute your primary port number.)

2. Your form tag should refer to your servlet in its action attribute:

```
/servlet/abc123.DataEntry
```

3. In Eclipse, create a new Java project. Name it Servlets. You will not actually run your servlets in Eclipse, so we suggest that you simply use this one project for the rest of your Java programming assignments. Just create new classes as you need them, and keep them all in this project. You will upload them as needed from the project into your mislab account.

4. This project needs some non-standard classes that are not automatically included in an Eclipse project. Download the file servlet.jar from our course web site. This is a java archive file that contains several servlet-related classes.

5. In Eclipse, right-click (or control-click) on your project name, then select Properties. This will open the Properties Window.

6. In the Properties Window, select Java Build Path along the left, then click the Libraries tab.

7. Click the "Add External JARs" button, and add servlet.jar to your project.

8. Create a class named DataEntry.

This class should be a subclass of HTTPServlet, not of java.lang.Object.

Click the Browse button. In the pop-up window, start typing the word httpser. When you see the HTTPServlet class listed, select it as your superclass.

9. Eclipse will generate an import statement:

```
import javax.servlet.http.HttpServlet;
```

We recommend changing this to:

```
import javax.servlet.http.*;
```

10. Add another import, for I/O exception handling:

```
import java.io.*;
```

11. Write a method that will be called automatically to respond to an HTTP get request:

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws IOException
{
}
```

12. Your servlet needs to send back a "doctype" indicator before it sends any other output. You can do this with your response parameter, which has a setContentType() method.

response.setContentType("text/html");

13. You can use your response parameter's getWriter() method to get a PrintWriter object, which you can use to send text back to the browser.

```
PrintWriter out=response.getWriter();
out.println("This is a test.");
```

14. Save your work in Eclipse, then upload your DataEntry.java file to tomcat/webapps/ROOT/WEB-INF/classes.

15. Telnet to your account and move to your classes directory. You need to compile your Java code on mislab, because tomcat does not yet support Java 1.8 byte code. mislab has Java compilers for Java 1.4, 1.5, 1.6, 1.7, and 1.8. In your classes directory, type:

```
javac DataEntry.java
```

16. Move to your tomcat/webapps/ROOT/WEB-INF directory.

For security purposes, your tomcat server will not serve any servlet that is not listed in your web.xml file.

Use pico to edit web.xml. You will see entries in the file for HelloWorldExample and SnooperServlet. You need to put an entry in for your DataEntry servlet.

```
<servlet>
      <servlet-name>DataEntry</servlet-name>
      <servlet-class>DataEntry</servlet-class>
</servlet>
<servlet-mapping>
      <servlet-name>DataEntry</servlet-name>
      <url-pattern>*.DataEntry</url-pattern>
</servlet-mapping>
```

Copy/paste in pico is a bit tricky, but here's how you do it. Move to the first <servlet> container for SnooperServlet. Press Shift-Control-6. At the bottom of the screen, you will see "[Mark Set]".

Arrow-down to highlight the lines you want to copy.

Press Control-k.

This will cut the highlight block, so immediately press Control-u to paste it right back where it was. Now you can Control-u again to paste it again, and have a copy that you can edit. Change every SnooperServlet to DataEntry.

VERY IMPORTANT: Any mistake in this file will prevent your tomcat server from working. If you make any mistake in this file, you will probably get a blank screen response from your tomcat server when you try to access any page.

17. You will need to stop your tomcat server and start it again, so it will process your web.xml file.

`stoptom`

`starttom`

18. Modify your servlet to retrieve the submitted form data from your HTML form. If you have a form field named lastName, for instance, you can retrieve the form data with:

```
String lastName=request.getParameter("lastName");
```

Echo information back to the browser to verify that you can indeed retrieve the submitted form data.

Remember that any time you edit your .java servlet, you must:

- save your work in Eclipse (to generate a new .java file)

- upload the .java file from your workspace to your classes folder

- switch over to your mislab account (telnet) and compile the .java file

- stop your tomcat server

- start your tomcat server

19. If you can get one more step done, you will be in really good shape. Write the submitted data into a server-based file (the file will be in your classes folder).

```
try
{
  FileWriter file=new FileWriter("DataEntry.txt",true);
  file.write(lastName+"\t"+firstName+"\n");
  file.close();
}
catch(Exception e)
{
}
```

20. In your index.html, change the href in the anchor tag for this assignment to request DataEntry.html. Unlike previous assignments, you will not display your Java source code now, because you now have a working web-based assignment. We want the DataEntry assignment to work, not simply display source code.

Assignment #9

Java Source Code Lister

Filenames: ListFiles.java, ListSource.java

Write a servlet which will display a directory listing of all of the .java files in your classes folder (on mislab). When the user clicks one of the listed file names, display the contents of that .java file.

1. In Eclipse, you can use your Servlets project, and simply create two new classes in that project. You have already added servlet.jar to that project, so this will save you a step.

2. Create a new class named ListFiles. It should be a subclass of HttpServlet.

3. You will need two import statement. You need to import

`javax.servlet.http.*`

`java.io.*`

4. You will need to write a `doGet()` method.

5. Your `doGet()` method should set the contentType of your response to text/html.

6. Since you will want to generate some output to the browser, you need to call your response object's `getWriter()` method:

```
PrintWriter out=response.getWriter();
```

7. You can instantiate a File object to get a listing of the files in the current directory. Once you get that File object, you can call its listFiles() method to get an array of file names.

```
File dir=new File(".");
```

```
File[] filesList=dir.listFiles();
```

"." indicates "the current directory".

8. Now use a for loop to step through the elements of the filesList array. Process each element in the array (each element is a File object).

You can determine whether a File object is a file or a directory by calling its `isFile()` method.

You can get a File object's filename by calling its `getName()` method.

Once you have a filename (a String object), you can use its matches() method to see if the filename ends with .java.

```
if(filename.matches(".*\\.java"))
```

If the filename ends in .java, generate a line of HTML that includes an anchor tag. Clicking on the anchor tag should submit a request for your ListSource servlet, passing it the current filename. You want to generate something like:

```
<a href='abc123.ListSource?name=Test.java'>Test</a>
```

9. Update web.xml, in your WEB-INF folder, to provide support for both ListFiles and ListSource.

10. When you get this servlet running, it will display a list of the .java files in your classes folder, displaying them as anchor tags.

11. Now write `ListSource.java`. It should also be a subclass of `HttpServlet`.

12. `ListSource` should set the `contentType` of your response to text/plain (not text/html, because you don't want the browser to try to render this Java source code).

13. Retrieve the submitted file name.

14. Instantiate a FileReader object, then a BufferedReader object.

15. Loop, reading a record from the .java file, and displaying it to the browser (as plain text).

16. In your `index.html`, change the href in the anchor tag for this assignment to request your ListFiles servlet.

Assignment #10

Trivia Servlet

Filenames: Trivia.html, Trivia.java

In this assignment, you will write a servlet that reads records from a server-based trivia text file, and displays selected trivia events on the user's browser.

1. Create an HTML page that provides two radio buttons – the user can get trivia events for:

- the current date, or

- a specified month and day.

Use your HTML experience to provide a good user interface for this.

2. Write a servlet that will:

- retrieve the radio button option

- if the user selected "current date", get the current month and day

- if the user specified a month and day, retrieve those from the form

3. Getting the current date in Java.

You can use the GregorianCalendar class to get the current date and time. GregorianCalendar inherits some useful methods and constants from its direct superclass, Calendar. GregorianCalendar inherits a general-purpose accessor method named get, which receives a primitive int. It also inherits several constants that you can pass to specify what you want to "get". To get the current day of the month, for instance, you can:

GregorianCalendar calendar=new GregorianCalendar();

int day=calendar.get(Calendar.DAY_OF_MONTH);

Make sure that you can get a numeric month and a numeric day for both options before you even think about reading the trivia file.

4. You can copy Trivia.fil into your classes directory:

```
cp /home/bis3523/resources/data/Trivia.fil .
```

Be sure to include the period at the end, which specifies the destination for the cp command.

5. Read the records from the trivia file, and display the events for the selected date. Generate good HTML.

6. In your index.html, change the href in the anchor tag for this assignment to Trivia.html.

Assignment #11

Servlet-based Survey System

Filenames: Survey.html, Survey.java

Create an HTML data entry form for some sort of data collection/survey system. Your survey should contain at least one set of radio buttons, so you can generate some counts and statistics (it's easy to count how many 1s, 2, and 3s people select from radio buttons; it's much more difficult to summarize free-form text).

Write a servlet that retrieves the submitted form data, saves it into a server-based text file, and displays up-to-date survey results (counts, percentages, means, whatever makes sense for your survey).

In your index.html, change the href in the anchor tag for this assignment to Survey.html.

Assignment #12

There is no assignment #12

Assignment #13

JDBC Database Trivia

Filenames: DatabaseTrivia.html, DatabaseTrivia.java

Modify your earlier Trivia assignment to retrieve trivia events from mislab's Trivia mysql table.

Assignment #xx (not this one)

States Database Query System

Filenames: States.html, States.java

Create an HTML form that the user can use to perform queries on the mislab States mysql table. You should provide the following options:

1. Sort order: List states alphabetically (by state name), by population, by area, or by population density. In all cases, provide options to list in either ascending or descending order.

2. Selection options: Provide a textfield for each of the four fields: state name, population, area, and population density. If the user enters any letters into the state name textfield, list states that begin with that string (if the user enters Mi, the list all states that begin with Mi; this search should be case-insensitive). If the user enters a number into any of the other three textfields, list all states that match that value or greater (treat the number as a minimum value).

3. Display your output in a well-formatted table.

Assignment #14

AJAX Customer/Database Query System

Filenames: CustomerList.java, CustomerQuery.java

Write a servlet that lists all customers in the mislab mysql customer table. Your list should include only customer number and customer name, listed in a well-formatted table. Provide at least one additional cell to the right of the customer name in your table – you will display detail information for that customer via AJAX.

Respond to clicking on a table row by submitting an AJAX request for that customer's information (submit a request for your CustomerQuery servlet). Display all information about that customer in the cell to the right of the customer name in your table.

In your index.html, change the href in the anchor tag for this assignment to request your CustomerList servlet.