This project has six objectives:

First, you are introduced to GlassFish. GlassFish is an open source application server that implements the latest JEE specification. This tool is used throughout the course. The NetBeans integrated development environment is introduced and is used to build source code and interact with GlassFish.

Second, you build your first set of distributed systems. These are four small web applications using Servlets and Java Server Pages.

Third, you are introduced to the Android simulator. In this simple project, you will be using the simulator's browser capabilities only.

Fourth, you are introduced to simple mobile device awareness and adapting content to be suitable for either desktop or mobile devices.

Fifth, you are introduced to the MVC pattern if you have not used it before.

And finally, as in all projects this semester, you should reflect on the functional and non-functional characteristics (e.g. security, scalability, failure handling, interoperability) of your solutions. There will be questions on the midterm and final exam concerning these characteristics. You should be able to demonstrate a nuanced comprehension of course content and be able to explain the technical aspects in relation to potential real-world applications.

For each project task, software documentation is required. The software that you write (HTML files, Java files and so on) must contain comments that describe what each significant piece of code is intended to accomplish. Points will be deducted if code is not well documented. Read the documentation-related links provided on the course schedule (for class #1) to understand what is expected.

Be sure to consult the rubric linked from the course schedule for details on grading. For each task below, you must submit screenshots that demonstrate your programs running. These screenshots will aid the grader in evaluating your project.
Task 1

Use the NetBeans Project Name: Project1Task1
You do not need to use an MVC framework for this project.

Write an index.jsp page that asks the user to enter a string of text data, and to make a choice of two hash functions using radio buttons. The hash function choices should be MD5 and SHA-1, with MD5 being the default. When the submit button is pressed a servlet is executed. The servlet must be named ComputeHashes.java. The servlet will compute the appropriate cryptographic hash value from the text transmitted by the browser. You will need to employ the Java crypto API to compute the MD5 or SHA-1 hash of the text. The original text will be echoed back to the browser along with the name of the hash, and the hash value. The hash values sent back to the browser should be displayed in two forms: as hexadecimal text and as base 64 notation. We will discuss the use of such hash values later in the course.

To compute the MD5 and SHA-1 hashes, use these standard java packages:

```java
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
```

To compute the Base64 encoding, use the following package:

```java
import sun.misc.BASE64Encoder;
```

The BASE64Encoder class is an internal non-documentated class. BASE64Encoder objects have a method with the signature String encode(byte[]). It returns a base 64 string encoding of an array of bytes.

To compute the hexadecimal representation of a byte array, use the following code:

```java
// From the web site "Real's How To"
public String getHexString(byte[] b) throws Exception {
    String result = "";
    for (int i=0; i < b.length; i++) {
        result += Integer.toHexString((b[i] & 0xff) + 0x100, 16).substring(1);
    }
    return result;
}
```

Be sure to provide a user friendly and attractive user interface.

If you are unfamiliar with HTML forms, a simple explanation can be found at: http://www.w3schools.com/html/html_forms.asp.

So that you may test your program, here are example hashes.
Hashes of the string "Hello":

- SHA-1 (Hex): F7FF9E8BB2E09B70935A5D785E0CC5D9D0ABF0
- SHA-1 (Base 64): 9/+ei3uy4Jtwk1pdeF4MxdnQq/A=
- MD5 (Hex): 8B1A9953C4611296A827ABF8C47804D7
- MD5 (Base 64): ixqZU8RhEpaoJ6v4xHgE1w==

Task 2
Use the NetBeans Project Name: Project1Task2
You do not need to use an MVC framework for this project.

Later in the semester when we are studying the RSA algorithm, it will be useful to be able to do math operations on some arbitrarily large integers. Therefore Task 2 is to create a useful calculator app for this purpose.

Write a simple web application that allows a user to perform one of six operations on two, possibly very large, integers (x, y). The operations will include:

1. Addition (x+y)
2. Multiplication (x*y)
3. An operation to determine if x and y are relatively prime
4. Modulo (x mod y)
5. A modular inverse (x⁻¹ mod y)
6. Raise x to the power of y (i.e. x^y)

A JSP page will present three input fields to the user. The first two will be used to collect the two integers, x and y. The third will be used to collect the operation type. The operations supported will be "add", "multiply", "relativelyPrime", "mod", "modInverse", and "power". Use drop down boxes in XHTML. A submit button will be provided and when it is hit a servlet will be visited. The servlet will be named BigCalc.java and will use the BigInteger class to perform the conversions from strings and the appropriate computation. The servlet will return the result to the browser marked up in HTML. You need to validate both integers and the operation. In the case of invalid input return an error message to the browser - but don't crash the server.

The BigInteger class has multiply, add, modInverse, mod, and pow methods to use. For the operation that determines if the two integers are relatively prime use the gcd() method of the BigInteger class. If the greatest common divisor of the two integers is one then the two integers are relatively prime. And finally, note that the exponent for the pow method takes an integer, not a BigInteger.

Be sure to provide a user friendly and attractive user interface.
Task 3

Use the NetBeans Project Name: Project1Task3
You do not need to use an MVC framework for this project.

Write another web application using NetBeans. This application will determine if a string entered into a browser is a palindrome. A string is a palindrome if it is empty, has a single character, or reads the same when reading from left to right or from right to left. Name your servlet Palin.java. Use an appropriate doctype for an Android mobile.

Download and install the Android simulator from Google. Use the browser on the simulator to visit this web application.

Produce a screen shot showing the simulator working on your web application.

Note:

- You will not be able to connect to the servlet from the Android simulator using the IP address of “localhost” because "localhost" will refer to the Android device itself (not your laptop). Android provides a loopback address of 10.0.2.2 that refers to the system that the simulator is running on.
- For judging a palindrome, only consider letters. Disregard case, punctuation and white space. For example "Madam I'm Adam" is a palindrome.

Task 4

Use NetBeans Project Name: Project1Task4
You **MUST** use an MVC framework for this project.

For task 4, you will build an application that finds the user's Anime doppelganger. Their doppelganger is someone who looks like they could be their twin. We will use images from the Anime Characters Database (http://animecharactersdatabase.com/). This site allows for searching by hair and eye color, hair length, and apparent age. We can ask the user for their hair, eye, and age characteristics and then search the Anime Characters Database for an Anime character that has those same characteristics; that is, finding their Anime doppelganger!

The Anime Characters Database does not have a public API, so we will therefore have to use screen scraping to get the information we need. Screen scraping will be demonstrated in class on January 21/22 and it essentially involves reading, searching,
and snipping from the HTML the information we need. This is a handy technique to use with older and legacy systems from which we can't get structured data.

Your app should start with a welcome page similar to the screenshot on the right. You can use my HTML form and styling if you like.

Select form items should be used to provide the user with the list of eye, hair and age options.

Once the user makes their selections and hits "Find my doppelganger" then an Anime character should be found from the Anime Characters Database and their name and picture should be displayed. There are typically several characters that meet the characteristics, so your program should randomly choose one each time the user selects that category.

At right is an example with blue eyes, brown hair to their ears, and an adult.

Notice that your results page should also allow the user to select characteristics again.

Your application should adapt for mobile access also, using different sized images. That is, you should use the Anime Characters Database thumbnail image for mobile. This is the image that is shown when the set of matches are shown such as at http://tinyurl.com/95702p1t4.

I found that you can get more pictures, up to 100, by adding mymax=100 to the query string.

When you click on one of the images, it takes you to a page with the full-sized image that should be used for the desktop version.
Make sure you have an appropriate DOCTYPE string (for desktop/laptop or mobile) defined as the first element of your HTML replies.

As mentioned earlier, the Anime Character Database does not provide an API to retrieve this data in a structured form. Rather you should screen-scrape the site to extract the information you are looking for. You should use text-based screen-scraping instead of parsing the site into a DOM because (a) text-based screen scraping is more general for cases where the HTML might not be legal XML and therefore would not parse correctly, and (b) we will be doing XML parsing into a DOM in a later assignment.

Finally, you MUST include the link to the Anime Characters Database on every screen (as shown in my examples).

Exceptions:

- If a doppelganger is not found, then your app should fail gracefully. In other words, the response to the user should be that no image for that category was found and allow them to search again. See the example at right. To test for this case, try the characteristics: eye color pink, hair color orange, hair length past waist, and age child.

You should submit screenshots demonstrating your application working.

- Include screenshots showing the welcome page and results pages with 3 very different characters (i.e. very different characteristics).
- Include screenshots showing 2 different characters with the same characteristics
- Include a screenshot when a character cannot be found.
- Include screenshots showing the welcome page and 2 characters on mobile

A working version of the application will be demonstrated in class on January 21/22, and example code to base your solution on will be distributed in time for the lab that day in class.
Questions:

Questions should be posted to the Blackboard Discussion Board, under the Project 1 Discussion Forum.

Summary:

There should be four projects in Netbeans.

The Netbeans projects will be named as follows:

- Project1Task1
- Project1Task2
- Project1Task3
- Project1Task4

You should also have four screenshots folders:

- Project1Task1 Screenshots
- Project1Task2 Screenshots
- Project1Task3 Screenshots
- Project1Task4 Screenshots

Copy all your NetBeans project folders and screenshot folders into a folder named with your andrew id.

Zip that folder, and submit it to Blackboard.

The submission should be a single zip file.