# CSCE 155 - Java

### Lab 11.0 - Objects

## Prior to Lab

Before attending this lab:

1. Read and familiarize yourself with this handout.
2. Review the following free textbook resources:
   - http://docs.oracle.com/javase/tutorial/java/javaOO/index.html
3. For additional Information on some advanced topics:
   - RSS: http://en.wikipedia.org/wiki/RSS
   - HTTP Protocol: http://en.wikipedia.org/wiki/HTTP

## Peer Programming Pair-Up

To encourage collaboration and a team environment, labs will be structured in a *pair programming* setup. At the start of each lab, you will be randomly paired up with another student (conflicts such as absences will be dealt with by the lab instructor). One of you will be designated the *driver* and the other the *navigator*.

The navigator will be responsible for reading the instructions and telling the driver what to do next. The driver will be in charge of the keyboard and workstation. Both driver and navigator are responsible for suggesting fixes and solutions together. Neither the navigator nor the driver is "in charge." Beyond your immediate pairing, you are encouraged to help and interact and with other pairs in the lab.

Each week you should alternate: if you were a driver last week, be a navigator next, etc. Resolve any issues (you were both drivers last week) within your pair. Ask the lab instructor to resolve issues only when you cannot come to a consensus.

Because of the peer programming setup of labs, it is absolutely essential that you complete any pre-lab activities and familiarize yourself with the handouts prior to coming to lab. Failure to do so will negatively impact your ability to collaborate and work with others which may mean that you will not be able to complete the lab.

# 1 Lab Objectives & Topics

At the end of this lab you should be familiar with the following

- Be familiar with the concepts of encapsulation & modularity
- Understand how to design, declare, and use Java classes
- Have some exposure to advanced topics such as sockets, the HTTP protocol, and XML processing

# 2 Background

An RSS feed (RDF Site Summary or "Really Simple Syndication") is a format used to publish frequently updated works. RSS enabled clients can subscribe to RSS feeds and update a user as to new or relevant news items. RSS feeds are most commonly formatted using XML (Extensible Markup Language) that use XML tags to indicate what the data represents (the title of the article, a short description, etc.). Clients "read" an RSS feed by making a connection to a server using the HyperText Transfer Protocol (HTTP).

For example, UNL has an RSS news feed available at http://newsroom.unl.edu/releases/?format=xml which serves XML data that looks something like the following:

```
<rss xmlns:media="http://search.yahoo.com/mrss/"
 ↪  xmlns:atom="http://www.w3.org/2005/Atom" version="2.0">
<channel>
  <title>UNL News Releases</title>
  <link>http://newsroom.unl.edu/releases/</link>
  <description>News from the University of
   ↪  Nebraska-Lincoln</description>
  <language>en-us</language>
  <copyright>Copyright 2012 University of
   ↪  Nebraska-Lincoln</copyright>
  <image>
    <title>UNL News Releases</title>
    <url>http://www.unl.edu/favicon.ico</url>
    <link>http://www.unl.edu/</link>
```

```
12    </image>
13    <item>
14      <title>Guerrilla Girls on Tour perform 'Feminists are Funny'
        ↪  Monday at Sheldon</title>
15
        ↪  <link>http://newsroom.unl.edu/releases/2012/03/09/Guerrilla</link>
16      <description>The Guerrilla Girls on Tour, an internationally
        ↪  acclaimed anonymous theater collective, will perform
        ↪  "Feminists are Funny" at the University of
        ↪  Nebraska-Lincoln's Sheldon Museum of Art, 12th and R
        ↪  streets, at 7 p.m. March 12. The 70-minute play is an...
17      </description>
18      <pubDate>Fri, 09 Mar 2012 02:00:00 -0600</pubDate>
19    </item>
20    ...
21    </items>
22  </channel>
```

## Classes in Java

An entity may be composed of several different pieces of data. A person for example may have a first and last name (strings), an age (integer), a birthdate (some date/time representation), etc. It is much easier and more natural to group each of these pieces of data into one entity. This is a concept known as *encapsulation*–a mechanism by which data can be grouped together to define an entity.

The Java programming language provides a mechanism to achieve encapsulation using classes. In Java, everything (except for primitive types) is a class. Many classes are provided by the SDK and users can define their own classes to model entities. Classes can have one or more data fields–variables which have a type and a name as well as member methods.

Instances of Java classes can be instantiated using a *constructor*. A constructor is a special method which has the same name as the class and which can be accessed using the new operator. You can define any number of constructors (that take different number and types of parameters). If you do not provide a user-defined constructor, Java provides a default, no-argument constructor that can be used.

Once you have an instance, you can access member variables and methods using the dot operator. For example:

```
1  MyObject myInstance = new MyObject();
2  myInstance.aPublicIntegerVariable = 10;
```

```
3  myInstance.executeSomeMethod();
```

It is generally bad practice (poor use of encapsulation) to define a class with public member variables. Instead, an instance's member variable values should be accessed through accessor and mutator methods (getters and setters) which can be conveniently generated for you by Eclipse.

### RSS Client Background

You have been provided with an incomplete RSS client written in Java. The client works as follows: it uses Java's URL class to open a connection to the given URL and obtain its raw RSS XML data. The server responds with a stream of data that the URL class reads into a buffer. This data stream can, in general, be any type of data, but we're expecting an RSS feed–a stream of plain text XML-formatted data conforming to the RSS standard. The data is placed into a `Document` object which parses the XML and provides an interface to access Nodes (XML elements) and attributes.

## 3 Activities

Clone the project code for this lab from GitHub by using the following URL: https://github.com/cbourke/CSCE155-Java-Lab11.

### 3.1 A Student Class

This activity will familiarize you with a completed program in which a Java class has been created to represent a student. A couple of constructors have been provided as well as getters and setters that offer several ways to build a `Student` object. The standard `toString()` method has also been implemented that creates a human-readable string representation of the object. The conversion from a string representing a birthdate to a Java 8 `LocalDate` object is also handled internally (encapsulated in the object).

#### Instructions

1. Examine the syntax of the `Student` class and understand how it works.
2. Change the values in the `main` method to your name, NUID, and birth date.
3. Compile and run the program. Refer back to this program in Activity 2 as needed.

## 3.2 Completing the RSS Client

In this activity, you will complete the RSS Client that connects to a UNL RSS feed, processes the XML data and outputs the results to the standard output. Most of the client has been completed for you. You just need to complete the design and implementation of a Java class that models the essential parts of an RSS item. Your structure will need to support an RSS item's title, link, description, and publication date.

### Instructions

1. Open and examine `RssReader.java` and `Rss.java`
2. In `RssReader.java`, set the `DEFAULT_URL` value to the URL for the RSS feed that you want to pull from (or add and use your own URL if you prefer).
3. Design and implement the RSS class, `Rss.java`
   - Define the class's state (its member variables)
   - Define the class's constructors
   - Define the class's methods (getters, setters, `toString`, etc.) as necessary
4. Complete the functionality in the `RssReader.java` class as follows.
   - Use your new `Rss` class in the `getNewsFeeds` method
   - In the main method, add appropriate code to print out the RSS feed in a readable manner (the formatting details are up to you).

# 4 Handin/Grader Instructions

1. Hand in your completed files:
   - `Statistics.java`
   - `worksheet.md`

   through the webhandin (https://cse-apps.unl.edu/handin) using your cse login and password.
2. Even if you worked with a partner, you *both* should turn in all files.
3. Verify your program by grading yourself through the webgrader (https://cse.unl.edu/~cse155h/grade/) using the same credentials.
4. Recall that both expected output and your program's output will be displayed. The formatting may differ slightly which is fine. As long as your program successfully

compiles, runs and outputs the *same values*, it is considered correct.

# 5 Advanced Activity (Optional)

Many RSS feeds include escaped HTML characters that start with an ampersand (for example: to display a less-than sign, `<` is used. To display the ampersand itself, `&` is used). This is necessary to avoid interpreting these characters as part of the HTML markup. However, since we are printing it in a human readable format, it would be better to reformat these characters as the literal characters that they represent. Write additional code to do this. A full list of HTML character encodings can be found here: http://www.w3schools.com/tags/ref_entities.asp