# Lecture 4

## Associations

- Package: `structure`
- Class: `Association`
- Program: `Courses.java`
- Next Steps

# Associations

In this lecture, we'll show how to implement and use a simple data structure called an *association*. An *association* has two parts: key and value.

The key is used to lookup the associated value in the (key, value) pair. Typically, associations allow the value to be changed, but not the key. In other words, the value can be reassigned.

In Python, you may have seen associations as part of the built-in dictionary data structure.

```
>>> d = dict()
>>> d["key"] = "value"
>>> d["key"]
'value'
>>> d[5] = "hello"
>>> d[6] = "hello"
>>> d[6] = "good-bye"
```

Creating and modifying a dictionary in the Python interactive shell (repl).

```
>>> d
{'key': 'value', 5: 'hello', 6: 'good-bye'}
>>> d.keys()
dict_keys(['key', 5, 6])
>>> d.values()
dict_values(['value', 'hello', 'good-bye'])
```

Examining the contents of dictionary's keys and values.

More broadly, we'll see how Duane's `Association` class fits into his `structure` package. We'll also write a sample program that uses the `Association` class.

# Package: `structure`

# Installation

# Installation

The `structure` package contains implementations of many useful data structures.

The package has an associated git repository. We'll clone it in our `~/cs136` folders.

```
ssh://lohani.cs.williams.edu/~bailey/js.git
```

Location of the `structure` git repository.

It uses the `ssh` protocol (instead of `https`) but we can clone it in the same way.

Note: If you clone from a personal computer, change `lohani` to `username@lohani` with your own CS credentials.

In our Java programs, we will use `import structure` (or `import structure5`) to use it.

However, we need to tell Java where to find the structure package.

- Full instructions can be found in `structure`'s `INSTALL.txt` file.

# Class Path

Java uses an environment variable `$CLASSPATH` to keep track of where packages are stored on your local system. The variable is a list of directories separated by the `:` symbol.

- The `echo` command allows you to check the value of an environment variable.
- Environment variables can be set using the = sign.

To use the `structure` package, we'll want to add its location (i.e., `~/cs136/js`) to this list.

```
[-> echo $CLASSPATH
.:/home/faculty/aaron/java:/usr/lib/java:/usr/cs-local/lib/classes
[-> CLASSPATH=$CLASSPATH:/home/faculty/aaron/cs136/js
[-> echo $CLASSPATH
.:/home/faculty/aaron/java:/usr/lib/java:/usr/cs-local/lib/classes:/home/faculty/aaron/cs136/js
```

Adding `~/cs136/js` to the CLASSPATH variable.
Remember that the list is separated by : symbols.
This modification adds the structure folder to the end fo the list.

Note: Environment variables are cleared or reset whenever you log out of the terminal window.
It would be nice to avoid typing this every time we log in!

# Bash Profile

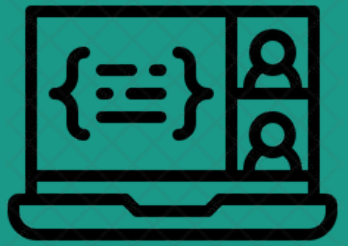
When you log into a `bash` shell (or `zsh` shell), certain scripts are run (e.g., `~/.bash_profile`). We'll modify our `.bash_profile` script to ensure that `$CLASSPATH` always includes the `structure` directory.

```
-> cat .local_bash_profile
export CLASSPATH=.:$HOME/cs136/js/bailey.jar:$CLASSPATH
```

Create a `.local_bash_profile` file in the home folder of your linux account.
Add the line `export CLASSPATH=.:$HOME/cs136/js/bailey.jar:$CLASSPATH` and save it.
This file is run (also known as being *sourced*) after your `.bash_profile` is run.

Note: In our department's linux environment, the `.bash_profile` cannot be edited. Instead, we'll create `.local_bash_profile` which is also run when logging in.

# Live Coding: Installing and Investigating `structure`



Let's use the previous steps to install `structure` (and `structure5`).

Let's also investigate environment variables in more detail.

Then we'll take our first look at what is contained in `structure`.

```
Last login: Fri Sep 17 16:01:30 2021 from 137.165.120.103
[-> echo "the echo command prints messages"
the echo command prints messages
[-> echo "it also expands environment variables"
it also expands environment variables
[-> echo $HOME
/home/faculty/aaron
[->
[-> echo "we can create environment variables"
we can create environment variables
[-> echo $LAB

[-> LAB=$HOME/cs136/lab0
[-> echo $LAB
/home/faculty/aaron/cs136/lab0
[-> pwd
/home/faculty/aaron
[-> cd $LAB
[-> pwd
/home/faculty/aaron/cs136/lab0
```

Initially, the `LAB` variable is empty.
We can assign it a value using `LAB=`.
The value of the variable is `$LAB`.

- Using the `echo` command to print messages and expand environment variables.
- Setting an environment variable `LAB` and using its value `$LAB`.

```
-> cd ~/cs136
-> git clone ssh://lohani.cs.williams.edu/~bailey/js.git
Cloning into 'js'...
aaron@lohani.cs.williams.edu's password:
remote: Enumerating objects: 389, done.
remote: Counting objects: 100% (389/389), done.
remote: Compressing objects: 100% (114/114), done.
remote: Total 389 (delta 274), reused 384 (delta 272)
Receiving objects: 100% (389/389), 2.18 MiB | 25.35 MiB/s, done.
Resolving deltas: 100% (274/274), done.
-> ls
js/  lab0/
```

Remember that steps like these must be done **once in the Unix environment**, and **on every Mac computer** you use.

Installing the `structures` package using `git`. It will appear in the `js` folder.
Make sure that you are in your `cs136` folder when you clone the repository.
(If you clone it in another location, then you can move the `js` folder using the `mv` command.

Source code is in `js/structure5/src/` and compiled code is in `js/bailey.jar`.
For full installation instructions, check out the `js/INSTALL.txt` file.

```
-> cd ~
-> nano .local_bash_profile
```

```
  GNU nano 4.8
export CLASSPATH=.:$HOME/cs136/js/bailey.jar:$CLASSPATH
```

This will put the current folder `.` first, followed by the `structure` package's `bailey.jar` file, followed by the current contents of `CLASSPATH`. The `:` are separators.

```
-> cat .local_bash_profile
export CLASSPATH=.:$HOME/cs136/js/bailey.jar:$CLASSPATH
->
```

Linux (shown above):

- Create a `.local_bash_profile` file in your home folder with the following line: `export CLASSPATH=.:$HOME/cs136/js/bailey.jar:$CLASSPATH`.

Mac:

- Edit the `.bash_profile` file in your home folder and add the same line at the bottom: `export CLASSPATH=.:$HOME/cs136/js/bailey.jar:$CLASSPATH`.

This step makes sure that the CLASSPATH variable is modified every time you login.

```
-> cat Check.java
import structure5.*;
public class Check {
        public static void main(String[] args) {
                Vector<String> v = new Vector<>();
                System.out.println("It works.");
        }
}
->
```

The sample program from `js/INSTALL.txt` needs to be modified slightly before it can be run.

This is the sample program from the `js/INSTALL.txt` file.
We'll name it `Check.java`. A copy will be added to the course website.

```
-> javac Check.java
Check.java:1: error: package structure5 does not exist
import structure5.*;
^
Check.java:4: error: cannot find symbol
                Vector<String> v = new Vector<>();
                ^
  symbol:   class Vector
  location: class Check
Check.java:4: error: cannot find symbol
                Vector<String> v = new Vector<>();
                                       ^
  symbol:   class Vector
  location: class Check
3 errors
->
-> echo $CLASSPATH
.:/home/faculty/aaron/java:/usr/lib/java:/usr/cs-local/lib/classes
```

The `structure` package's `bailey.jar` file is not in the `CLASSPATH`, so `javac` can't find it.

Note: Your `CLASSPATH` may differ.

- If you try compiling the program without the `CLASSPATH` variable set property, then you will get several error messages.

```
-> source ~/.local_bash_profile
-> echo $CLASSPATH
.:/home/faculty/aaron/cs136/js/bailey.jar:.:/home/faculty/aaron/java:/usr/lib/java:/usr/cs-
->
-> javac Check.java
-> java Check
It works.
->
```

The `structure` package's `bailey.jar` file is now in the `CLASSPATH`.

Note: The path to `bailey.jar` should include your home folder (and not `aaron`'s).

The contents of `.local_bash_profile` and/or `.bash_profile` will run when you logout and login again. Or you can use `source` to run it without logging out (as shown above).

- If you try compiling the program with the `CLASSPATH` variable set property, then you should get the above `It works.` message.

# Class: `Association`

```
// A class for binding key/value pairs.
// (c) 1998,2001 duane a. bailey
package structure5;
import java.util.Map;
```

```
public class Association<K,V> implements Map.Entry<K,V>
{
    /**
     * The immutable key.  An arbitrary object.
     */
    protected K theKey; // the key of the key-value pair
    /**
     * The mutable value.  An arbitrary object.
     */
    protected V theValue; // the value of the key-value pair

    /*
       for example:
       Association<String,Integer> personAttribute =
           new Assocation<String,Integer>("Age",34);
     */
    /**
     * Constructs a pair from a key and value.
     *
     * @pre key is non-null
     * @post constructs a key-value pair
     * @param key A non-null object.
     * @param value A (possibly null) object.
     */
    public Association(K key, V value)
    {
        Assert.pre(key != null, "Key must not be null.");
        theKey = key;
        theValue = value;
    }
```

What do `package` and `import` mean?

What do `implements` and `Map.Entry<K,V>` mean?

- https://docs.oracle.com/javase/8/docs/api/java/util/Map.html
- https://docs.oracle.com/javase/8/docs/api/java/util/Map.Entry.html

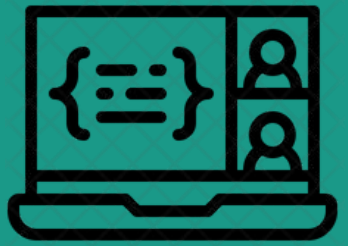
# Program: `Courses.java`

```
/**
 * A class implementing a key-value pair.  This class associates an
 * immutable key with a mutable value.  Used in many other structures.
 * <P>
 * Example Usage:
 * <P>
 * To store the number of classes a student has taken from five different
 * professors and to output this information, we could use the following.
 * <P>
 * <pre>
 * public static void main(String[] argv){
 *      //store the number of classes taken by the student in an array of associations
 *      {@link Association} [] classesTaken = new Association[5];
 *      classesTaken[0] = new {@link #Association(Object,Object) Association("Andrea", new Integer(5))};
 *      classesTaken[1] = new Association("Barbara", new Integer(1));
 *      classesTaken[2] = new Association("Bill", new Integer(3));
 *      classesTaken[3] = new Association("Duane", new Integer(2));
 *      classesTaken[4] = new Association("Tom", new Integer(1));
 *
 *      //print out each item in the array
 *      for (int i = 0; i< classesTaken.length; i++){
 *          System.out.println("This Student has taken " + classesTaken[i].{@link #getValue()} +
 *                             " classes from " + classesTaken[i].{@link #getKey()}+ ".");
 *      }
 * }
 * </pre>
 * @version $Id: Association.java 34 2007-08-09 14:43:44Z bailey $
 * @author, 2001 duane a. bailey
 */
```

The `js/src/structure5/Association.java` file includes a suggested sample program. Let's updated and fill out this starter code and create a working program called `Courses.java`.

# Live Coding: Creating `Courses.java`

Our goal is to create a program that outputs the following.
(We'll also update the faculty names!)



```
~/temp$ java Courses
This Student has taken 5 classes from Andrea.
This Student has taken 1 classes from Barbara.
This Student has taken 3 classes from Bill.
This Student has taken 2 classes from Duane.
This Student has taken 1 classes from Tom.
```

GNU nano 4.8 Courses.java

```
import structure.Association;

public class Courses {

    public static void main(String[] argv){
         //store the number of classes taken by the student in an array of associations
        Association[] classesTaken = new Association[5];
        classesTaken[0] = new Association("Aaron", new Integer(5));
        classesTaken[1] = new Association("Kelly", new Integer(1));
        classesTaken[2] = new Association("Rohit", new Integer(3));
        classesTaken[3] = new Association("Sam", new Integer(2));
        classesTaken[4] = new Association("Shikha", new Integer(1));

        //print out each item in the array
        for (int i = 0; i< classesTaken.length; i++){
             System.out.println("This Student has taken " + classesTaken[i].getValue() +
                                " classes from " + classesTaken[i].getKey()+ ".");
        }
    }

}
```

The `Courses.java` file after editing it.
A copy will be added to the course website.

# Next Steps

# Next Steps

## Lecture 5: Vectors

- Our first non-trivial data structure.

## Lecture 6: Complexity

- A tool that will help us analyze the efficiency of data structures (including vectors).

## Lab 1: Coin Strip

- You will design a data structure and use it in a simple game involving coins.

You may want to get the `structure` package and `Courses.java` program working on your machine before next week begins.