Lecture 4

Associations

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

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.

```python
>>> 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).

```python
>>> 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.
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.

```bash
$ 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 of 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.

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.

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.
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`. 
Using the `echo` command to print messages and expand environment variables.

Setting an environment variable `LAB` and using its value `$LAB`.

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

```bash
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
```
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.
Linux (shown above):

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

This step makes sure that the CLASSPATH variable is modified every time you log in.
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.
If you try compiling the program without the CLASSPATH variable set property, then you will get several error messages.

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

Note: Your CLASSPATH may differ.
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.

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).
Class: Association
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.html)
- [https://docs.oracle.com/javase/8/docs/api/java/util/Map.Entry.html](https://docs.oracle.com/javase/8/docs/api/java/util/Map.Entry.html)
Program: Courses.java
The `js/src/structure5/Association.java` file includes a suggested sample program. Let's update 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.
```
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.