Lesson 82: Sorting #1 (W29D2)

Balboa High School

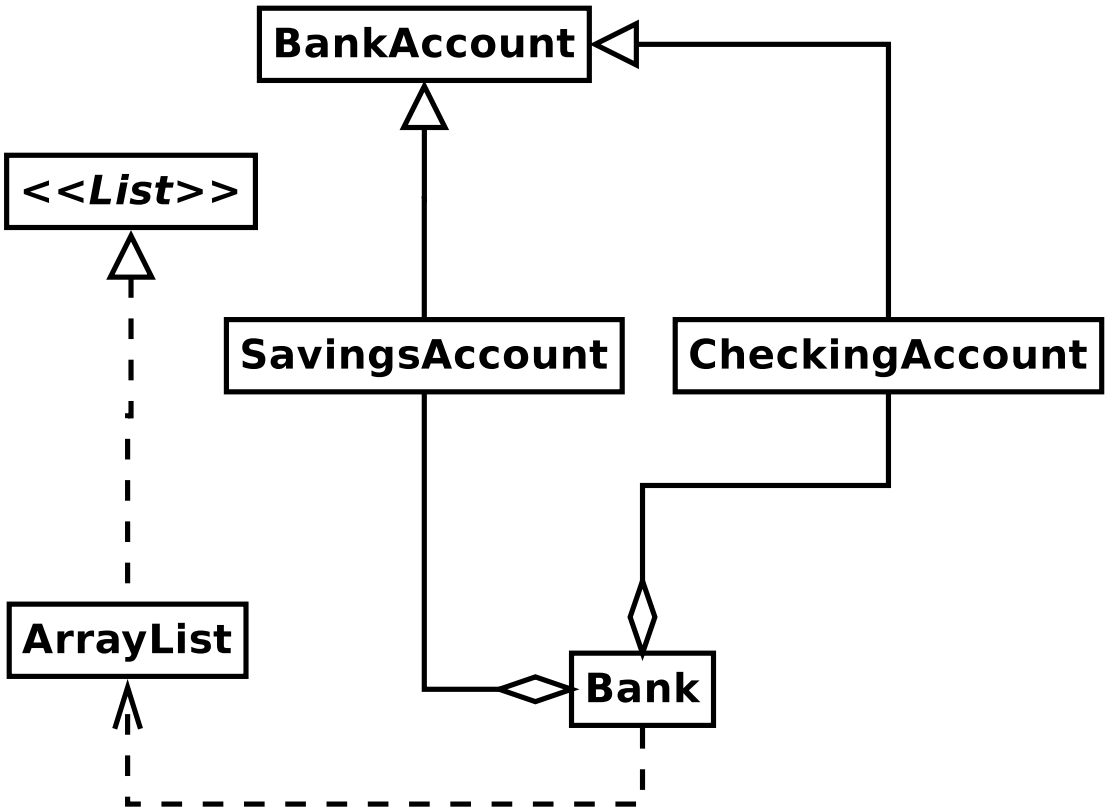
Michael Ferraro

March 15, 2016

Do Now

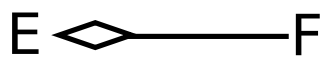
1. Create a new project in Eclipse: `L82`
2. Import sources from a JAR file:
   - download from here
   - right-click `src` folder → `Import` → `General` → `Archive File...`
3. On **paper**, construct a class diagram showing the relationship between all six classes.
   - refer to the next slide to review the various kinds of class relationships and how to represent them
   - don't focus yet on what the various methods do

Recap of Class Diagram Relationships

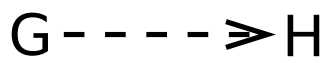


"A implements B"

"C inherits from D"
(C is subclass of D)



"E has-a F"



"G depends on H"

If a Bank has fields that are of type SavingsAccount and CheckingAccount, there's a "has-a" relationship between those classes as shown above. If the Bank uses ArrayLists in a method (as an argument or as a local variable), then there's a dependency relationship (i.e., Bank uses ArrayList). ArrayList implements the interface List, while CheckingAccount inherits from BankAccount.

Aim

Students will learn about two sorting algorithms and practice using them.

Points for Discussion

- What is the purpose of the `Sort` interface? And why an interface as opposed to an abstract class?

[1] These are thus called *class variables* and not *fields*.

Points for Discussion

- What is the purpose of the `Sort` interface? And why an interface as opposed to an abstract class?
- Why is it OK that the fields in the driver class are declared `static`?[1]

[1] These are thus called *class variables* and not *fields*.

How Do You Sort? (3-4min)

Given elements to be put in sorted order (e.g., arranging a list of `ints` in ascending order), what's the best way you can come up with to put the elts in order?

Do the following with a partner:

- Using a provided deck of cards or the provided Java applet, see if you can come up with a good way to sort elements.
- Why is your method good? Consider:
  - # of steps needed
  - how much "memory" you needed to use

Sorting Algorithms on the AP Exam

These are the sorting algorithms that you'll need to be familiar with for the AP exam:

- Selection Sort (Litvin §14.5)
- Insertion Sort (Litvin §14.6)
- Mergesort (Litvin §14.7)
- Quicksort (Litvin §14.8)

```
for( Sort s : allSorts ) {
  learnHowToPerform( s );
  understandJavaImplementation( s );
  knowWorstCaseRunningTime( s );
}
```

Selection Sort

General Idea:

- find the max elt in the set, swap with the last (n^{th}) elt
- now ignore last elt and do the same from the 1^{st} through $(n-1)^{st}$ elt

Selection Sort

- Watch a video demonstration (full screen & high resolution)
- See a convenient reference here
- You can practice here

Selection Sort

Let's follow a Java implementation as we perform the sort. Open `SelectionSort.java` in your Eclipse editor.

Things to watch for:

- how to keep track of the max value seen so far
- how to swap elts

Insertion Sort

General Idea:

- start n at 1 (meaning the first elt, which is at position 0 in an array)
- always keep the 1^{st} through n^{th} elts in order
- when the $(n+1)^{st}$ elt is not in the right place, shuffle elts to the right and *insert* it where it belongs in order

Insertion Sort

- Watch a video demonstration (full screen & high resolution)
- See a convenient reference here
- You can practice here

Classwork

With a partner, sort these arrays **on paper** twice — once using Selection Sort and again using Insertion Sort.

1. | -8 | 4 | -11 | 1 | 0 |
2. | 1 | 2 | 3 | 4 | 0 |
3. | 50 | 10 | 10 | 11 | 12 | 13 |

HW

1. Read through `InsertionSort.java` and practice applying it to a few arrays ON PAPER. You will need to understand how the code works for the quiz in two class days!
2. Check out this vintage sorting algorithms video from University of Toronto.
3. Look for online animations for Mergesort and Quicksort (see YouTube and sorting-algorithms.com). That will help prepare you for the next lesson.