Topics: Review of OOP concepts, parameter passing, one-dimensional array

Reading (JV): Sec 6.1, "Selection Sort" in Sec 6.3

OOP ideas

- An object has *variables* (characteristics) and *methods* (behaviors). These are *instance variables* and *methods*.
- A class definition is the "blue print" or "plan" for making objects. Characteristics and behaviors that don't logically belong to each object but relates to the entire collection of objects are *class variables* and *methods*. Keyword: **`static`**
- An object is *an instance* of a class
  ⇒ the class ≠ the object
  ⇒ defining a class ≠ creating the object
- Use keyword **`new`** to create an object
- An object is *referenced*, not stored *in* a variable
- Methods perform action. An *instance method* executes from (on) a particular instance.
- Method headers are "contracts" between the programmer and the user. Knowing the method header allows the user to *use* the method without knowing how the method actually works.
- Encapsulation:
  - Protect data
  - Hide details from clients
  - Keyword: **`private`**

Why doesn't this swap work?

```
/* Try to swap two values */
public class BadSwap {
   public static void main(String[] args){
      int x = Keyboard.readInt();  int y = Keyboard.readInt();
      swap(x,y);
   }
   public static void swap(int x, int y) {
      int tmp;
      tmp = x;
      x = y;
      y = tmp;
   }
} //class BadSwap
```

Arrays

- Arrays are objects. An array is an ordered list of values (or objects) of one type
- The entire array has one name (identifier)
- Each element in the array has an integer index (begins at 0)
- An array of size *N* is indexed from 0 to *N*-1

Array declaration and construction

- Declaration syntax: ***`type`*`[] identifier;`**
  Examples:

```
int[] counts;
double[] price;
String[] names;
Room[] caves;  // assuming a Room class has been defined
```

- Instantiation syntax: **new *type*[*size*]**
  size is an integer
  Example:
  ```
  new int[4]
  ```
- Declartion and instantiation
  ```
  int limit = 4;
  double[] price;                // declaration
  price = new double[limit];     // instantiation and assignment
  ```

Index operator []

The expression ***identifier*[*integer_expression*]** accesses an element in the array referred to by ***identifier***

Examples:
```
int[] freq = new int[101];  // declaration & instantiation
freq[9+70] = 17;  // set freq[79] to 1 (freq[79] is the 80th element in freq)
int grade = 79;
freq[grade] = freq[grade] + 1;
freq[grade]++;
```

In the example above, the expression **freq[2]** represents an integer and can be used anywhere an **int** variable can be used.

The size of an array is held in the constant **length**. **length** is *automatically* defined when an array is created and *cannot be changed*. In the above example, the expression **freq.length** gives the size of the array **freq**.

Pattern for processing an array

```
// assume an array has been created and is referred to by variable arr
for (i=0; i<arr.length; i++) {

  // perform some process (on arr[i])

}
```

Sorting

- Arrange elements in a list by some specified order
- Sort "in-place" means sort without using extra memory space for holding another copy of the array
- There are many sorting algorithms: ***selection sort***, *insertion sort*, *bubble sort*, etc.

Template for selection sort (ascending order)

```
// loop from first to second last element

      // find index of minimum value _____________________________


      // swap ith element with minimum value
```