Announcements

- Read for next time Chap. 11.1
  - Files
- Classwork from today is due Thurs, Sep 9
  - Finish before the next class
- Reading Quiz for next time

What we will do today

- Review Classwork from last time
- Lecture
  - Arrays
- Classwork today
  - APTs using arrays and loops

Classwork from last time

- Sqrt
  - Review examples
  - Why create estimate method?
- Needle
  - Review examples
  - Why have Needle class?
Arrays

Array Access

Array Syntax

Array

- Creating an array
  \texttt{new \texttt{typeName}[length]}
  \textbf{Example:} new \texttt{double[10]}
  \textbf{Purpose:} To construct an array with a given number of elements.

- Accessing elements
  \texttt{arrayReference[index]}
  \textbf{Example:} data[2]
  \textbf{Purpose:} To access an element in an array.

- Declare and initialize an array of integers
  \texttt{int[] values = \texttt{new int[12];}}

- Set it to these values:
  \begin{tabular}{cccccccccccc}
  8 & 3 & 4 & 3 & 8 & 2 & 4 & 4 & 6 & 2 & 8 & 4 \\
  \end{tabular}

- Access item in slot 6 in the array
  \texttt{values[6]}

- Array is fixed size. The size is:
  \texttt{values.length}
Self Check 7.1
What elements does the data array contain after the following statements?
```java
double[] data = new double[10];
for (int i = 0; i < data.length; i++) data[i] = i * i;
```
Answer:
```
(0, 1, 4, 9, 16, 25, 36, 49, 64, 81)
```

Self Check 7.2
What do the following program segments print? Or, if there is an error, describe the error and specify whether it is detected at compile-time or at run-time.
a) ```java
double[] a = new double[10];
System.out.println(a[0]);
```
b) ```java
double[] b = new double[10];
System.out.println(b[10]);
```
c) ```java
double[] c;
System.out.println(c[0]);
```
Answer:
a) 
b) 
c)

Loops
- Traverses all elements of a collection:
  ```java
double[] data = ...;
double sum = 0;
for (double e : data)
  // Read this loop as
  // "for each e in data"
  { 
    sum = sum + e;
  }
```
- Traditional alternative:
  ```java
double[] data = ...;
double sum = 0;
for (int i = 0; i < data.length; i++)
  { 
    double e = data[i];
    sum = sum + e;
  }
```

ArrayList
- Class vs. primitive
- ArrayList
  - Can grow and shrink
  - Has methods for common tasks (see API)
  - Only holds objects
- Can’t have an ArrayList of int or double
  - Need to use wrapper class like Integer or Double
ArrayList (cont)

- Create an ArrayList
  ```java
  ArrayList<Integer> idlist = new ArrayList<Integer>();
  ```
- Add an element to the ArrayList
  ```java
  idlist.add(8);
  ```
- Modify kth element in an ArrayList
  ```java
  idlist.set(k, 8);
  ```
- Sum the elements in the ArrayList
  ```java
  // sum up integers in the ArrayList
  int sum = 0;
  for (Integer current : idlist)
  {
      sum += current;
  }
  ```

ArrayList vs. array

- Methods
  - Sort an ArrayList called numbers
    ```java
    Collections.sort(numbers);
    ```
  - Sort an array called a
    ```java
    Arrays.sort(a);
    ```
- Types
  - Arrays can hold any type
  - ArrayLists only work with objects
- Can convert from one to the other
- APTs only pass and return arrays

Example: singleNumbers

- Given an integer array that could have duplicates, return an array that has only unique numbers from the original array (get rid of duplicates!)
- For example if the parameter array is:
  - 8 5 5 8 5
- Then the array to return should be:
  - 8 5

First convert array to ArrayList

```java
public int[] singleNumbers(int[] ids) {
    // convert the array "ids" into an ArrayList "idlist"
    ArrayList<Integer> idlist = new ArrayList<Integer>();
    for (int k = 0; k < ids.length; k++) {
        idlist.add(ids[k]);
    }
    // further processing...
}
```
Second, find unique numbers

```java
// create an ArrayList that will hold unique numbers
ArrayList<Integer> singles = new ArrayList<Integer>();
singles.add(idlist.get(0)); // first number is unique
for (Integer current : idlist) {
    boolean isIn = false;
    for (Integer currentSingle : singles) {
        if (current.equals(currentSingle))
            isIn = true;
    }
    if (!isIn)
        singles.add(current);
}
```

or…

- Convert ArrayList to array
  Use ArrayList’s `toArray()` method
  ```java
  Integer[] answer = (Integer[]) singles.toArray();
  ```

- Convert array to ArrayList
  Use Array’s static `asList()` method
  ```java
  ArrayList<String> nameList =
  (ArrayList<String>) Arrays.asList(names);
  ```

Third, convert ArrayList to Array

```java
// convert ArrayList to array
int[] answer = new int[singles.size()];
int position = 0;
for (Integer currentSingle : singles) {
    answer[position] = currentSingle;
    position++;
}
return answer;
```

Strings

- String
  - a sequence of characters
  - `objects` of the `String` class
- String constants:
  - "Hello, World!"
- String variables:
  - `message` = "Hello, World!"
- String length:
  - `message.length()`
- Empty string: ""
- Concatenating Strings
  - Use the `+` operator:
    ```java
    String name = "Dave";
    String message = "Hello, " + name;
    ```
- Automatic type conversion
  ```java
  String a = "Agent";
  int n = 7;
  String bond = a + n; // bond is "Agent7"
  ```