Java Basics – Arrays

- **Should be a very familiar idea**
  - Problem: Deal with exam grades in a course
    - Could have variable for each student
    - Would need unique name for each variable
    - Need lots of custom code
    - Instead, assume named array; use index to get values

- **Example: method to count number of A grades**
  ```java
  public static int getAs(int[] grades) {
      int aCount = 0;
      for (int k=0; k<grades.length; k++) {
          if (grades[k] >= 90) {
              aCount++;
          }
      }
      return aCount;
  }
  ```

- **Explain**

Java Basics – Arrays

- **Array Bounds**
  - Index must in range
  - 0 <= k < grades.length
  - Else get `ArrayIndexOutOfBoundsException`

- **Short Circuit Evaluation**
  - Allows expression of the form
  ```java
  if (k>=0 && k<grades.length && grades[k] >= 90) {
      aCount++;
  }
  ```
  - Safe from exception
  - Order important!
  - Holds for any chain of ANDed terms
  - Similar rules for chain of ORed terms

Java Basics – Arrays

- **Creating Arrays (by example)**
  ```java
  int[] counts = new int[101];
  String[] colors = new String[3];
  Counter[] tallies = new Counter[10];
  int[] codes;
  codes = new int[17];
  ```

- **Using Initializer Lists**
  ```java
  String[] colors = {"red", "green", "blue"};
  double[] shims = {1.0, 1.1, 1.3, 1.6, 1.9, 2.5};
  ```

- **Initializing Object Arrays**
  - For most object arrays, need to create objects in the array
  ```java
  for (int k = 0; k < tallies.length; k++) {
      tallies[k] = new Counter();
  }
  ```

Java Basics – Arrays

- **Arrays are Objects**
  - Behavior of arrays similar to other objects
  - Thus `grades.length` works

- **Assignments (Warning!**
  - Since array identifiers are just references
    - Array assignment doesn’t create new array!
  - Use `newArrayname = arrayname.clone();`
    - This works well for arrays of primitives
    - What happens for arrays of objects?

- **Shallow vs Deep Copy**
  - Java vs C++
Java Basics – Simple I/O

- **Output methods (Java console)**
  - `System.out.println(. . .);`
  - `System.out.print(. . .);`
  - Overloaded for `String` and primitive types

- **Warning: tries to convert argument to string**
  - What is the output for the following?
    ```java
    int k = 5;
    System.out.println(k);
    System.out.println(k + 1);
    System.out.println(k - 1);
    System.out.println("the answer is " + k);
    System.out.println("the answer is " + k + 1);
    System.out.println("the answer is " + k - 1);
    ```

Java Basics – Simple I/O

- **Input methods (Java console)**
  - Need to use `Scanner` object
  - Parses the input and give us back tokens.

- **Scanner Class**
  - Use `next()` method where `type` is primitive
  - Use `next()` for `String`;
    ```java
    Scanner s = new Scanner(System.in);
    double d = s.nextDouble();
    String s = s.next();
    ```
  - You've used this in lab

- **Check out Scanner class**

Java Basics – Classes and Packages

- **Class must be in file**
  - Filename must be `className.java`
  - If it is an application, it must include `public static void main(String[] args)` method

- **Nested Classes**
  - Defined inside of a class
  - Usually used only by the outer class

- **Packages**
  - A set of classes in a subdirectory can be a package
  - Directory name matches package name
  - Each file must start with `package packageName;`