Introduction to Arrays

Array: collection of values stored under a single reference

- Declaration: data type [] name;
- Instantiation: name = new data type[size];
- Initialization:

```
for (int x=0; x<name.length; x++) {
     name[x] = expression;
}
```

- Example: TestDice

Arrays as parameters / return values

- Individual array elements can be passed to methods just like any other variable
- Passing whole arrays: "status" appears in the method declaration; for example:

```
public int [] getArray() // heading indicates an array return value
public void fillArray(int [] array) // indicates array parameter
```

- Method call just uses name of array, not array notation
    - Example: StringBackwards
- Important note: arrays are reference parameters – this means you can change the values in an array while a method is running, and the changes will remain after the method returns
    - Example: DeckOfCards

Arrays of Objects

- Instantiating array just creates a set of empty objects
- Each individual object must be instantiated in and of itself

Example:

```
Random rg = new Random();
Fraction [] ratios = new Fraction[100];  // creates empty array
for (int x=0; x<ratios.length; x++) {     // populates with objects
     ratios[x] = new Fraction(rg.nextInt(10)+1, rg.nextInt(20)+1);
```

Array Applications

- Frequency counting: CountASCII
- Sorting: Sorts

2 (or more)-dimensional arrays

- 1st dimension is always number of rows (row-major order)
- A row can contain columns (2nd dimension): row is an array of arrays
- More dimensions possible (3D not uncommon)
- Example: TextEncrypt