`this` keyword

- **`this`** : A reference to the *implicit parameter*
  (the object on which a method/constructor is called)

- Syntax:
  - To refer to a field: `this.`**`field`**
  - To call a method: `this.`**`method`**`(`**`parameters`**`);`
  - To call a constructor from another constructor: `this(`**`parameters`**`);`

Preconditions

- **precondition**: Something your method *assumes is true* at the start of its execution.
  - Often documented as a comment on the method's header:

```
// Returns the element at the given index.
// Precondition: 0 <= index < size
public int get(int index) {
    return elementData[index];
}
```

  - Stating a precondition doesn't really "solve" the problem, but it at least documents our decision and warns the client what not to do.
  - What if we want to actually enforce the precondition?

Throwing exceptions (4.4)

```
throw new ExceptionType();
throw new ExceptionType("message");
```

- Generates an exception that will crash the program, unless it has code to handle ("catch") the exception.

- Common exception types:
  - `ArithmeticException, ArrayIndexOutOfBoundsException, FileNotFoundException, IllegalArgumentException, IllegalStateException, IOException, NoSuchElementException, NullPointerException, RuntimeException, UnsupportedOperationException`

- Why would anyone ever *want* a program to crash?

The `Arrays` class

- The `Arrays` class in `java.util` has many useful methods:

Method name	Description
`binarySearch(`**array**`, `**value**`)`	returns the index of the given value in a *sorted* array (or < 0 if not found)
`binarySearch(`**array**`, `**minIndex**`, `**maxIndex**`, `**value**`)`	returns index of given value in a *sorted* array between indexes *min* / *max* - 1 (< 0 if not found)
`copyOf(`**array**`, `**length**`)`	returns a new resized copy of an array
`equals(`**array1**`, `**array2**`)`	returns `true` if the two arrays contain same elements in the same order
`fill(`**array**`, `**value**`)`	sets every element to the given value
`sort(`**array**`)`	arranges the elements into sorted order
`toString(`**array**`)`	returns a string representing the array, such as `"[10, 30, -25, 17]"`

- Syntax: `Arrays.`**`methodName`**`(`**`parameters`**`)`

Postconditions

- **postcondition**: Something your method *promises will be true* at the *end* of its execution.
  - Often documented as a comment on the method's header:

```
// Makes sure that this list's internal array is large
// enough to store the given number of elements.
// Postcondition: elementData.length >= capacity
public void ensureCapacity(int capacity) {
    // double in size until large enough
    int newCapacity = elementData.length;
    while (capacity > newCapacity) {
        newCapacity = newCapacity * 2;
   }
   elementData = Arrays.copyOf(elementData, newCapacity);
}
```

  - If your method states a postcondition, clients should be able to rely on that statement being true after they call the method.

Tips for testing

- You cannot test every possible input, parameter value, etc.
  - Think of a limited set of tests likely to expose bugs.
- Think about boundary cases
  - Positive; zero; negative numbers
  - Right at the edge of an array or collection's size
- Think about empty cases and error cases
  - 0, -1, null; an empty list or array
- test behavior in combination
  - Maybe `add` usually works, but fails after you call `remove`
  - Make multiple calls; maybe `size` fails the second time only