Assertions

- Assertion is statement of truth about aspect of a program's logic
  - Like a boolean statement that should be true at a certain point
- Assertions can be written as comments to identify design logic

```
// Assertion: intVal >= 0
```

Question 4 Suppose that you have an array of positive integers. The following statements find the largest integer in the array. What assertion can you write as a comment after the for loop?

```
int max = 0;
for (int index = 0; index < array.length; index++)
{
   if (array[index] > max)
         max = array[index];
} // Assertion:
```

4. // Assertion: max is the largest of array[0],..., array[index]

Assert Statement

- **`assert someVal < 0;`**
- if true, program does nothing
- If false, program execution terminates

  Exception in thread “main” java.lang.AssertionError
- **assert sum > 0 : sum;**

  adds the value of sum to the error message in case sum ≤ 0.

Or

  - **assert sum > 0 : "sum greater than zero";**

- By default, assert statements are disabled at execution time.

Assertions

Assertions are a form of testing that allow you to check for correct assumptions throughout your code. For example: If you assume that your method will calculate a negative value, you can use an assertion.

- Assertions can be used to check internal logic of a single method:
  - Internal invariants
  - Control flow invariants
  - Class invariants
- Assertions can be disabled at run time; therefore:
  - Do not use assertions to check parameters.
  - Do not use methods that can cause side effects in an assertion check.

Assertion Syntax

There are two different assertion statements.

- If the <boolean_expression> evaluates as false, then an AssertionError is thrown.
- A second argument is optional, but can be declared and will be converted to a string to serve as a description to the AssertionError message displayed.

```
assert <boolean_expression> ;
assert <boolean_expression> : <detail_expression> ;
```

Internal Invariants

Internal invariants are testing values and evaluations in your methods.

```
if (x > 0) {
    // do this
} else {
    assert ( x == 0 );
    // do that
    // what if x is negative?
}
```

Control Flow Invariants

Assertions can be made inside control flow statements such as shown below. These are called control flow invariants.

```
switch (suit) {
    case Suit.CLUBS: // ...
        break;
    case Suit.DIAMONDS: // ...
        break;
    case Suit.HEARTS: // ...
        break;
    case Suit.SPADES: // ...
        break;
    default:
        assert false : "Unknown playing card suit";
        break;
}
```

Class Invariants

A class invariant is an invariant used to evaluate the assumptions of the class instances, which is an O*bject* in the following example:

```
public Object pop() {
    int size = this.getElementCount();
    if (size == 0) {
        throw new RuntimeException("Attempt to pop
from empty stack");
}

Object result = /* code to retrieve the popped
element */ ;
// test the postcondition
    assert (this.getElementCount() == size - 1);

    return result;
}
```

Terminology

Key terms used in this lesson included:

Assertions

Internal Invariant

Control Flow Invariant

Class Invariant