A Little Bit of Unit Testing

October 17, 2024

Most Useful Function

```
assertEquals(expected, actual)
```

how it works

```
if expected and actual are the same
   test passes
else
   test fails
```

- function defined for several different types
- `expected`, `actual` must be same type
- typical `actual`: call function to test

Example

```
assertEquals(expected, actual)
```

```
void testSum() {
    assertEquals(10, LameMath.sum(2,8));
    assertEquals(-10, LameMath.sum(-2,-8));
    assertEquals(-6, LameMath.sum(2,-8));
    assertEquals(0, LameMath.sum(0,0));
}
```

For True/False Values

```
assertTrue(some boolean condition)
assertFalse(some boolean condition)
```

Example

```
void testIsEven() {
    assertTrue(LameMath.isEven(2));
    assertTrue(LameMath.isEven(42));
    assertTrue(LameMath.isEven(-16));
    assertFalse(LameMath.isEven(-1));
    assertFalse(LameMath.isEven(11));
}
```

Testing Arrays

```
assertArrayEquals(expectedArrayRef, actualArrayRef)
```

Note

- like `assertEquals` but for arrays
- array elements objects of class? test uses class' `equals()`

Testing Arrays Example

```
assertArrayEquals(expectedArrayRef, actualArrayRef)
```

Example

```
public static void fill(int []A, int initVal) {
    for (int i = 0; i < A.length; i++) {
        A[i] = initVal;
    }
}
...
@Test
void testFill() {
    int []A = new int[SIZE];
    int []expected = {3, 3, 3, 3, 3};

    ArrayStuff.fill(A, 3);
    assertArrayEquals(expected, A);
}
```

Testing Arrays: Another Example

```
assertArrayEquals(expectedArrayRef, actualArrayRef)
```

Another Example. Return Array

```
public static int []newArr(int size, int initVal) {
    int []A = new int[size];

    for (int i = 0; i < A.length; i++) {
        A[i] = initVal;
    }

    return A;
}
...

@Test
void testNewArr() {
    assertArrayEquals(new int[] {3, 3, 3, 3, 3},
            ArrayStuff.newArr(SIZE, 3));
}
```

JUnit in Eclipse

To Add JUnit to Project

- Right click on a class (⌘-click on a Mac)
- File $\rightarrow$ New $\rightarrow$ JUnit Test Case
- Be sure "New JUnit Jupiter test" is selected
- Click "Next"
- Check off functions you want tested, then click "Finish"

To Run the Tests

- Right click on the test class (⌘-click on a Mac)
- Run $\rightarrow$ Run as $\rightarrow$ JUnit Test