Software Testing

Test case (collection of which is called a test suite)

- define all input values, conditions, or variables
- define a procedure
- define the expected behavior

Writing Tests

- setup initial conditions
- call the method being tested
- check for the expected behavior

black-box or functional testing - designing tests solely based on input and output behavior

white-box, structural, or logic-driven testing - designing tests using the program's logic

unit testing - testing one class at a time

integration testing - testing multiple classes together

system testing - testing the whole project at once

class: FlowerPicker

test class: FlowerPickerTest

Sample Test Method Syntax:

```java
public void testPickFlowers()
{
    // 1. set up initial conditions
    Lab04Island island = new Lab04Island();
    FlowerPicker picker = new FlowerPicker();
    island.addObject(picker, 1, 2);

    // 2. call the method
    picker.pickFlowers();

    // 3. check expected results
    assertThat(picker.getX()).isEqualTo(6);
    assertThat(picker.getY()).isEqualTo(2);
    assertThat(picker.getFlowers()).isEqualTo(5);
    assertEquals(5, picker.getFlowers());
    assertThat(picker.getHeading()).isEqualTo(EAST);
}
```

assertThat() - checks if something is as expected

assertEquals()

Implement pickFlowers:

```java
public void pickFlowers()
{
    while (this.seesFlower(AHEAD))
    {
        this.hop();
        this.pick();
    }
}
```

More About Methods

```java
public void pickFlowersAndDisableNets() // returns nothing, is called for the actions it performs
public int addHops() // returns an int value as its result

// Passing information using parameters
public void turnAndDisable(RelativeDirection direction)
turnAndDisable(RIGHT); // to call the method

public void turnThenHop(RelativeDirection direction, int hops) // multiple parameters
turnThenHop(RIGHT, 7);
```

Short Circuit Evaluation - boolean expression evaluated from left to right

```java
basil.isFacing(WEST) && basil.seesNet(AHEAD)
basil.isFacing(NORTH) || basil.seesNet(AHEAD)

Removing the parenthesis flips between && and ||
!(A && B) is the same as !A || !B

!(A || B) is the same as !A && !B
```

		Relational Operators
Operator	**Example**	**Meaning**
`==`	`x == y`	x *is equal to* y
`!=`	`x != y`	x *is not equal to* y
`>`	`x > y`	x *is greater than* y
`<`	`x < y`	x *is less than* y
`>=`	`x >= y`	x *is greater than or equal to* y
`<=`	`x <=  y`	x *is less than or equal to* y