Preface
In this document, we discuss methodologies for unit testing your models.
Using a Framework

There are a few Java frameworks that you can use to Unit test your models, however, they are beyond the scope of this class. We recommend writing your own unit tests using this guide.

Suppose that we create a Person class as follows:

```java
public class Person {
    private String firstName;
    private String lastName;
    private int age;

    public Person(String firstName, String lastName, int age) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
    }

    public String getFirstName() {
        return this.firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return this.lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public int getAge() {
        return this.age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

In short, we have the following methods:

- `new Person(firstName, lastName, age)`
- `new Person(fullName, age)`
- `person.setFirstName()`
- `person.getFirstName()`
- `person.setLastName()`
- `person.getLastName()`
- `person.setAge()`
- `person.getAge()`;

We need to exhaustively test each of these methods. We recommend creating a java class called `[ModelName]Tester.java`. In it, you define all your test data and produce meaningful output.

```java
class PersonTester {
```
private String[] firstNames = {"Bill", "Bob", "Ben", "Joe", "Seth"};
private String[] lastNames = {"Jones", "Smith", "Plato", "Jacobs", "Vargo"};
private int[] ages = {22, 34, 55, 22, 92};

public static void main(String args[]) {
    testConstructor();
    testFirstName();
    testLastName();
    testAge();
}

public static void testConstructor() {
    for(int i = 0; i < firstNames.length; i++) {
        Person p = new Person(firstNames[i], lastNames[i], ages[i]);
        assert(firstNames[i].equals(p.getFirstName()));
        assert(lastNames[i].equals(p.getLastName()));
        assert(ages[i] == p.getAge());
    }
}

public static void testFirstName() {
    Person p = new Person(firstNames[0], lastNames[0], ages[0]);
    for(int i = 0; i < firstNames.length; i++) {
        p.setFirstName(firstNames[i]);
        assert(firstNames[i].equals(p.getFirstName()));
    }
}

public static void testLastName() {
    Person p = new Person(firstNames[0], lastNames[0], ages[0]);
    for(int i = 0; i < lastNames.length; i++) {
        p.setLastName(lastNames[i]);
        assert(lastNames[i].equals(p.getLastName()));
    }
}

If any of the tests fail, you can trace the error and determine where it's failing. Note that with Unit Testing, you only test a single Class at a time. Never would you call a method of another class. After you complete unit testing, you'll want to create another test file specifically for Integration testing.