Video Name: JUnit Instance

Topics:

- testing instance methods
- @Before
- testing methods that return floating point values
- adding JUnit library to your project

Java Class(es):

- junitGame package, Location, LocationTest
- utilities package, MyConversions
- tests package, TestConversions

Testing instance methods

- We don’t typically test getters/setters
- Need to test methods that have some real behavior/complex logic

Need to set up objects for testing. One option is to use a setUp method. The @Before annotation tells the JUnit framework that this method should be run before every test method (i.e., it will reset the object to a known state)

May also create an instance variable which is local to the test method. No need for @Before.

Remember: static methods do not need an object. Instance methods do. So to test an instance method, must create an object that can be used for testing.

Floating point values. Tests often check for equality between a returned value and an expected value. With int values, this comparison can be done exactly. But since floating point values may not have an exact representation, it is not safe to compare two floating points using assertEquals. Instead, must use a tolerance, such that the numbers are considered equal if the difference is within some very small epsilon.

Code:

```java
public static final double EPSILON = .0001;
assertEquals(expected, actual, EPSILON);
```
JUnit Build Path. If JUnit is not on the build path (e.g., if sharing code between teammates, JUnit might not be on the build path). All the JUnit functions such as assertEquals will generate compiler errors. You need to:

- Right click on project
- Select Build Path
- Add Library
- Select JUnit then JUnit 4
- Finish/OK