|  |
| --- |
|  |

CS308 Building Software Systems

JUnit Testing Strategy

[MW-Wed-Team-4] | | [20/2/18]

# Identifying JUnit Test Cases

In the model package of the Gizmo-ball system, we intend to implement a fair amount of JUnit test cases to test the development of the system thoroughly. We may also use a test suite in Intellij to run multiple unit tests at once. We also intend to use the testing coverage tool to test line coverage and aim to have a high percentage of code coverage in the automated tests.

In our implementation we intend to override the equals and hash code methods and test these vigorously using the JUnit tests, testing equality of various objects. We will test for each relevant functional requirement, and test that these meet the validation of the initial system specification. This will ensure the development does what it is supposed to do and checks errors in the system.

Implementation of unit test cases to test equality, symmetry, transitivity and reflexivity. Testing objects against one another to test for both equality and equivalence. The java API provides multiple test case methods which we will utilize such as

# Our Approach to Junit Testing

Using the test-driven development approach, we intend to code a little, test a little and continues this process throughout the development phase. Running the tests frequently and often will also identify any new bugs found within the system which can then be changed.

Ideally when coding the main implementation of the system, we would like to write the test cases first and then code the system to meet the requirements of the test cases. Forcing us to think about the main requirements and designing the system to meet those requirements.