CISC124 – Today’s Topics

- Software Testing
- Junit framework

Testing Setup

Core Activities in Testing Medium-to High Complexity-Software:

- **Environment Setup** → Hardware, Code Instrumentation, Test Automation Scripts
- **Data Preparation** → Sets of Data Representing Special Events, Acceptable Values and Value, Ranges, Border Conditions
- **Verification of Assertions** → Code-Oriented Assertions, Systemic Assertions, Software System Operation Assertions

Aspects of Testing

Core Aspects of Testing:

- **Abstraction** → Easily supports modifications to handle scaled up problems within a limited scope
- **Coverage** → Capable to operate on different hardware and software with minimal modifications
- **Consistency** → Easy to modularize and segregate components for testing and integration

Levels of Testing

- **Unit Testing** → Scope is small sections of code under test for specific values, ranges and combinations of values
- **Integration Testing** → Scope is interactions between sections of code (messages, functions calls, interaction protocols) and their insertion in the operational environment
- **User Acceptance Testing** → Verification of correct responses produced by the software to high-level actions initiated by its users

Junit Framework

- **SCOPE IS UNIT TESTING OF JAVA CLASSES AND THEIR METHODS**
  - **Environment** → Requires a hosting environment (i.e. Eclipse) to easily assemble and run test cases.
  - **Test Workflow** → Assembled in one or more TEST CLASSES containing Junit stubs that configure the testing actions to perform on one or more CLASSES UNDER TEST and their methods
  - **Test Actions** → Expressed in Junit stubs that configure their sequencing or implement assertions about the code in the CLASS UNDER TEST.

Winter 2019 CISC124

Winter 2018 CISC124 – Section 2

Winter 2018 CISC124 – Section 2
### JUnit Framework

- **JUNIT STUBS:**
  - **FLOW CONTROL**
    - `@BeforeAll` (Test method before a Class)
    - `@AfterAll` (Test method after a Class)
    - `@BeforeEach` (Test method before each Method)
    - `@AfterEach` (Test method after each Method)
  - **ASSERTIONS**
    - `assertEquals()`, `assertEqualsArray()`, `assertNotEquals()` (for values or arrays of values)
    - `assertSame()`, `assertNotSame()` (for objects)
    - `assertTrue()`, `assertFalse()` (for boolean expressions)
    - `assertNull()` (for references)

---

### JUnit Test Example

- **CLASS UNDER TEST**: Student
  - **FOUR METHODS UNDER TEST**
  - **TEST CLASS**: StudentTest
    - IMPLEMENTS FLOW CONTROL
    - IMPLEMENTS VARIOUS TEST CASES FOR THE FOUR METHODS UNDER TEST