



SWE 434

SOFTWARE TESTING AND VALIDATION

LAB



INTRODUCTION



GRADING

- **20% of the course grade.**
- **For class participations and Quizzes.**



WHY TESTING?

- **Software is everywhere.**
- **Software contains defects. They hide.**
- **Defects can cause software failures.**
- **Failures can cost money, and even be mortal!**
- **Testing assures the quality of the software.**
- **Testing accelerates software development.**

**Software testing is a fundamental part
of professional software development!**

WHY TESTING?

- More than 3,200 US prisoners have been released early because of a software glitch.



Technology

US prisoners released early by software bug

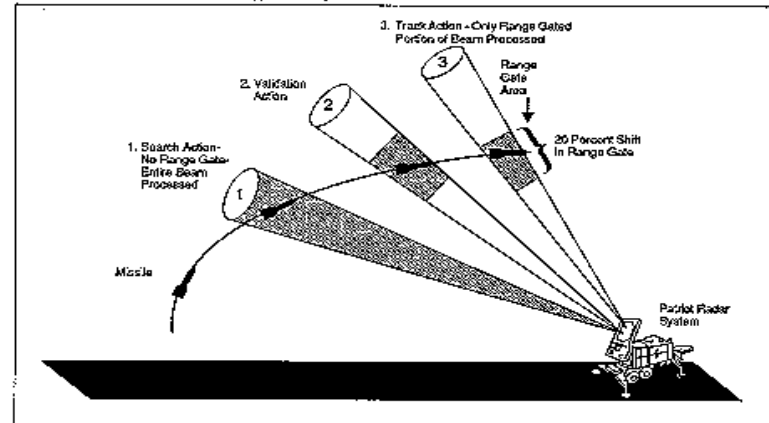
© 23 December 2015 | Technology

WHY TESTING?

- AL Dhahran scud Attack 1990, killing 28 U.S. soldiers and injuring another 98.
- Patriot's failed to track and intercept the incoming Scud because of a bug in Patriot's radar and tracking software



Figure 4: Calculated Range Gate After Approximately 8 Hours





WHY TESTING?

- Healthcare.gov.
- No proper testing.
- Testing was left to the last minute.

The screenshot shows the Healthcare.gov website interface. At the top, there is a navigation bar with the logo, "Learn", "Get Insurance", "Log in", and "Español" buttons. Below this is a secondary navigation bar with "Individuals & Families", "Small Businesses", and "All Topics" dropdown menus, along with a search bar and a "SEARCH" button. The main content area displays a large heading: "The System is down at the moment." followed by the text: "We're working to resolve the issue as soon as possible. Please try again later." Below this, there is a support contact instruction: "Please include the reference ID below if you wish to contact us at 1-800-318-2596 for support." and an error message: "Error from: https%3A//www.healthcare.gov/marketplace/global/en_US/registration%23signUpStepOne" with a reference ID: "0.cdd74f17.1380634949.2f9c301c". At the bottom, there is a "Health Insurance Marketplace" logo, a "181 DAYS LEFT TO ENROLL" counter, and a calendar showing enrollment periods: "OCT 1 Open Enrollment Began", "JAN 1 Coverage Can Begin", and "MAR 31 Open Enrollment Closes". A "Live Chat" button is located in the bottom right corner.



WHY TESTING IS IMPORTANT TO ME?

- **Job opportunities in Saudi Arabia.**
- **There are few professional testers!**
- **It is a great if you are a developer and you know Testing.**



TYPES OF TESTING

- Unit testing
- Verification
- Regression testing
- Inspections
- Static analysis
- Symbolic execution
- Program slicing
- Acceptance testing
- Integration testing
- Data-flow analysis
- Reliability testing
- Penetration testing
- Cleanroom development
- Web crawlers
- Use case testing

MAIN GOAL

- Understand and Gain hands-on experience with popular testing tools and techniques



Cucumber

JUnit



MANUAL TESTING

- **How do you test your code?**



MANUAL TESTING

Print statement? Is it enough?

```
anaconda> python DPLLS.py -f CNF_sentences.txt
DPLLS.py
Reading sentences no 1 .....
<True, {'Q': False, 'P': False}, set([])>
Done!
Reading sentences no 2 .....
<True, {'A': False, 'C': True, 'B': False}, []>
Done!
Reading sentences no 3 .....
True {'A': True, 'C': True} ['B']
Done!
Reading sentences no 4 .....
True {'A': True, 'C': True, 'B': False, 'E': True, 'D': False, 'G': True, 'F': True, 'I': True, 'H': True, 'K': False, 'M': True, 'L': True, 'O': False, 'N': False, 'P': True} ['J']
Done!
Reading sentences no 5 .....
True {'Q': False, 'P': True, 'S': False, 'R': True} []
Done!
Reading sentences no 6 .....
True {'A': True, 'C': True, 'B': True} []
Done!
Reading sentences no 7 .....
True {'Q': True, 'P': True} set([])
```

Manual Testing using excessive print statements



MANUAL TESTING

- Debugging the code using *system.out.println()* will lead to manual scanning of the whole output every time the program is run to ensure the code is doing the expected operations.
- in the long run, it takes lesser time to code JUnit methods and test them on class files.

```
public class manualTest {  
  
    //Trivial Example to show how print statements usually used to verify code  
    //@ali  
    public static void main(String[] args) {  
        int x= 5 ;  
        int y =3;  
        int RectangleArea=x*y;  
        System.out.println("The area is:" + RectangleArea); // Check the RectangleArea is true before using it again  
        int building = RectangleArea*4; //  
  
    }  
  
}
```



UNIT TESTING

- **The testing of single entity (class or method). Unit testing is very essential to every software company and developer.**



JUNIT

The screenshot displays an IDE interface with the following components:

- Package Explorer:** Shows a project named "DOP-Project1" with a test runner "JUnit". The test results summary indicates "Runs: 14/14", "Errors: 0", and "Failures: 0". A tree view shows the following test classes and methods:
 - UniversityTest [Runner: JUnit 4]
 - testAddStudent
 - testChangePhone
 - testGetStudents
 - testCreate
 - StudentTest [Runner: JUnit 4]
 - testID
 - testToString
 - testCreate
 - TestProject1 [Runner: JUnit 4] (0.002 s)
 - StudentTest (0.002 s)
 - UniversityTest (0.005 s)
- Code Editor:** Displays the source code for "UniversityTest.java":

```
import org.junit.*;

public class UniversityTest
{
    private University university;

    @Before
    public void runBeforeEachTest()
    {
        university = new University ("University of Computin");
    }

    @Test
    public void testCreate()
    {
        String name = university.getName();
        int numStudents = university.getNumStudents();

        assertEquals(name, "University of Computing");
        assertEquals(numStudents, 0);
    }
}
```
- Right Panel:** Includes a "Find" search bar, a "Connect My" notification, and a "Unive" tree view.
- Bottom Panel:** Contains tabs for "Problems", "Javadoc", "Declaration", and "Console".

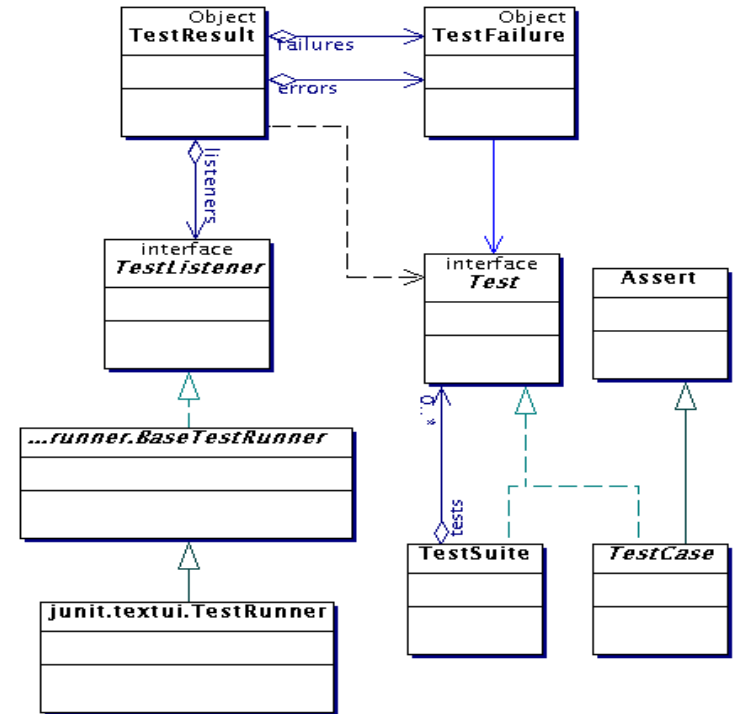


FEATURES OF JUNIT

- **It is an open source framework.**
- **Provides Annotation to identify the test methods.**
- **Provides Assertions for testing expected results.**
- **Provides Test runners for running tests.**
- **JUnit tests can be run automatically and they check their own results and provide immediate feedback.**
- **JUnit tests can be organized into test suites containing test cases and even other test suites.**
- **JUnit shows test progress in a bar that is green if test is going fine and it turns red when a test fails.**

ARCHITECTURAL OVERVIEW

- JUnit test framework is a package of classes that lets you write tests for each method, then easily run those tests
- TestRunner runs tests and reports TestResults
- You test your class by extending abstract class *TestCase*
- To write test cases, you need to know and understand the Assert class





EXPECTATIONS

- **Attend labs.**
- **Do the lab tutorials**
- **Practice the tools we learn in lab**



NEXT WEEK

- **Junit**