

# ITCS-2830 Applications Implementation & Testing

## Project #7 – Creating JUnit Unit Tests & TDD

### Objectives

Now that you have a solid grasp on **Eclipse, Java and JUnit**, this project will introduce you to writing *your own Java class AND JUnit unit tests*. Remember, practice makes perfect!

### Overview

In this project you are going to create JUnit unit tests for a class you will write from scratch. *Don't worry; you don't need a Ph.D. in Java for this assignment.* I will give you several hints along the way. You will write the class and it is up to you to design solid JUnit **passing** tests.

### Project 7

On Tuesday, June 4, 1996, the Ariane 5 rocket, flight 501, left the surface of the earth for the last time. Thirty seven seconds into the flight, the rocket veered off course and exploded under tremendous aerodynamic pressures and eventually self-terminated (see <http://goo.gl/Opxnq> for a video of the launch). The cost of the failure was estimated at \$370,000,000 (yes, millions) and is now known as one of the most expensive software bug in history.

[\[http://en.wikipedia.org/wiki/Ariane\\_5\\_Flight\\_501\]](http://en.wikipedia.org/wiki/Ariane_5_Flight_501)

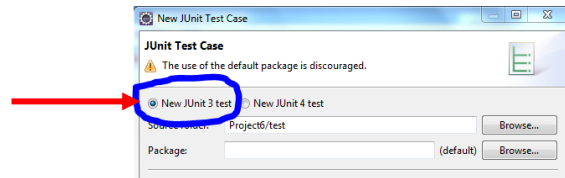
The problem arose when a 64-bit floating point number was converted to a 16-bit integer number. This caused an “integer overflow” [\[http://en.wikipedia.org/wiki/Integer\\_overflow\]](http://en.wikipedia.org/wiki/Integer_overflow) and the eventual crash of the rocket due to incorrect calculations. How did this happen? Engineers used software from the Ariane 4 (used smaller numbers) rocket which was not fast as the Ariane 5 rocket. This “conversion” from a larger number to a smaller number is what caused the rocket to explode. The original software was written in ADA but we will attempt to simulate the Ariane 5 rocket integer overflow using Java (the concept is the same). We will use simple JUnit tests to catch the integer overflow error (is that NASA calling?).

*When you develop the unit tests, try your hand at TDD by creating the unit tests first, running the failing test, developing the class code, and then creating the passing test.*

Your task is to write the source code (GuidanceUnit.java), and design, implement, and execute passing JUnit unit tests for the GuidanceUnit.java. Create a new project in Eclipse called **“Project 7”**. Create a class called **“GuidanceUnit.java”** in your **“src”** directory. Create a new directory called **“test”** to hold your JUnit **VERSION 3** (not **VERSION 4**) unit tests. *When you create the JUnit test, don't forget to check the box for Version 3.* Let's call this test class **“GuidanceUnit Test.java”**.

# ITCS-2830 Applications Implementation & Testing

## Project #7 – Creating JUnit Unit Tests & TDD



- A. Create the two .java files listed above.
- B. Use the *appropriate assert* methods to test your class. [Note: you may want to reference [previous videos you watched for hints and tips on how to write those tests.](#)]
- C. You are going to create four (4) test methods. [Note: When testing your values you may need to cast the double to a short using **(short)**]
  - a. `testGuidanceUnitAtLaunch()`
  - b. `testGuidanceUnitJustBeforeIntegerOverflow()`
  - c. `testGuidanceUnitOverflowByOne()`
  - d. `testGuidanceUnitMaxHorizontalAcceleration()`
- D. For `testGuidanceUnitAtLaunch()`, test for a value that asserts to “0”. This is when the rocket is sitting still on the launch pad.
- E. For `testGuidanceUnitJustBeforeIntegerOverflow()`, test for the condition just before a short value will overflow, 32,767. [The short data type is a 16-bit signed two's complement integer. It has a minimum value of -32,768 and a maximum value of 32,767 (inclusive)] <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>
- F. For `testGuidanceUnitOverflowByOne()`, test for one over the maximum value that a short can contain, 32,768. Make this test fail by asserting the number is this value (not the overflow condition) is true.
- G. For `testGuidanceUnitMaxHorizontalAcceleration()`, let's assume some large random number to test for the Ariane 5 rocket much larger than the short, say 999,999. Make the test fail by asserting this number should be true.
- H. **Your goal here is to catch the failing tests for both F and G above.** Tests D and E should pass because these values fall in the range of a short. Number values outside this range should fail.

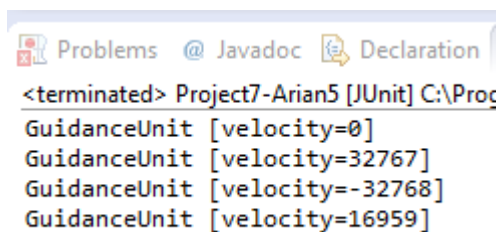
# ITCS-2830 Applications Implementation & Testing

## Project #7 – Creating JUnit Unit Tests & TDD

I. Now, on to the **GuidanceUnit.java** class. Use the source code below.

```
public class GuidanceUnit {  
    private short horizontalAcceleration = 0;  
  
    public void setHorizontalAcceleration(short vel) {  
        horizontalAcceleration = vel;  
    }  
  
    public short getHorizontalAcceleration() {  
        return horizontalAcceleration;  
    }  
  
    @Override  
    public String toString() {  
        return "GuidanceUnit [velocity=" + horizontalAcceleration + "];"  
    }  
}
```

J. The output will look similar to the screenshot below.



The screenshot shows a Java IDE console window with the following output:

```
<terminated> Project7-Arian5 [JUnit] C:\Pro  
GuidanceUnit [velocity=0]  
GuidanceUnit [velocity=32767]  
GuidanceUnit [velocity=-32768]  
GuidanceUnit [velocity=16959]
```

K. Did you get the failing tests? If so, congratulations are in order. You just saved our country \$320 million.



You are now complete with Project #7.

## **Deliverables**

Locate the workspace for your project.

(1) Take a screenshot of your **FAILING** JUnit tests. (.png please)

# ITCS-2830 Applications Implementation & Testing

## Project #7 – Creating JUnit Unit Tests & TDD

(2) Zip your entire **Project7**. A total of TWO separate files (1 - .png screenshot, and 1 – zipped Eclipse project) (no .rar files please, only .zip)

### Check List

1 - .png screenshot of Deliverable (1).....☐

2 – zip your Eclipse Project7 (2).....☐

Upload your **TWO** project *files* to the dropbox for **Project #7**.

*Please* do not zip the screenshots with the zipped Eclipse project – you will have FOUR files to upload into the dropbox.