CT3535 Assignment 2

University Class :

```
import java.util.ArrayList;

public class University {
    private ArrayList<Venue> venues;
    private ArrayList<Modules> modules;
    private ArrayList<Programmes> programmes;

    //Venue subclass
    class Venue{
        //Declaring variables
        private String name;
        private int capacity;

        //Venue Class costructor
        public Venue(String VenName, int Amount) {
            name = VenName;
            capacity = Amount;
        }

        //Accessors
        public String getName(){
            return name;
        }

        public int getCapacity() {
            return capacity;
        }

        //toString method
        public String toString() {
            return "Name: " + name + "\n" + "Capacity: " + capacity;
        }

    }
    //Subclass end

    public Venue newVenue(String name, int capacity){
        return new Venue(name, capacity);
    }

    //Constructor

    public University() {
        venues = new ArrayList<Venue>();
        modules = new ArrayList<Modules>();
        programmes = new ArrayList<Programmes>();
    }

    //addVenues function, which adds a venue to the list venues
    public void addVenues(Venue place) {
        venues.add(place);
    }

    //removeVenues function, removes a venue from the venues list
    public void removeVenues(Venue place) {
        venues.remove(place);
    }

    //getVenues function, returns the venues list
    public ArrayList<Venue> getVenues(){
        return venues;
    }

    //addModules, adds a module to the modules list
    public void addModules (Modules mod) {
        modules.add(mod);
    }

    //removeModules, remove a module from the module list
    public void removeModules (Modules mod) {
        modules.remove(mod);
    }

    //getModules, returns modules list
    public ArrayList<Modules> getModules(){
        return modules;
    }

    //addProgrammes, adds a program to the programmes list
    public void addProgrammes(Programmes prog){
        programmes.add(prog);
    }

    //removeProgrammes, removes a program from the programmes list
    public void removeProgrammes(Programmes prog) {
        programmes.remove(prog);
    }

    //getProgrammes, returns programmes list
    public ArrayList<Programmes> getProgrammes(){
        return programmes;
    }

    //toString method
    public String toString() {
        return "Venues: " + getVenues() + "\n" + "Modules: " + getModules()
+ "\n" + "Programmes: " + getProgrammes() + "\n";
    }
}//University end
```

Modules Class :

```
public class Modules{
      //Variables
      private String name;
      private String code;

      //Constructor
      public Modules(String ModName, String ModCode) {
            name = ModName;
            code = ModCode;
      }

      //Accessors
      public String getName() {
            return name;
```

```
    }

    public String getCode() {
        return code;
    }

    //toString method
    public String toString() {
        return "Name: " + name + "\n" + "Code: " + code;
    }
}
```

Programmes Class:

```
public class Programmes {
    //Initialising variables
    private String title;
    private String code;

    //constructor
    public Programmes(String ProgName, String ProgCode) {
        title = ProgName;
        code = ProgCode;
    }

    //Accessor's
    public String getTitle() {
        return title;
    }

    public String getCode() {
        return code;
    }

    //toString method
    public String toString() {
        return "Title: " + title + "\n" + "Code: " + code;
    }
}
```

JUnit Test Class :

```
import static org.junit.Assert.*;

import org.junit.After;
import org.junit.Before;
import org.junit.Ignore;
import org.junit.Test;

public class UniversityTest {

    //variable setup
    University u;
    Modules m;
    Programmes p;
    University.Venue v;

    @Ignore
    @Test
    public void test() {
        fail("Not yet implemented");
    }
```

```
//Before
@Before
public void setUp() {
    u = new University();
    m = new Modules("OOP", "CT3535");
    p = new Programmes("Bachelor of Science", "GY301");
    v = u.newVenue("Theatre", 50);
}

//Constructor and accessors test
@Test
public void testConstructor() {
    assertNotNull(u);
    assertEquals(0, u.getModules().size());
    assertEquals(0, u.getVenues().size());
    assertEquals(0, u.getProgrammes().size());
}

@Test
public void testVenues() {
    //testing add function
    u.addVenues(v);
    assertEquals(1, u.getVenues().size());

    //testing accessors
    assertEquals("Theatre", v.getName());
    assertEquals(50, v.getCapacity());

    //testing remove function
    u.removeVenues(v);
    assertEquals(0, u.getVenues().size());
}

@Test
public void testModules(){
    //testing addfunction
    u.addModules(m);
    assertEquals(1, u.getModules().size());

    //testing accessors
    assertEquals("OOP", m.getName());
    assertEquals("CT3535", m.getCode());

    //testing remove function
    u.removeModules(m);
    assertEquals(0, u.getModules().size());
}
@Test
public void testProgrammes() {
    u.addProgrammes(p);
    assertEquals(1, u.getProgrammes().size());

    //testing accessors
    assertEquals("Bachelor of Science", p.getTitle());
    assertEquals("GY301" ,p.getCode());

    //testing remove function
    u.removeProgrammes(p);
    assertEquals(0, u.getProgrammes().size());
}
```

```
    //ToString Test
    @Test
    public void testToString(){
        assertEquals(u.toString(), "Venues: " + "[]" + "\n" + "Modules: " +
"[]" + "\n" + "Programmes: " + "[]" + "\n");
    }
    //After
    @After
    public void tearDown() {
        u = null;
        m = null;
        p = null;
        v = null;
    }


}
```

JUnit Test Class :

```
import static org.junit.Assert.*;

import org.junit.After;
import org.junit.Before;
import org.junit.Ignore;
import org.junit.Test;

public class UniversityTest {

    //variable setup
    University u;
    Modules m;
    Programmes p;
    University.Venue v;

    @Ignore
    @Test
    public void test() {
        fail("Not yet implemented");
    }

    //Before
    @Before
    public void setUp() {
        u = new University();
        m = new Modules("OOP", "CT3535");
        p = new Programmes("Bachelor of Science", "GY301");
        v = u.newVenue("Theatre", 50);
    }

    //Constructor and accessors test
    @Test
    public void testConstructor() {
        assertNotNull(u);
        assertEquals(0, u.getModules().size());
        assertEquals(0, u.getVenues().size());
        assertEquals(0, u.getProgrammes().size());
    }

    @Test
    public void testVenues() {
        //testing add function
        u.addVenues(v);
        assertEquals(1, u.getVenues().size());

        //testing accessors
        assertEquals("Theatre", v.getName());
        assertEquals(50, v.getCapacity());

        //testing remove function
        u.removeVenues(v);
        assertEquals(0, u.getVenues().size());
    }

    @Test
    public void testModules(){
        //testing addfunction
        u.addModules(m);
        assertEquals(1, u.getModules().size());

        //testing accessors
        assertEquals("OOP", m.getName());
        assertEquals("CT3535", m.getCode());

        //testing remove function
        u.removeModules(m);
        assertEquals(0, u.getModules().size());
    }
    @Test
    public void testProgrammes() {
        u.addProgrammes(p);
        assertEquals(1, u.getProgrammes().size());

        //testing accessors
        assertEquals("Bachelor of Science", p.getTitle());
        assertEquals("GY301" ,p.getCode());

        //testing remove function
        u.removeProgrammes(p);
        assertEquals(0, u.getProgrammes().size());
    }

    //ToString Test
    @Test
    public void testToString(){
        assertEquals(u.toString(), "Venues: " + "[]" + "\n" + "Modules: " +
"[]" + "\n" + "Programmes: " + "[]" + "\n");
    }
    //After
    @After
    public void tearDown() {
        u = null;
        m = null;
        p = null;
        v = null;
    }


}
```

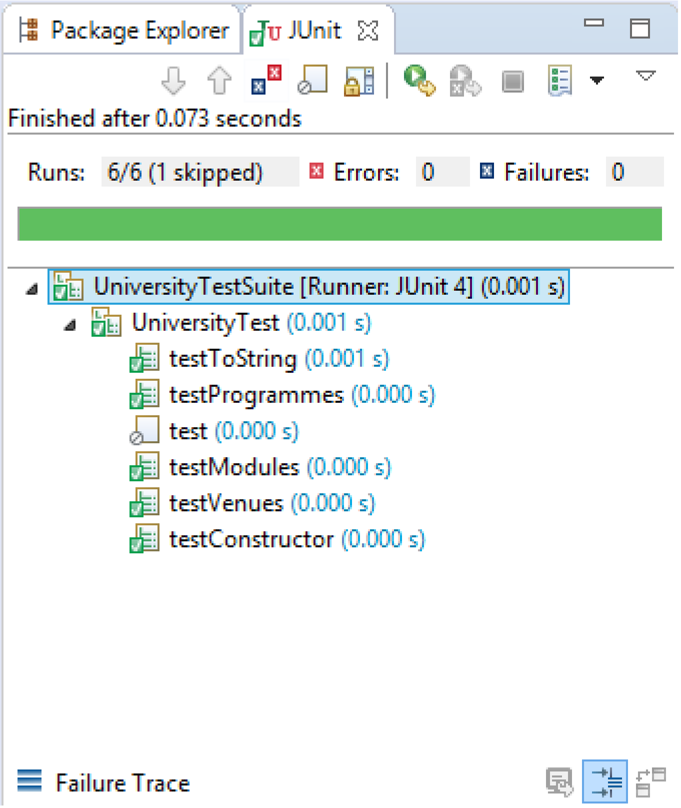
JUnit Test Suite :

```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({ UniversityTest.class })
public class UniversityTestSuite {

}
```

JUnit Suite results:



```
Finished after 0.073 seconds

Runs: 6/6 (1 skipped)   Errors: 0   Failures: 0

UniversityTestSuite [Runner: JUnit 4] (0.001 s)
  UniversityTest (0.001 s)
    testToString (0.001 s)
    testProgrammes (0.000 s)
    test (0.000 s)
    testModules (0.000 s)
    testVenues (0.000 s)
    testConstructor (0.000 s)
```