# Homework 1 - Vector - Self Evaluation

It took me five hours to complete this homework.

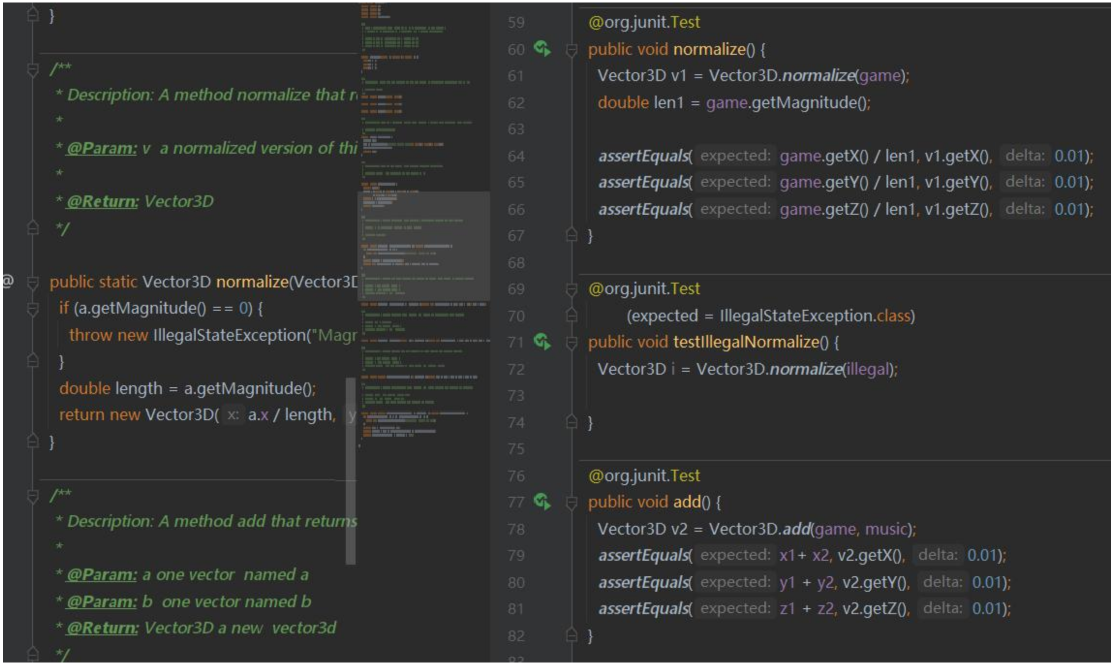
The test error I've got was : *method X in class Y can not be applied to given types.* It was caused by using static keyword in my code. From this error I learn more about *the static method,* which is more important is that I had a deeper understanding of the Object-Oriented Design.

*Static methods and variables* are part of the class, not the instances of the class. A static method can access only static variables of class and invoke only static methods of the class. Usually, static methods are utility methods that we want to expose to be used by other classes without the need of creating an instance.

*The object method* belongs to a method of a class. Before calling it, we need to create an instance of the class and call it through the instance of the class.

Here are some screenshots of this homework.

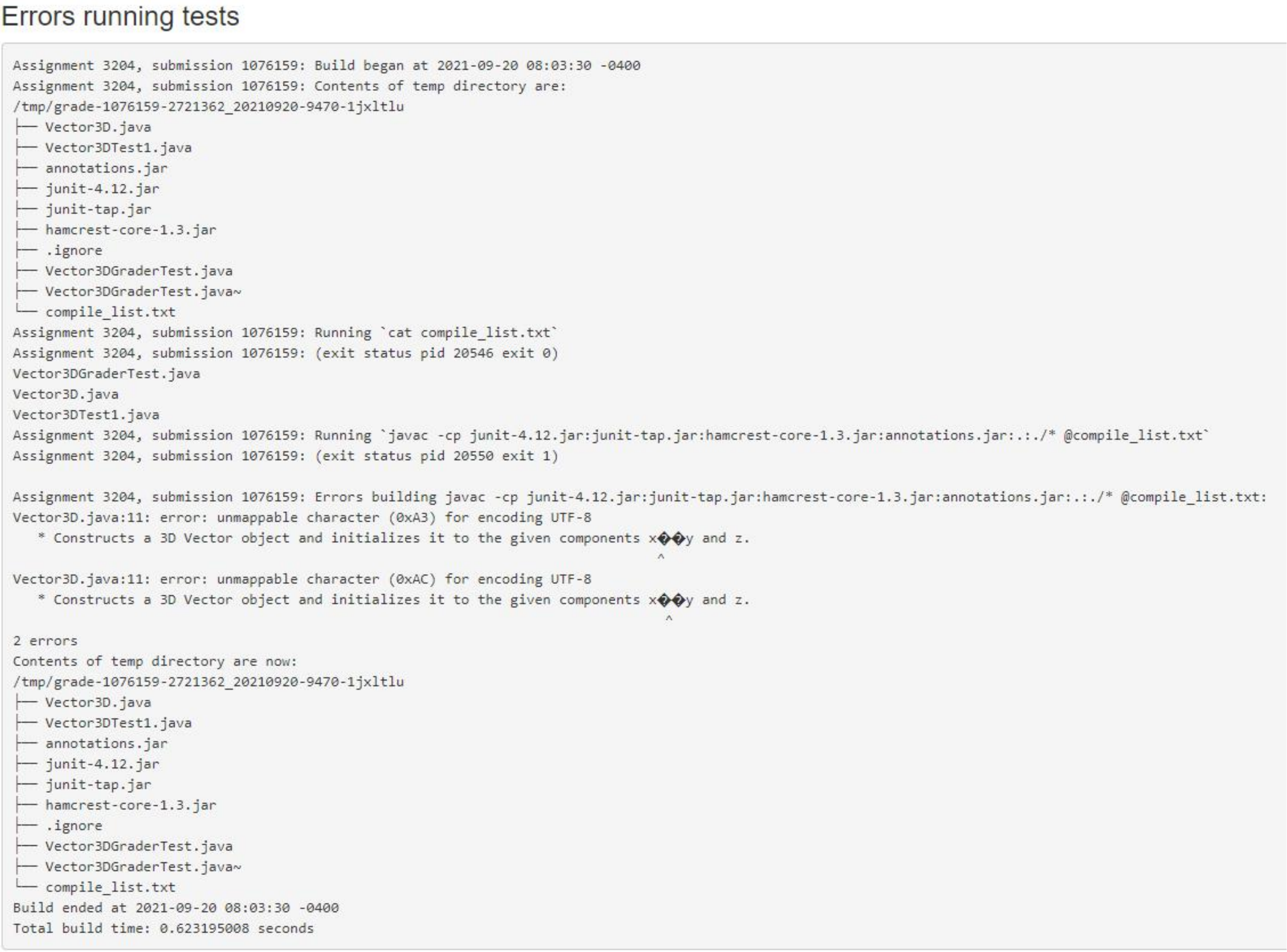
*Primary code with static method*



```
}

/**
 * Description: A method normalize that r
 *
 * @Param: v  a normalized version of thi
 *
 * @Return: Vector3D
 */

public static Vector3D normalize(Vector3D
  if (a.getMagnitude() == 0) {
    throw new IllegalStateException("Magr
  }
  double length = a.getMagnitude();
  return new Vector3D( x: a.x / length,  y
}

/**
 * Description: A method add that returns
 *
 * @Param: a one vector  named a
 * @Param: b  one vector named b
 * @Return: Vector3D a new  vector3d
 */
```

```
@org.junit.Test
public void normalize() {
  Vector3D v1 = Vector3D.normalize(game);
  double len1 = game.getMagnitude();

  assertEquals( expected: game.getX() / len1, v1.getX(),  delta: 0.01);
  assertEquals( expected: game.getY() / len1, v1.getY(),  delta: 0.01);
  assertEquals( expected: game.getZ() / len1, v1.getZ(),  delta: 0.01);
}

@org.junit.Test
      (expected = IllegalStateException.class)
public void testIllegalNormalize() {
  Vector3D i = Vector3D.normalize(illegal);


}

@org.junit.Test
public void add() {
  Vector3D v2 = Vector3D.add(game, music);
  assertEquals( expected: x1+ x2, v2.getX(),  delta: 0.01);
  assertEquals( expected: y1 + y2, v2.getY(),  delta: 0.01);
  assertEquals( expected: z1 + z2, v2.getZ(),  delta: 0.01);
}
```

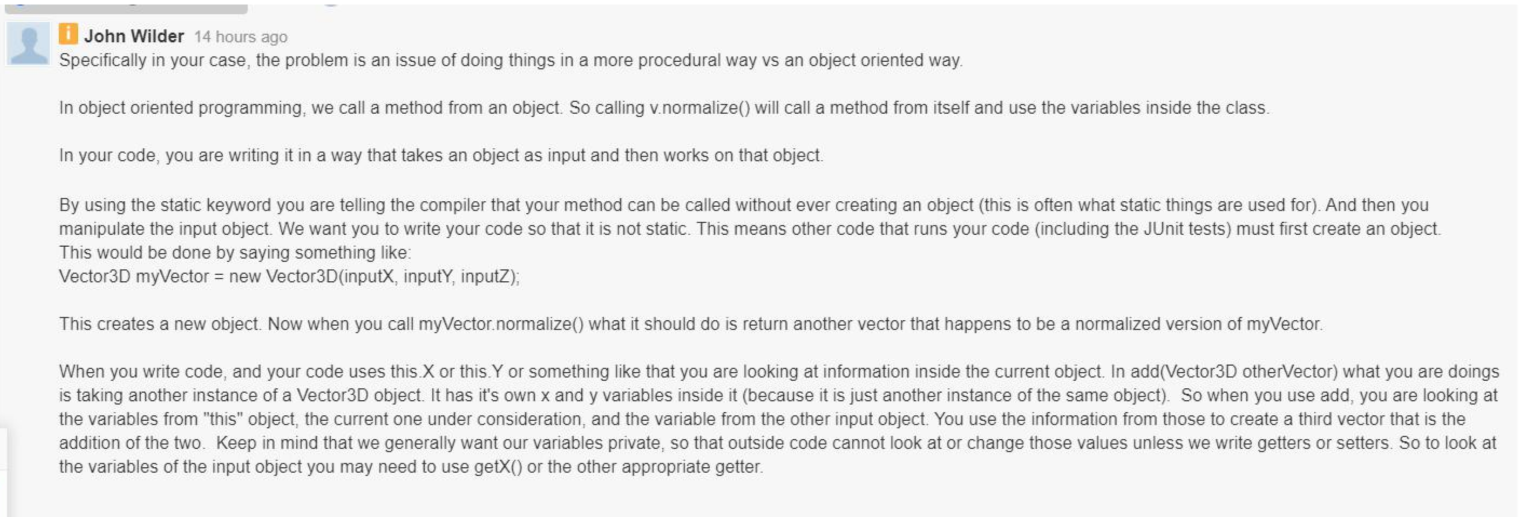
*The error*



## Errors running tests

```
Assignment 3204, submission 1076159: Build began at 2021-09-20 08:03:30 -0400
Assignment 3204, submission 1076159: Contents of temp directory are:
/tmp/grade-1076159-2721362_20210920-9470-1jxltlu
├── Vector3D.java
├── Vector3DTest1.java
├── annotations.jar
├── junit-4.12.jar
├── junit-tap.jar
├── hamcrest-core-1.3.jar
├── .ignore
├── Vector3DGraderTest.java
├── Vector3DGraderTest.java~
└── compile_list.txt
Assignment 3204, submission 1076159: Running `cat compile_list.txt`
Assignment 3204, submission 1076159: (exit status pid 20546 exit 0)
Vector3DGraderTest.java
Vector3D.java
Vector3DTest1.java
Assignment 3204, submission 1076159: Running `javac -cp junit-4.12.jar:junit-tap.jar:hamcrest-core-1.3.jar:annotations.jar:.:./* @compile_list.txt`
Assignment 3204, submission 1076159: (exit status pid 20550 exit 1)

Assignment 3204, submission 1076159: Errors building javac -cp junit-4.12.jar:junit-tap.jar:hamcrest-core-1.3.jar:annotations.jar:.:./* @compile_list.txt:
Vector3D.java:11: error: unmappable character (0xA3) for encoding UTF-8
   * Constructs a 3D Vector object and initializes it to the given components x��y and z.
                                                                               ^

Vector3D.java:11: error: unmappable character (0xAC) for encoding UTF-8
   * Constructs a 3D Vector object and initializes it to the given components x��y and z.
                                                                                ^

2 errors
Contents of temp directory are now:
/tmp/grade-1076159-2721362_20210920-9470-1jxltlu
├── Vector3D.java
├── Vector3DTest1.java
├── annotations.jar
├── junit-4.12.jar
├── junit-tap.jar
├── hamcrest-core-1.3.jar
├── .ignore
├── Vector3DGraderTest.java
├── Vector3DGraderTest.java~
└── compile_list.txt
Build ended at 2021-09-20 08:03:30 -0400
Total build time: 0.623195008 seconds
```

*The instructor's answer targeted to above*



**John Wilder** 14 hours ago

Specifically in your case, the problem is an issue of doing things in a more procedural way vs an object oriented way.

In object oriented programming, we call a method from an object. So calling v.normalize() will call a method from itself and use the variables inside the class.

In your code, you are writing it in a way that takes an object as input and then works on that object.

By using the static keyword you are telling the compiler that your method can be called without ever creating an object (this is often what static things are used for). And then you manipulate the input object. We want you to write your code so that it is not static. This means other code that runs your code (including the JUnit tests) must first create an object.
This would be done by saying something like:
Vector3D myVector = new Vector3D(inputX, inputY, inputZ);

This creates a new object. Now when you call myVector.normalize() what it should do is return another vector that happens to be a normalized version of myVector.

When you write code, and your code uses this.X or this.Y or something like that you are looking at information inside the current object. In add(Vector3D otherVector) what you are doings is taking another instance of a Vector3D object. It has it's own x and y variables inside it (because it is just another instance of the same object). So when you use add, you are looking at the variables from "this" object, the current one under consideration, and the variable from the other input object. You use the information from those to create a third vector that is the addition of the two. Keep in mind that we generally want our variables private, so that outside code cannot look at or change those values unless we write getters or setters. So to look at the variables of the input object you may need to use getX() or the other appropriate getter.