**ID Number: 18023355**

Answer 1:

**1a.**

public class Point {

final double x;

final double y;

final double z;

public Point(double x, double y, double z) {

this.x = x;

this.y = y;

this.z = z;

}

double distance(Point p2){

double dx = this.x - p2.x;

double dy = this.y - p2.y;

double dz = this.z - p2.z;

return Math.sqrt(dx\*dx + dy\*dy + dz\*dz);

}

}

**1b.**

/\*\*

\* Test of distance method, of class Point.

\*/

@Test

public void testPointDistance() {

System.out.println("Point distance test");

Point p1 = new Point(8, 2, 6);

Point p2 = new Point(8, 6, 3);

assertTrue(Double.compare(5, p1.distance(p2)) == 0);

assertTrue(Double.compare(5, p2.distance(p1)) == 0);

}

**Answer 2:**

2a.

**Coupling** in the context of this moduleis a measure of how much a package, class and method relies on other modules. It is desirable to reduce coupling, or reduce the amount that a given module relies on the other modules of a system.

2b.

**Extend an observer.**

import java.util.Observable;

public class ObservableValue extends Observable

{

private int n = 0;

public ObservableValue(int n)

{

this.n = n;

}

public void setValue(int n)

{

this.n = n;

setChanged();

notifyObservers();

}

public int getValue()

{

return n;

}

}

**Implement an observer**

import java.util.Observer;

import java.util.Observable;

public class TextObserver implements Observer

{

private ObservableValue ov = null;

public TextObserver(ObservableValue ov)

{

this.ov = ov;

}

public void update(Observable obs, Object obj)

{

if (obs == ov)

{

System.out.println(ov.getValue());

}

}

}

**Finally, Tie both together.**

public class Main

{

public Main()

{

ObservableValue ov = new ObservableValue(0);

TextObserver to = new TextObserver(ov);

ov.addObserver(to);

}

public static void main(String [] args)

{

Main m = new Main();

}

}

2c. I disagree with the statement that the observer design pattern reduces coupling between the observer and observed classes because observed classes depends too much on the observer class and without the observer class the observed class would struggle to work.