****

**ASSIGNMENT 1**

**SOFTWARE ENGINEERING: PROCESS AND TOOLS**

**PRT 452**

**SUBMITTED BY-**

**AMAN KATHED  
S301828**

**MITSE**

**SOLUTION 1:**

I created a Github repository for this assignment with the name “[**ASSIGNMENT-1-PRT452**](https://github.com/AmanKanthed/ASSIGNMENT-1-PRT452/upload/master/)”. The screen shot of that repository is as follows:

**Solution 2:**

**CODE:**

package kanthed;

import java.io.\*;

import java.util.\*;

//Directed graph using adjacency

public class CycleGraph

{

private int V; // No. of vertices

private LinkedList<Integer> adj[]; //Adjacency List

public CycleGraph(int v)

{

V = v;

adj = new LinkedList[v];

for (int i=0; i<v; ++i)

adj[i] = new LinkedList();

}

// Edge into the graph

public void addEdge(int v,int w)

{

adj[v].add(w);

adj[w].add(v);

}

// Function that uses visited[] reachable from vertex v.

public Boolean isRechable(int v, Boolean visited[], int parent)

{

// Visited Node

visited[v] = true;

Integer i;

// Vertices adjacent to the vertex

Iterator<Integer> it = adj[v].iterator();

while (it.hasNext())

{

i = it.next();

// If an adjacent is not visited, then recur for

// that adjacent

if (!visited[i])

{

if (isRechable(i, visited, v))

return true;

}

// If an adjacent is visited

else if (i != parent)

return true;

}

return false;

}

// Returns true if the graph is connected.

public Boolean connected()

{

// Mark as not visited

Boolean visited[] = new Boolean[V];

for (int i = 0; i < V; i++)

visited[i] = false;

// Marks all vertices connected from 0.

if (isRechable(0, visited, -1))

return false;

// Vertex if not reachable from 0, return false

for (int u = 0; u < V; u++)

if (!visited[u])

return false;

return true;

}

}

(Coursehero.com, 2017)

**Junit Testing:**

import kanthed.CycleGraph;

import org.junit.Test;

public class Check {

@Test

public void test() {

// CycleGraph test1 = new CycleGraph(2); TEST FAILS

CycleGraph test1 = new CycleGraph(5);

test1.addEdge(1, 0);

test1.addEdge(1, 2);

test1.addEdge(2, 3);

test1.addEdge(3, 4);

if (test1.connected())

System.out.println("Connected");

else

System.out.println("Not Connected");

// CycleGraph test2 = new CycleGraph(3); TEST FAILS

CycleGraph test2 = new CycleGraph(6);

test2.addEdge(1, 0);

test2.addEdge(1, 2);

test2.addEdge(0, 3);

test2.addEdge(0, 4);

test2.addEdge(3, 4);

if (test2.connected())

System.out.println("Connected");

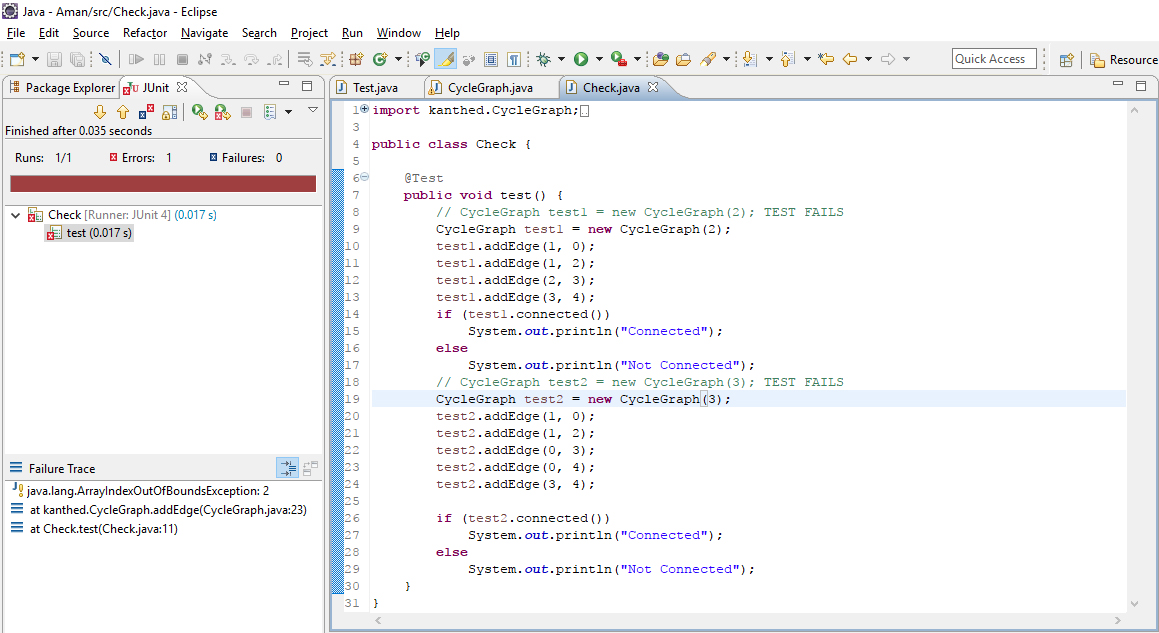
else

System.out.println("Not Connected");

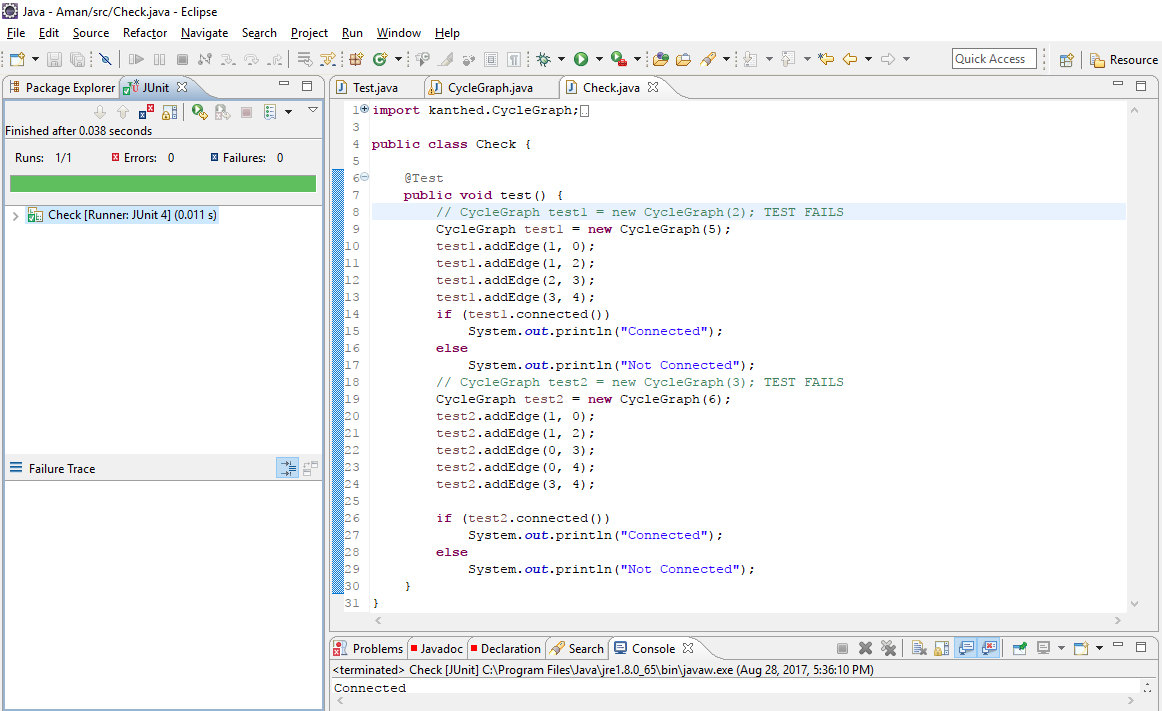
}

}

**PASSED JUNIT TEST SCREENSHOT:**

****

**FAILED JUNIT TEST SCREENSHOT:**

****

**Solution 3:**

**SWITCH:**

**public** **static** **boolean** contentEquals(**final** File file1, **final** File file2) **throws** IOException {

**final** **boolean** file1Exists = file1.exists();

**if** (file1Exists != file2.exists()) {

**return** **false**;

}

**if** (!file1Exists) {

// two not existing files are equal

**return** **true**;

}

**if** (file1.isDirectory() || file2.isDirectory()) {

// don't want to compare directory contents

**throw** **new** IOException("Can't compare directories, only files");

}

**if** (file1.length() != file2.length()) {

// lengths differ, cannot be equal

**return** **false**;

}

**if** (file1.getCanonicalFile().equals(file2.getCanonicalFile())) {

// same file

**return** **true**;

}

**try** (InputStream input1 = **new** FileInputStream(file1);

InputStream input2 = **new** FileInputStream(file2)) {

**return** IOUtils.contentEquals(input1, input2);

}

}

(GitHub, 2017)

**MIDDLE MAN:**

**Example:**

class Y{

}

public class YL {

Vector<Y> y;

String name;

public YL(String name) {

y = new Vector<y>();

this.name = name;

}

public void addY(String name, int a, double b,

double c, String a, String g) {

}

public Patient findY(String name) {

}

public void removeY(String name) {

}

public void printSummaryForY(String name) {

}

}

class Main{

public static void main(String[] args){

YL yl=new PL();

yl.addY(a,b,c,d,e);

yl.addY(a,b,c,d,e);

yl.remove();

}

}

(smell?, 2017)

**References:**

* Coursehero.com. (2017). *Graph.java - package com.amey import java.util.Iterator import java.util.LinkedList/This class represents a directed graph using adjacency list*. [online] Available at: https://www.coursehero.com/file/22091303/Graphjava/ [Accessed 22 Aug. 2017].
* GitHub. (2017). *JeanPierrePortier/commons-io*. [online] Available at: https://github.com/JeanPierrePortier/commons-io/blob/master/src/main/java/org/apache/commons/io/FileUtils.java [Accessed 26 Aug. 2017].
* smell?, D. (2017). *Does this class count under "Middle Man" code smell?*. [online] Stackoverflow.com. Available at: https://stackoverflow.com/questions/8158124/does-this-class-count-under-middle-man-code-smell [Accessed 25 Aug. 2017].