**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

****

**INFORMATIKOS FAKULTETAS**

**PROGRAMŲ SISTEMŲ TESTAVIMAS**

**Studentas: Lukas Gužauskas**

**Dėstytojas: Dominykas Barisas**

**KAUNAS 2018-2019**

# Lab 3. Static testing

Užduoties tikslas:

* Įvertinkite pasirinktą programinės įrangos kokybę, naudodami statinio analizės metodus ir įrankius.

Testavimo įrankiai:

Šioje dalyje naudojamas ankstesnės laboratorinės užduoties algoritmas. Šiam testavimui naudojamas „Find Bugs“ testavimo įrankis. Testavimo tema: Statinės kodo analizė, klaidų aptikimas ir pataisymas, ir statinės analizės taisyklių papildymas.

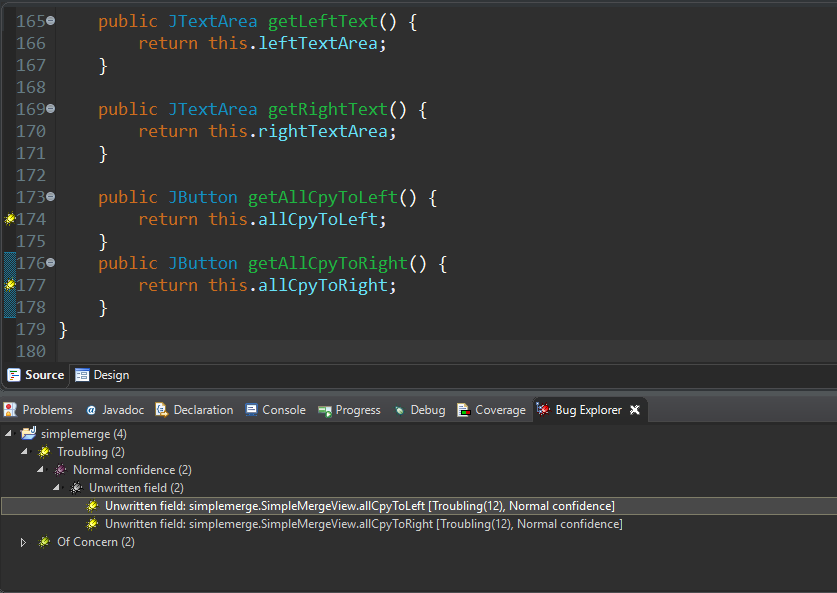
# Kodo peržiūros kontrolinis sąrašas:

1. Struktūra
2. Dokumentacija
3. Kintamieji

# Statinė analizė naudojant Find Bugs įrankius:

Analizuojamas savos programos kodas, panaudojant FindBugs įrankius. Kodo redaktorius aptiko 4 programinės įrangos netikslumus.

# Kodo redaktoriaus pastebėjimas 01:



Pastebėjimas:

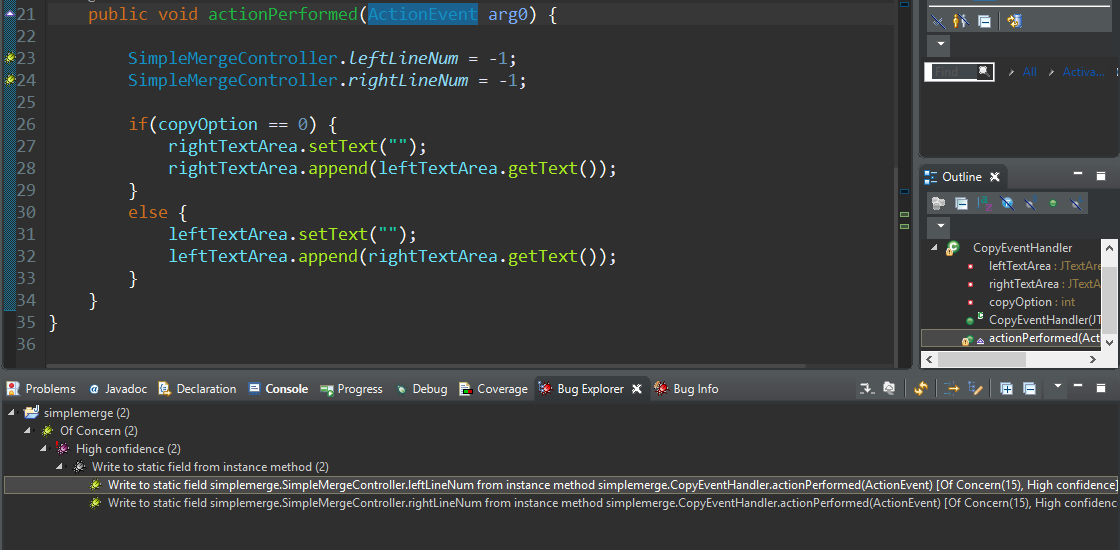
Ten yra 2 redaktoriaus pastebėjimai – 174 ir 177 kodo eilutės, priežastis yra šie laukai niekada nėra parašyti.

Pataisymas:

Patikrinta, ar nėra klaidų, jei pastebėjimas buvo inicijuotas. Netikslumo pašalinimui reikia įdėti papildomus kodus ant JButton konstruktoriaus kaip parodyta paveiksle apačioje:



# Kodo redaktoriaus pastebėjimas 02:



Pastebėjimas:

Ten yra 2 redaktoriaus pastebėjimai – 23 ir 24 eilutės, kad parašyti į statinį lauką Controller klasėje šio metodo atveju. Tai yra sudėtinga teisingai išspręsti, kai manipuliuojama keliais metodo atvejais ir apskritai bloga praktika.

Pataisymas:

Buvo sukurti 2 statiniai set metodai vietoj Controller klasės. Tada pašalinamos 2 nuorodos kintamųjų ir įvedami kiekiai į argumento vietą metodo, kuris gali įsiųsti į Controller klasę. Tuomet nelieka pranešimo apie tokią klaidą.



Pradzia gera. Svarbiausias punktas dabar parasyti savo taisykle. Jei FIndBugs irankiui nepavytu sukurti taisykles, tada paieskoti kito irankio. Pvz galbut PMD iranki. tinka

# Statinės analizės taisyklė

Mano taisyklė:

Sukūriau savo statinės analizės taisyklę, kuris vadinasi „cDetector“ ir pasirinkau statinės analizės įrankį – „SpotBugs“ (buvęs FindBugs). Mano taisyklės tikslas yra savo programos kodas, kuris suranda kintamuosius, kurių pavadinimas neatitinka taisyklės. Taisyklė apibrėžia, kad *final* tipo kintamojo pavadinimas turi prasidėti raide *c*. Tada programa pabraukia geltonai ir pasiūlo pakeisti, pvz.: public int skaicius; -> public int cSkaicius.

Rezultatas:

Programų kodas:

package detector;

import org.apache.bcel.classfile.JavaClass;

import org.apache.bcel.classfile.Field;

import edu.umd.cs.findbugs.BugInstance;

import edu.umd.cs.findbugs.BugReporter;

import edu.umd.cs.findbugs.bcel.OpcodeStackDetector;

public class SimpleMergeDetector extends OpcodeStackDetector {

    final private BugReporter reporter;

    private String currentClass;

    public SimpleMergeDetector(BugReporter reporter) {

        this.reporter = reporter;

    }

    public void visit(JavaClass obj) {

        currentClass = obj.getClassName();

        for ( Field field : obj.getFields()) {

            if (field.isFinal()) {

                String fieldName = "c" + field.getName();

                boolean found = false;

                for (Field cField : obj.getFields()) {

                    if (cField.getName().equals(fieldName)) {

                        found = true;

                        break;

                    }

                }

                if (false == found)

                    reporter.reportBug(new BugInstance(this, "UNUSED\_LOCAL\_VARIABLES\_BUG", NORMAL\_PRIORITY).addClass(currentClass)

                            .addField(this).addString("Change c").addSourceLine(this));

            }

        }

        super.visit(obj);

    }

    @Override

    public void sawOpcode(int arg0) {

    }

}