EXERCISES 3

CKJM

Java code for calculate Chidamber and Kemerer Java Metrics

Repository: https://github.com/dspinellis/ckjm

Dependency: https://mvnrepository.com/artifact/gr.spinellis.ckjm/ckjm_ext

```
package vn.edu.iuh.fit.ckjm;


import java.io.File;
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.atomic.AtomicReference;


import gr.spinellis.ckjm.CkjmOutputHandler;
import gr.spinellis.ckjm.ClassMetrics;
import gr.spinellis.ckjm.MetricsFilter;

public class MetricsFilterTest {


    public void test() throws Exception {
       final CountDownLatch latch = new CountDownLatch(1);
       final AtomicReference<ClassMetrics> ref = new AtomicReference<>();
       CkjmOutputHandler outputHandler = new CkjmOutputHandler() {
          @Override
          public void handleClass(String name, ClassMetrics c) {
             System.out.println("name: " + name + ", WMC: " + c.getWmc());
             System.out.println("name: " + name + ", LCOM: " + c.getLcom());
//           System.out.println("name: " + name + ", LCOM: " + c.*******());
             ref.set(c);
             latch.countDown();
          }
       };
       File f = new
File("build/classes/java/main/vn/edu/iuh/fit/ckjm/MetricsFilterTest.class");
       MetricsFilter.runMetrics(new String[] { f.getAbsolutePath() },
outputHandler, false);
       latch.await(1, TimeUnit.SECONDS);
    }

    public static void main(String[] args)throws Exception {
       new MetricsFilterTest().test();
    }
}
```

BCEL

Determining the LCOM4 (Lack of Cohesion in Methods) by parsing the Java Bytecode using BCEL (Byte Code Engineering Library)

The dependency of BCEL: https://mvnrepository.com/artifact/org.apache.bcel/bcel

Reference: *https://stackoverflow.com/questions/44040918/determining-the-lcom4-lack-of-cohesion-in-methods-by-parsing-the-java-bytecode*

```java
package vn.edu.iuh.fit;

import java.util.HashSet;
import java.util.Set;

public class Group {
    private final Set<String> fields = new HashSet<>();
    private final Set<String> methods = new HashSet<>();

    public Group addFields(String...fields) {
        for (String field: fields) {
            this.fields.add(field);
        }
        return this;
    }

    public Group addMethods(String... methods) {
        for (String method: methods) {
            this.methods.add(method);
        }
        return this;
    }

    public int fields() {
        return fields.size();
    }

    public int methods() {
        return methods.size();
    }

    public boolean intersects(Group other) {
        for (String field: other.fields) {
            if (fields.contains(field)) {
                return true;
            }
        }
        for (String method: other.methods) {
            if (methods.contains(method)) {
                return true;
            }
        }
        return false;
    }

    public void merge(Group other) {
        fields.addAll(other.fields);
        methods.addAll(other.methods);
    }
    @Override
    public String toString() {
        return "Group{" + "fields=" + fields + ", methods=" + methods + '}';
    }
}
```

```
package vn.edu.iuh.fit;

import org.apache.bcel.classfile.*;
import org.apache.bcel.generic.*;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

public class LCOM4Calculation {
    public List<Group> loadGroups(File file) throws IOException {
        try (InputStream in = new FileInputStream(file)) {
            ClassParser parser = new ClassParser(in, file.getName());
            JavaClass clazz = parser.parse();
            String className = clazz.getClassName();
            ConstantPoolGen cp = new
ConstantPoolGen(clazz.getConstantPool());
            List<Group> groups = new ArrayList<Group>();
            for (Field field: clazz.getFields()) {
                groups.add(new Group().addFields(field.getName()));
            }
            for (Method method: clazz.getMethods()) {
                Group group = new Group().addMethods(method.getName());
                Code code = method.getCode();
                InstructionList instrs = new
InstructionList(code.getCode());
                for (InstructionHandle ih: instrs) {
                    Instruction instr = ih.getInstruction();
                    if (instr instanceof FieldInstruction) {
                        FieldInstruction fld = (FieldInstruction)instr;
                        if (fld.getClassName(cp).equals(className)) {
                            group.addFields(fld.getFieldName(cp));
                        }
                    } else if (instr instanceof InvokeInstruction) {
                        InvokeInstruction inv = (InvokeInstruction)instr;
                        if (inv.getClassName(cp).equals(className)) {
                            group.addMethods(inv.getMethodName(cp));
                        }
                    }
                }
                if (group.fields() > 0 || group.methods() > 1) {
                    int i = groups.size();
                    while (i > 0) {
                        --i;
                        Group g = groups.get(i);
                        if (g.intersects(group)) {
                            group.merge(g);
                            groups.remove(i);
                        }
                    }
                    groups.add(group);
                }
            }
            return groups;
        }
    }
}
```

```
package vn.edu.iuh.fit;

import org.apache.bcel.classfile.*;
import org.apache.bcel.generic.*;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

public class LCOM4Calculation {
    public List<Group> loadGroups(File file) throws IOException {
        try (InputStream in = new FileInputStream(file)) {
            ClassParser parser = new ClassParser(in, file.getName());
            JavaClass clazz = parser.parse();
            String className = clazz.getClassName();
            ConstantPoolGen cp = new
ConstantPoolGen(clazz.getConstantPool());
            List<Group> groups = new ArrayList<Group>();
            for (Field field: clazz.getFields()) {
                groups.add(new Group().addFields(field.getName()));
            }
            for (Method method: clazz.getMethods()) {
                Group group = new Group().addMethods(method.getName());
                Code code = method.getCode();
                InstructionList instrs = new
InstructionList(code.getCode());
                for (InstructionHandle ih: instrs) {
                    Instruction instr = ih.getInstruction();
                    if (instr instanceof FieldInstruction) {
                        FieldInstruction fld = (FieldInstruction)instr;
                        if (fld.getClassName(cp).equals(className)) {
                            group.addFields(fld.getFieldName(cp));
                        }
                    } else if (instr instanceof InvokeInstruction) {
                        InvokeInstruction inv = (InvokeInstruction)instr;
                        if (inv.getClassName(cp).equals(className)) {
                            group.addMethods(inv.getMethodName(cp));
                        }
                    }
                }
                if (group.fields() > 0 || group.methods() > 1) {
                    int i = groups.size();
                    while (i > 0) {
                        --i;
                        Group g = groups.get(i);
                        if (g.intersects(group)) {
                            group.merge(g);
                            groups.remove(i);
                        }
                    }
                    groups.add(group);
                }
            }
            return groups;
        }
    }
}
```

Driver class

```
package vn.edu.iuh.fit;

import java.io.File;
import java.util.List;

public class Drivers {
    public static void main(String[] args) throws Exception{
        LCOM4Calculation calculation =new LCOM4Calculation();
        File file=new File("T:\\VoVanHai\\source-code\\test-
jdepend\\build\\classes\\java\\main\\org\\example\\CBOMetric.class");
        List<Group> lst = calculation.loadGroups(file);
        lst.forEach(System.out::println);
        int lcom4 = calculation.loadGroups(file).size();
        System.out.printf("LCOM4 of class %s is %d\n",file.getName(),lcom4);
    }
}
```

JDepend

A library for calculate coupling and others metrics

Usage: http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/

Dependency: https://mvnrepository.com/artifact/guru.nidi/jdepend

Using framework

```
jdepend.framework.JDepend d = new jdepend.framework.JDepend();
 d.addDirectory("T:\\VoVanHai\\source-code\\test-jdepend");

Collection<JavaPackage> cols = d.analyze();
cols.forEach(pkg -> {
    System.out.printf("Pakage %s, Ca= %d; Ce=%d;\n", pkg.getName(),
pkg.getAfferents().size(), pkg.getEfferents().size());
});
```

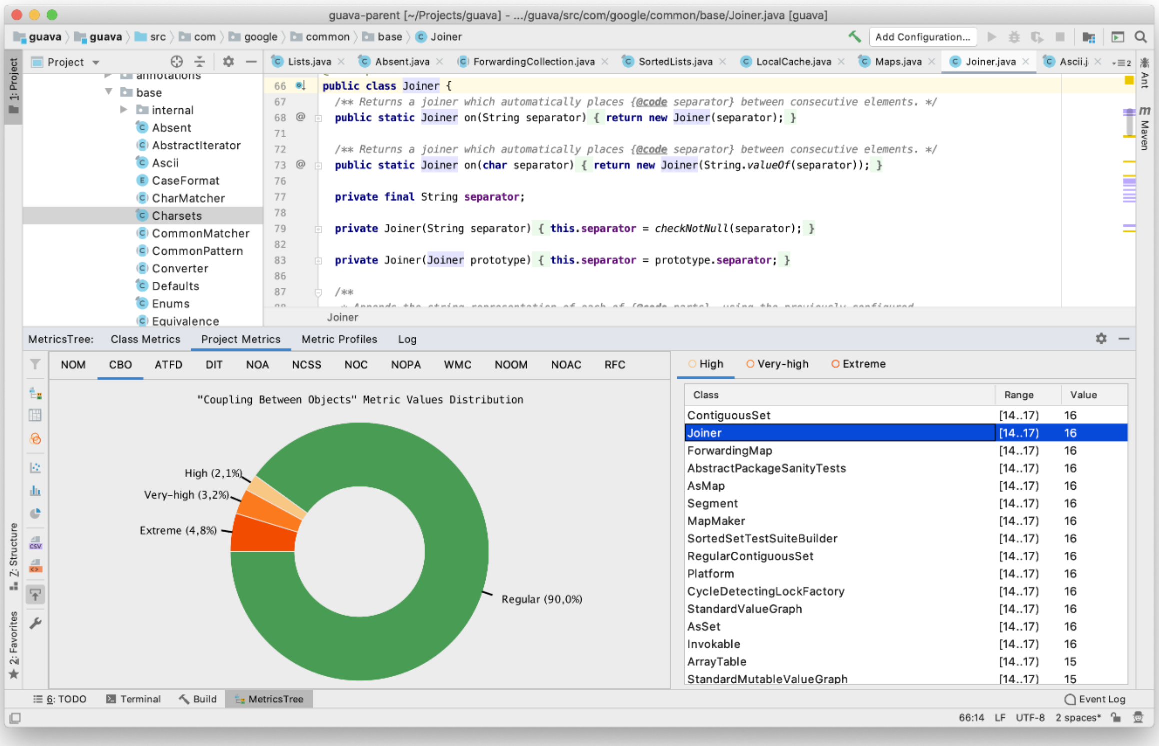
Export xml

```
try (PrintWriter out = new PrintWriter(new FileOutputStream("results.xml")))
{

            jdepend.xmlui.JDepend xml = new jdepend.xmlui.JDepend(out);
            xml.addDirectory("T:\\VoVanHai\\source-code\\test-jdepend");

            PackageFilter f = PackageFilter.all();
//            f.including("vn.edu.iuh");
            f.accept("vn.edu");
            f.excluding("org");
            xml.setFilter(f);

            xml.analyze();
        }
```

Tools

MetricTree

IntelliJ IDEA plugin: https://plugins.jetbrains.com/plugin/13959-metricstree



```java
public class Joiner {
  /** Returns a joiner which automatically places {@code separator} between consecutive elements. */
  public static Joiner on(String separator) { return new Joiner(separator); }

  /** Returns a joiner which automatically places {@code separator} between consecutive elements. */
  public static Joiner on(char separator) { return new Joiner(String.valueOf(separator)); }

  private final String separator;

  private Joiner(String separator) { this.separator = checkNotNull(separator); }

  private Joiner(Joiner prototype) { this.separator = prototype.separator; }

  /**
```

Class	Range	Value
ContiguousSet	[14..17)	16
Joiner	[14..17)	16
ForwardingMap	[14..17)	16
AbstractPackageSanityTests	[14..17)	16
AsMap	[14..17)	16
Segment	[14..17)	16
MapMaker	[14..17)	16
SortedSetTestSuiteBuilder	[14..17)	16
RegularContiguousSet	[14..17)	16
Platform	[14..17)	16
CycleDetectingLockFactory	[14..17)	16
StandardValueGraph	[14..17)	16
AsSet	[14..17)	16
Invokable	[14..17)	16
ArrayTable	[14..17)	15
StandardMutableValueGraph	[14..17)	15

SonaQube

Read yourself

Task2

1. using JDepend (xmlui) analyses the project (your project – using recursion) --> XML files

2. parse - xml documents (using XMLJ)

3. Generate report (packages ok, packages not ok-->shows statistics and actions