МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное образовательное учреждение высшего образования “Национальный исследовательский университет ИТМО”

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине

“МЕТОДЫ И СРЕДСТВА ПРОГРАММНОЙ ИНЖЕНЕРИИ”

Вариант: 345680.

выполнили:

Иванов Матвей Сергеевич

Сосновцев Григорий Алексеевич

Преподаватель

Бострикова Дарья Константиновна

г. Санкт-Петербург, 2023

Содержание

Задание

Написать сценарий для утилиты Apache Ant, реализующий компиляцию, тестирование и упаковку в jar-архив кода проекта из лабораторной работы №3 по дисциплине "Веб-программирование".

Каждый этап должен быть выделен в отдельный блок сценария; все переменные и константы, используемые в сценарии, должны быть вынесены в отдельный файл параметров; MANIFEST.MF должен содержать информацию о версии и о запускаемом классе.

Сценарий должен реализовывать следующие цели (targets):

1. **compile** -- компиляция исходных кодов проекта.
2. **build** -- компиляция исходных кодов проекта и их упаковка в исполняемый jar-архив. Компиляцию исходных кодов реализовать посредством вызова цели **compile**.
3. **clean** -- удаление скомпилированных классов проекта и всех временных файлов (если они есть).
4. **test** -- запуск junit-тестов проекта. Перед запуском тестов необходимо осуществить сборку проекта (цель **build**).
5. **xml** - валидация всех xml-файлов в проекте.
6. **scp** - перемещение собранного проекта по scp на выбранный сервер по завершению сборки. Предварительно необходимо выполнить сборку проекта (цель **build**)
7. **native2ascii** - преобразование native2ascii для копий файлов локализации (для тестирования сценария все строковые параметры необходимо вынести из классов в файлы локализации).
8. **report** - в случае успешного прохождения тестов сохраняет отчет junit в формате xml, добавляет его в репозиторий svn и выполняет commit.
9. **history** - если проект не удаётся скомпилировать (цель **compile**), загружается предыдущая версия из репозитория git. Операция повторяется до тех пор, пока проект не удастся собрать, либо не будет получена самая первая ревизия из репозитория. Если такая ревизия найдена, то формируется файл, содержащий результат операции diff для всех файлов, изменённых в ревизии, следующей непосредственно за последней работающей.
10. **env** - осуществляет сборку и запуск программы в альтернативных окружениях; окружение задается версией java и набором аргументов виртуальной машины в файле параметров.

Реализация

Для выполнения работы была взят код проекта Лабораторной работы №3 по дисциплине “Веб-программирование”: https://github.com/Mrjoulin/ITMOLabs/tree/master/semester_3/WebLabs/Lab3

1. Файл инициализации репозитория

 # init.sh

```
REPO_NAME=mamose_lab_3
REPORT_REPO=report
REPORT_FILENAME=report.xml

git config --global user.name "Matthew"
git config --global user.email matthewthedigital@gmail.com

rm -rf $REPO_NAME
rm -rf ~/.svnrepos/$REPO_NAME

git init $REPO_NAME
cd $REPO_NAME

# SVN report repo init

svnadmin create ~/.svnrepos/$REPO_NAME

svn mkdir -m "Create repository structure." \
    file://$HOME/.svnrepos/$REPO_NAME/trunk \
    file://$HOME/.svnrepos/$REPO_NAME/branches \
    file://$HOME/.svnrepos/$REPO_NAME/tags

svn checkout file://$HOME/.svnrepos/$REPO_NAME/trunk $REPORT_REPO
cd $REPORT_REPO

touch $REPORT_FILENAME
svn add --force $REPORT_FILENAME
svn commit -m "Initial commit" --username="base"
cd ..
```

```
# Git repo init

cp -R ../WebLab3/src ./
cp -R ../build/* ./

git add .
git commit -m "Init repository"

echo "Meow">>src/main/java/com/joulin/lab3/utils/CoordinatesValidation.java
git add .
git commit -m "Meow to file"

echo "Done!"
```

2. Основные файлы сборки Apache Ant

 # build.xml

```
<project name="WebLab3" default="build"
xmlns:antcontrib="http://ant-contrib.sourceforge.net"
xmlns:ant="http://www.w3.org/1999/XSL/Transform">
    <property file="build.properties"/>
    <property name="javav" value="${ant.java.version}"/>
    <property name="jar.file"
value="${jar_dir}/${ant.project.name}.jar"/>

    <path id="classpath">
        <fileset dir="lib" includes="*.jar"/>
    </path>

    <taskdef resource="net/sf/antcontrib/antcontrib.properties">
        <classpath>
            <pathelement location="lib/ant-contrib-1.0b3.jar"/>
        </classpath>
    </taskdef>

    <taskdef resource="net/sf/antcontrib/antlib.xml">
        <classpath>
            <pathelement location="lib/ant-contrib-1.0b3.jar"/>
        </classpath>
    </taskdef>
```

```
    <scriptdef name="propertyreset" language="javascript"
description="Allows to assign @{property} new value">
        <attribute name="name"/>
        <attribute name="value"/>
            project.setProperty(attributes.get("name"),
attributes.get("value"));
    </scriptdef>

    <macrodef name="changeversion">
        <attribute name="version"/>
        <sequential>
            <var name="javav" unset="true" />
            <property name="javav" value="@{version}"/>
            <echo>Set Java version to ${javav}</echo>
        </sequential>
    </macrodef>

    <!-- TARGETS -->

    <target name="compile" description="компиляция исходных кодов
проекта.">
        <echo message="Compiling (JVM v.${javav})..." />
        <mkdir dir="${class_dir}"/>
        <javac includeantruntime="false" target="${javav}"
source="${javav}" srcdir="${source_dir}" destdir="${class_dir}">
            <classpath refid="classpath"/>
        </javac>
        <echo>Done!</echo>
    </target>

    <target name="build" depends="compile" description="упаковка
скомпилированных файлов в исполняемый jar-архив">
        <echo>Building...</echo>
        <mkdir dir="${jar_dir}"/>
        <jar basedir="${class_dir}" destfile="${jar.file}"
manifest="${manifest_path}" />
        <echo>Done!</echo>
    </target>
```

```
    <target name="scp" depends="build" description="перемещение
собранного проекта по scp на выбранный сервер по завершению сборки.
Предварительно необходимо выполнить сборку проекта (цель build)">
        <echo>Copy to server...</echo>
        <scp file="${jar.file}" todir="${scp_dest}:~/"
port="${scp_port}" password="${scp_pass}"/>
        <echo>Done!</echo>
    </target>

    <target name="clean" description="удаление скомпилированных
классов проекта и всех временных файлов">
        <echo>Cleaning...</echo>
        <delete dir="${class_dir}"/>
        <delete dir="${copy_localization_dir}"/>
        <delete file="${diff_file}"/>
        <echo>Done!</echo>
    </target>

    <target name="xml" description="валидация всех xml-файлов в
проекте.">
        <xmlvalidate failonerror="no" lenient="yes" warn="yes">
            <fileset dir="${source_dir}">
                <include name="**/*.xml"/>
            </fileset>
        </xmlvalidate>
    </target>

    <target name="history"
            description="если проект не удаётся скомпилировать (цель
compile), загружается предыдущая версия из репозитория git. Операция
повторяется до тех пор, пока проект не удастся собрать, либо не
будет получена самая первая ревизия из репозитория. Если такая
ревизия найдена, то формируется файл, содержащий результат операции
diff для всех файлов, изменённых в ревизии, следующей
непосредственно за последней работающей.">
        <echo>Checking current version...</echo>
        <trycatch>
            <try>
                <antcall target="compile"/>
            </try>
            <catch>
```

```
                <antcall target="load-previous-revision"/>
                <exec executable="git"
outputproperty="git.cur_commit">
                    <arg value="rev-parse" />
                    <arg value="HEAD" />
                </exec>
                <echo message="Checkout to commit:
${git.cur_commit}" />
            </catch>
        </trycatch>
        <echo>Done!</echo>
    </target>

    <target name="load-previous-revision">
        <exec executable="git" outputproperty="git.next_commit">
            <arg value="rev-parse" />
            <arg value="HEAD" />
        </exec>
        <exec executable="git" outputproperty="git.reverse_commits">
            <arg value="log" />
            <arg value=" --pretty=oneline" />
            <arg value="--reverse"/>
        </exec>
        <propertyregex property="git.first_commit"
input="${git.reverse_commits}" regexp="(\w+)" select="\1" />

        <if>
            <not>
                <equals arg1="${git.first_commit}"
arg2="${git.next_commit}" />
            </not>
            <then>
                <echo>Checking previous version...</echo>
                <exec executable="git">
                    <arg value="checkout" />
                    <arg value="HEAD^" />
                </exec>

                <trycatch>
                    <try>
                        <antcall target="compile"/>
```

```
                        <exec executable="git">
                            <arg value="diff"/>
                            <arg value="${git.next_commit}"/>
                            <redirector output="${diff_file}"/>
                        </exec>
                    </try>
                    <catch>
                        <antcall target="load-previous-revision"/>
                    </catch>
                </trycatch>
            </then>
        </if>
    </target>

    <target name="test" depends="build" description="запуск
junit-тестов проекта. Перед запуском тестов необходимо осуществить
сборку проекта (цель build)">
        <echo>Testing...</echo>
        <mkdir dir="${test_results_dir}" />
        <junit haltonfailure="yes">
            <classpath refid="classpath"/>
            <classpath location="${class_dir}"/>

            <formatter type="plain"/>
            <batchtest fork="yes" todir="${test_results_dir}">
                <fileset dir="${exec_dir}">
                    <include name="**/*Test.java"/>
                </fileset>
                <formatter type="xml"/>
            </batchtest>
        </junit>
        <echo>Done!</echo>
    </target>

    <target name="report" depends="test" description="в случае
успешного прохождения тестов сохраняет отчет junit в формате xml,
добавляет его в репозиторий svn и выполняет commit.">
        <junitreport tofile="${report_name}.xml"
todir="${report_dir}">
            <fileset dir="${test_results_dir}">
                <include name="*.xml"/>
```

```
            </fileset>
            <report format="frames" todir="${report_dir}/html"/>
        </junitreport>
        <exec executable="svn">
            <arg value="update" />
            <arg value="${report_dir}"/>
        </exec>
        <exec executable="svn">
            <arg value="commit" />
            <arg value="${report_dir}"/>
            <arg value="-m" />
            <arg value="Update report" />
        </exec>
    </target>

    <target name="native2ascii" description="преобразование
native2ascii для копий файлов локализации (для тестирования сценария
все строковые параметры необходимо вынести из классов в файлы
локализации).">
        <echo>Native2ASCII coding...</echo>
        <mkdir dir="${copy_localization_dir}"/>
        <native2ascii encoding="EUCJIS" src="${localization_dir}"
dest="${copy_localization_dir}" includes="**/*.loc"/>
        <echo>Done!</echo>
    </target>

    <target name="env" description="осуществляет сборку и запуск
программы в альтернативных окружениях; окружение задается версией
java и набором аргументов виртуальной машины в файле параметров">
        <echo>Set up new environment...</echo>

        <changeversion version="${env_javav}" />
        <antcall target="run"/>
        <changeversion version="${ant.java.version}" />

        <echo>Done!</echo>
    </target>

    <target name="run" depends="build" description="запуск
приложения">
        <echo message="Runing on JVM v.${javav}..." />
```

```
        <exec executable="/usr/libexec/java_home"
outputproperty="java_version.path">
            <arg value="-v=${javav}" />
        </exec>

        <java jar="${jar.file}" fork="true"
jvm="${java_version.path}/bin/java">
            <classpath refid="classpath"/>
            <classpath location="${class_dir}"/>
            <jvmarg value="${java_args}"/>
        </java>
        <echo>Run compile!</echo>
    </target>
</project>
```

 # build.properties

```
source_dir=src
exec_dir=src/main/java
manifest_path=src/main/java/META-INF/MANIFEST.mf
target_dir=target
class_dir=target/classes
jar_dir=target/release
scp_dest=s335105@se.ifmo.ru
scp_port=2222
scp_pass=4uckhelios
localization_dir=src/main/java/com/joulin/lab3/tests
copy_localization_dir=copy_localization
diff_file=diff.txt
env_javav=8
java_args=-Xmx512m
test_results_dir=report/tests
report_dir=report
report_name=report
```

3. Файлы JUnit тестов

 # AreaCheckTest.java

```
package com.joulin.lab3.tests;

import com.joulin.lab3.beans.Coordinates;
import com.joulin.lab3.utils.AreaCheck;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.Vector;

import com.joulin.lab3.utils.CoordinatesValidation;
import org.junit.BeforeClass;
import org.junit.Test;

import static org.junit.Assert.*;

public class AreaCheckTest {
    private static final AreaCheck areaCheck = new AreaCheck();
    private static final Vector<Coordinates> areaCheckCoordinates =
new Vector<>();
    private static final Vector<Boolean> correctAns = new
Vector<>();
    private static final String localisationFilePath =
"src/main/java/com/joulin/lab3/tests/areaCheck.loc";
    private static final int NUM_TESTS = 8;

    @BeforeClass
    public static void setUp() throws FileNotFoundException {
        File loc = new File(localisationFilePath);
        Scanner scanner = new Scanner(loc);
        for (int i = 0; i < NUM_TESTS && scanner.hasNext(); i++) {
            double x = scanner.nextDouble();
            double y = scanner.nextDouble();
            double r = scanner.nextDouble();
            areaCheckCoordinates.add(new Coordinates(x, y, r));
            boolean correct = scanner.nextBoolean();
            correctAns.add(correct);
        }
```

```
    }

    @Test
    public void testNullCoordinates(){
        assertFalse(areaCheck.isHit(null));
    }

    @Test
    public void testAreaCheck_0() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (areaCheckCoordinates.size() > cur_test)

assertEquals(areaCheck.isHit(areaCheckCoordinates.get(cur_test)),
correctAns.get(cur_test));
    }

    @Test
    public void testAreaCheck_1() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (areaCheckCoordinates.size() > cur_test)

assertEquals(areaCheck.isHit(areaCheckCoordinates.get(cur_test)),
correctAns.get(cur_test));
    }

    @Test
    public void testAreaCheck_2() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (areaCheckCoordinates.size() > cur_test)

assertEquals(areaCheck.isHit(areaCheckCoordinates.get(cur_test)),
correctAns.get(cur_test));
    }

    @Test
```

```
    public void testAreaCheck_3() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (areaCheckCoordinates.size() > cur_test)

assertEquals(areaCheck.isHit(areaCheckCoordinates.get(cur_test)),
correctAns.get(cur_test));
    }

    @Test
    public void testAreaCheck_4() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (areaCheckCoordinates.size() > cur_test)

assertEquals(areaCheck.isHit(areaCheckCoordinates.get(cur_test)),
correctAns.get(cur_test));
    }

    @Test
    public void testAreaCheck_5() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (areaCheckCoordinates.size() > cur_test)

assertEquals(areaCheck.isHit(areaCheckCoordinates.get(cur_test)),
correctAns.get(cur_test));
    }

    @Test
    public void testAreaCheck_6() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (areaCheckCoordinates.size() > cur_test)

assertEquals(areaCheck.isHit(areaCheckCoordinates.get(cur_test)),
correctAns.get(cur_test));
```

```
    }

    @Test
    public void testAreaCheck_7() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (areaCheckCoordinates.size() > cur_test)

assertEquals(areaCheck.isHit(areaCheckCoordinates.get(cur_test)),
correctAns.get(cur_test));
    }
}
```

 # CoordinatesValidationTest.java

```
package com.joulin.lab3.tests;

import com.joulin.lab3.beans.Coordinates;
import com.joulin.lab3.utils.CoordinatesValidation;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.Vector;
import org.junit.BeforeClass;
import org.junit.Test;

import static org.junit.Assert.*;

public class CoordinatesValidationTest {
    private static final Vector<Coordinates> coordinatesVector = new
Vector<>();
    private static final Vector<Boolean> correctAns = new
Vector<>();
    private static final String localisationFilePath =
"src/main/java/com/joulin/lab3/tests/coordinatesValidation.loc";
    private static final int NUM_TESTS = 14;

    @BeforeClass
    public static void setUp() throws FileNotFoundException {
```

```
        File loc = new File(localisationFilePath);
        Scanner scanner = new Scanner(loc);
        for (int i = 0; i < NUM_TESTS && scanner.hasNext(); i++) {
            double x = scanner.nextDouble();
            double y = scanner.nextDouble();
            double r = scanner.nextDouble();
            coordinatesVector.add(new Coordinates(x, y, r));
            boolean correct = scanner.nextBoolean();
            correctAns.add(correct);
        }
    }

    @Test
    public void testNullCoordinates(){
        assertFalse(CoordinatesValidation.validate(null));
    }

    @Test
    public void testCoordinatesValidation_0() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (coordinatesVector.size() > cur_test)

assertEquals(CoordinatesValidation.validate(coordinatesVector.get(cu
r_test)), correctAns.get(cur_test));
    }

    @Test
    public void testCoordinatesValidation_1() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (coordinatesVector.size() > cur_test)

assertEquals(CoordinatesValidation.validate(coordinatesVector.get(cu
r_test)), correctAns.get(cur_test));
    }

    @Test
    public void testCoordinatesValidation_2() {
```

```
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (coordinatesVector.size() > cur_test)

assertEquals(CoordinatesValidation.validate(coordinatesVector.get(cu
r_test)), correctAns.get(cur_test));
    }

    @Test
    public void testCoordinatesValidation_3() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (coordinatesVector.size() > cur_test)

assertEquals(CoordinatesValidation.validate(coordinatesVector.get(cu
r_test)), correctAns.get(cur_test));
    }

    @Test
    public void testCoordinatesValidation_4() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (coordinatesVector.size() > cur_test)

assertEquals(CoordinatesValidation.validate(coordinatesVector.get(cu
r_test)), correctAns.get(cur_test));
    }

    @Test
    public void testCoordinatesValidation_5() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (coordinatesVector.size() > cur_test)

assertEquals(CoordinatesValidation.validate(coordinatesVector.get(cu
r_test)), correctAns.get(cur_test));
    }
```

```
    @Test
    public void testCoordinatesValidation_6() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (coordinatesVector.size() > cur_test)

assertEquals(CoordinatesValidation.validate(coordinatesVector.get(cu
r_test)), correctAns.get(cur_test));
    }

    @Test
    public void testCoordinatesValidation_7() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (coordinatesVector.size() > cur_test)

assertEquals(CoordinatesValidation.validate(coordinatesVector.get(cu
r_test)), correctAns.get(cur_test));
    }

    @Test
    public void testCoordinatesValidation_8() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (coordinatesVector.size() > cur_test)

assertEquals(CoordinatesValidation.validate(coordinatesVector.get(cu
r_test)), correctAns.get(cur_test));
    }

    @Test
    public void testCoordinatesValidation_9() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (coordinatesVector.size() > cur_test)
```

```
assertEquals(CoordinatesValidation.validate(coordinatesVector.get(cu
r_test)), correctAns.get(cur_test));
    }

    @Test
    public void testCoordinatesValidation_10() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (coordinatesVector.size() > cur_test)

assertEquals(CoordinatesValidation.validate(coordinatesVector.get(cu
r_test)), correctAns.get(cur_test));
    }

    @Test
    public void testCoordinatesValidation_11() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (coordinatesVector.size() > cur_test)

assertEquals(CoordinatesValidation.validate(coordinatesVector.get(cu
r_test)), correctAns.get(cur_test));
    }

    @Test
    public void testCoordinatesValidation_12() {
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (coordinatesVector.size() > cur_test)

assertEquals(CoordinatesValidation.validate(coordinatesVector.get(cu
r_test)), correctAns.get(cur_test));
    }

    @Test
    public void testCoordinatesValidation_13() {
```

```
        int cur_test =
Integer.parseInt(Thread.currentThread().getStackTrace()[1].getMethod
Name().split("_")[1]);
        if (coordinatesVector.size() > cur_test)

assertEquals(CoordinatesValidation.validate(coordinatesVector.get(cu
r_test)), correctAns.get(cur_test));
    }
}
```

Репозиторий с кодом лабораторной работы:

https://github.com/Mrjoulin/ITMOLabs/tree/master/semester_4/MaMoSE/Lab03

Выводы

В результате выполнения этой лабораторной работы мы научились пользоваться утилитой Apache Ant, что было увлекательно, но очень вряд ли нам пригодится в жизни, так как это доисторическая утилита и все уже давно пользуются Maven и Gradle. Но вот написание JUnit тесов и их внутреннее устройство было весьма полезно узнать, так что не всё так плохо.