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**»

Выполнили:

Фирстова Екатерина Витальевна

Абакумова Ирина Андреевна

Факультет: ПИиКТ

Группа: P32131

Преподаватель:

Бострикова Дарья Константинова

Санкт-Петербург, 2023

Вариант: 315312

Задание варианта:

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. **music** - воспроизведение музыки по завершению сборки (цель **build**).
6. **xml** - валидация всех xml-файлов в проекте.
7. **native2ascii** - преобразование native2ascii для копий файлов локализации (для тестирования сценария все строковые параметры необходимо вынести из классов в файлы локализации).
8. **alt** - создаёт альтернативную версию программы с измененными именами переменных и классов (используя задание replace/replaceregexp в файлах параметров) и упаковывает её в jar-архив. Для создания jar-архива использует цель **build**.
9. **diff** - осуществляет проверку состояния рабочей копии, и, если изменения не касаются классов, указанных в файле параметров выполняет commit в репозиторий git.
10. **report** - в случае успешного прохождения тестов сохраняет отчет junit в формате xml, добавляет его в репозиторий svn и выполняет commit.

Исходный код:

build.xml:

```
<?xml version="1.0"?>
<project name="MispiLab3" default="compile">

	<description>
	Ant build for web third lab
	</description>

	<property file="build.properties"/>
	<property name="git.executable" value="git" />

	<property name="git.repository.dir"
value="/home/studs/s373215/msp-3/web-lab3222/web-lab3" />

	<property name="git.commit.message" value="Commit message" />
	<property name="classes.to.ignore.file" value="./ignore.txt" />

<taskdef resource="net/sf/antcontrib/antcontrib.properties">

	<classpath>
		<pathelement
location="home/studs/s373215/msp-3/web-lab3222/web-lab3/libs/ant-contrib-1.0b3.jar"/>
```

```
        </classpath>
</taskdef>

  <path id="classpath.source">
        <fileset dir="${lib}" includes="*.jar"/>
</path>

<path id="classpath.test">
        <pathelement location="${ant.contrib}"/>
</path>
<!--компиляция исходных кодов проекта-->
<target name="compile">
        <mkdir dir="${compiled.classes}"/>
        <javac srcdir="${main_dir}" destdir="${compiled.classes}" includeantruntime="false"
encoding="UTF-8"
                source="11">
        <classpath>
 <path refid="classpath.source"/>
         <pathelement location="${compiled.classes}"/>
        </classpath>
        </javac>
        <copy todir="${compiled.classes}">
        <fileset dir="${meta-inf}"/>
        </copy>
        <mkdir dir="${compiled.tests}"/>
        <javac srcdir="${test_dir}" destdir="${compiled.tests}" includeantruntime="false"
encoding="UTF-8" source="11">
        <classpath>
                <path refid="classpath.source"/>
                <pathelement location="${compiled.classes}"/>
        </classpath>
        </javac>
</target>

<!--компиляция исходных кодов проекта и их упаковка в исполняемый jar-архив-->
```

```
<target name="build" depends="compile">
	<delete file="${file.jar}"/>
	<mkdir dir="${webapp}"/>
	<copy todir="${webapp}">
	<fileset dir="${web_dir}"/>
	</copy>
	<mkdir dir="${webInf}/classes"/>
	<copy todir="${webInf}/classes">
	<fileset dir="${compiled.classes}"/>
	</copy>
	<mkdir dir="${webInf}/lib"/>
	<copy todir="${webInf}/lib">
	<fileset dir="${lib}"/>
	</copy>
	<jar destfile="${file.jar}" basedir="${webapp}"/>
</target>

<!--запуск junit-тестов проекта-->
<target name="test" depends="build">
	<mkdir dir="${reports}"/>
	<junit fork="true" printsummary="yes" showoutput="yes" failureproperty="tests.failed">
	<formatter type="xml"/>
	<classpath>
		<path refid="classpath.source"/>
		<pathelement location="${compiled.classes}"/>
		<pathelement location="${compiled.tests}"/>
	</classpath>
	<batchtest todir="${reports}">
		<fileset dir="${test2}" includes="*.java"/>
	</batchtest>
	</junit>
</target>


<!-- Воспроизведение музыки по завершению сборки -->
<target name="music" depends="build">
```

```
        <property name="music.file"
value="/home/studs/s373215/msp-3/web-lab3222/web-lab3/music.mp3"/>

        <exec executable="afplay" os="Mac OS X" failonerror="false">
        <arg value="${music.file}"/>
        </exec>

</target>

<!-- Валидация всех XML-файлов -->
<target name="xmlvalidate">
        <apply executable="xmllint" failonerror="true">
        <arg value="--noout"/>
        <srcfile/>
        <fileset dir="/home/studs/s373215/msp-3/web-lab3222/web-lab3" includes="**/*.xml"/>
        <redirector output="validation.log"/>
        </apply>
</target>

<!--удаление скомпилированных классов проекта и всех временных файлов-->
<target name="clean">
        <delete dir="${target}"/>
</target>

<!--создаёт альтернативную версию программы с измененными именами-->
<target name="alt" depends="build">
        <!-- Копирование исходных файлов в новую директорию -->
        <copy todir="${srcDir2}">
        <fileset dir="${src}"/>
        </copy>

        <!-- Замена имен переменных и классов в файлах параметров -->
        <replaceregexp file="build.properties" match="./src/main" replace="./src2/main"/>
        <replaceregexp file="build.properties" match="./src/test" replace="./src2/test"/>

        <!-- Перекомпиляция измененных исходных файлов -->
        <antcall target="compile"/>
```

```
	<!-- Упаковка альтернативной версии программы в jar-архив -->
	<delete file="${file.alt.jar}"/>
	<mkdir dir="${webapp.alt}"/>
	<copy todir="${webapp.alt}">
	<fileset dir="${web_dir}"/>
	</copy>
	<mkdir dir="${webInf.alt}/classes"/>
	<copy todir="${webInf.alt}/classes">
	<fileset dir="${compiled.classes}"/>
	</copy>
	<mkdir dir="${webInf.alt}/lib"/>
	<copy todir="${webInf.alt}/lib">
	<fileset dir="${lib}"/>
	</copy>
	<jar destfile="${file.alt.jar}" basedir="${webapp.alt}"/>
</target>


<!--преобразование native2ascii для копий файлов локализации-->
<target name="native2ascii">
	<native2ascii encoding="UTF-8" src="./src/resources/" dest="./src/resources_new"
		includes="*.properties.utf8" ext=""/>
</target>


<!-- Сохранение отчета JUnit в формате XML -->
<target name="report" depends="test" description="Reports test result to xml file">
	<mkdir dir="report"/>
	<exec executable="svn">
	<arg value="add"/>
	<arg value="*"/>
	</exec>

	<exec executable="svn">
	<arg value="commit"/>
	<arg value="-m 'prekol_${rev.id}'"/>
```

```
        </exec>
        <exec executable="svn">
        <arg value="update"/>
        </exec>

        <propertyfile file="build.properties">
        <entry key="rev.id" type="int" operation="+" value="1"/>
        </propertyfile>
</target>


<!-- Задача для выполнения команды 'git diff' -->
<target name="diff">
        <exec executable="${git.executable}" dir="${git.repository.dir}">
        <arg value="diff" />
        </exec>
</target>

<!-- Задача для выполнения команды 'git commit' -->
        <target name="git-commit">
        <exec executable="${git.executable}" dir="${git.repository.dir}">
        <arg value="commit" />
        <arg value="-m" />
        <arg value="${git.commit.message}" />
        </exec>
</target>

<!-- Задача для проверки изменений и выполнения коммита -->
        <target name="diff-and-commit" depends="diff">
        <condition property="commit.required">
        <and>
                <!-- Проверка наличия изменений в рабочей копии -->
                <available file="${git.repository.dir}/.git/index" />
                <not>
                <!-- Проверка, что изменения затрагивают только указанные классы -->
                <uptodate targetfile="${classes.to.ignore.file}">
                <srcfiles dir="${git.repository.dir}">
```

```
                <exclude name=".git/**" />
            </srcfiles>
            </uptodate>
            </not>
        </and>
        </condition>

        <!-- Выполнение коммита, если изменения не затрагивают указанные классы -->
        <if>
        <isset property="commit.required" />
        <then>
            <antcall target="git-commit" />
        </then>
        </if>
</target>


<target name="commit" if="${found-diff}">
        <exec executable="svn">
        <arg line="add"/>
        <arg value="*"/>
        <arg value="--force"/>
        </exec>
</target>
</project>
```

junit-тесты:

```
public class WebLabTests {

        private static Point point;


        @BeforeAll
        public static void create() {
            point = new Point();
            point.setId(100);
```

```
        point.setX(0.0);
        point.setR(0.0);
        point.setY(0.0);
}

@Test
public void isHit() {
        double[] xValues = {1, -5, 0, -5, -10};
        double[] yValues = {0, -5, 0, -1, -10};

        boolean[] expected = {true, false, true, false, false};
        boolean[] actual = new boolean[5];
        for (int i = 0; i < xValues.length; i++) {
        Point point = new Point();
        point.setX(xValues[i]);
        point.setY(yValues[i]);
        point.setR(2.0);
        actual[i] = point.isHit(point.setX(xValues[i]),
                point.setY(yValues[i]));
        }
        Assertions.assertArrayEquals(expected, actual);
}


@Test
public void checkValidPoint() {
        InputValidator inputValidator = new InputValidator();

        double[] xValues = {1.0, -5.0, 0.0, -5.0, -10.0};
        double[] yValues = {2, -5, 0, -1, -10};
        double[] rValues = {3, 2, 0, 3, -10};

        boolean[] expected = {true, false, false, false, false};
        boolean[] result = new boolean[5];

        for (int i = 0; i < xValues.length; i++) {
        Point point1 = new Point();
```

```
            point.setX(xValues[i]);
            point.setY(yValues[i]);
            point.setR(rValues[i]);
            result[i] = inputValidator.validateAll(point.setX(xValues[i]),
                    point.setY(yValues[i]), point.setR(rValues[i]));
            }
            Assertions.assertArrayEquals(expected, result);
    }

    @Test()
    public void nullPointFields() {
            Point point = new Point();
            Assertions.assertNull(point.getX());
            Assertions.assertNull(point.getY());
            Assertions.assertNull(point.getR());



    }

    @AfterAll
    public static void remove() {
            point = null;
    }
}
```

Вывод:

В ходе выполнения данной лабораторной работы мы познакомились с системой автоматической сборки Ant, а также написали несколько junit-тестов. В дальнейшем планируем пользоваться более удобными системами, например Maven или Gradle.