Lab14: Maven-д Суурилсан Java Төсөлд Git Workflow болон GitHub Actions Хэрэглэх

Зорилго: Git workflow-ийн ойлголтуудыг GitHub дээр Maven-д суурилсан Java төсөлд хамтын ажиллагааны программ хангамж хөгжүүлэлтийн үйл явцад хэрэглэж, хувилбарын удирдлага, салбарлах, болон хувилбар гаргахыг загварчлах. GitHub Actions ашиглан JUnit тест, Checkstyle-ийн кодын загварын шалгалт, болон JaCoCo-ийн 100% branch coverage-ийг автоматжуулсан Continuous Integration (CI) процесс хэрэгжүүлэх. main салбарыг шууд push-ээс хамгаалж, Checkstyle эсвэл 100% branch coverage хангагдаагүй бол merge-ийг автоматаар татгалзах.

Заавар: Доорх даалгавруудыг гүйцэтгэнэ үү. Ажлаа олон нийтэд нээлттэй GitHub репозиторийн холбоосоор ирүүлнэ. Репозитори нь таны Git workflow-ийг харуулж, JUnit тест, Checkstyle шалгалт, 100% branch coverage-ийг баталгаажуулдаг ажиллагаатай GitHub Actions CI процесс агуулсан байх ёстой.

Даалгавруud

1. Maven-д Суурилсан Java Репозитори Бий Болгох

❿ Зохиомол Java төслийн (жишээ нь, энгийн тооны машины апп) **олон нийтэд нээлттэй** GitHub репозитори үүсгэ.

❿ mvn archetype:generate эсвэл захиалгат pom.xml ашиглан Maven төслийн бүтцийг эхлүүл. Дараахыг оруул:

❿ JUnit 5 (junit-jupiter), Maven Surefire Plugin (тест хийхэд), Maven Checkstyle Plugin (кодын загвар шалгахад), болон JaCoCo Maven Plugin (code coverage шалгахад) зэрэг хамаарлуудыг агуулсан pom.xml.

❿ Эх кодын хувьд src/main/java/labxx/sict/must/edu/mn, тест файлуудын хувьд src/test/java/labxx/sict/must/edu/mn гэсэн энгийн файлын бүтэц.

❿ Төслийн үндсэн хавтаст Google-ийн Checkstyle дүрэм эсвэл захиалгат тохиргоотой checkstyle.xml файл.

❿ Төслийн тодорхойлолт болон хэрхэн угсрах/ажиллуулахыг (жишээ нь, mvn test, mvn checkstyle:check, mvn jacoco:report) тайлбарласан README.md.

❿ Java/Maven-д зориулсан .gitignore файл (жишээ нь, target/, .idea/, *.class хасах).

❿ Эхний бүтцийг main салбарт тодорхой коммитын мессежтэй (жишээ нь, "Maven-д суурилсан Java тооцоолуурын төслийг эхлүүлэх") коммит хий.

❿ **Main салбарын хамгаалалт**:

❿ GitHub дээр репозиторийн "Settings" > "Branches" хэсэгт оч.

❿ "Branch protection rules" дээр "Add rule" дарж, main салбарт дараахыг тохируул:

❿ "Require a pull request before merging" идэвхжүүл.

❿ "Require status checks to pass before merging" идэвхжүүлж, CI workflow-ийн нэр (жишээ нь, "CI Процесс") сонго.

❿ "Require branches to be up to date before merging" идэвхжүүл.

❿ "Do not allow bypassing the above settings" идэвхжүүл.

❿ Энэ нь main руу шууд push хийхийг хориглож, зөвхөн CI шалгалт амжилттай болсон PR-ээр нэгтгэх боломжийг өгнэ.

Жишээ pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>labxx.sict.must.edu.mn</groupId>
    <artifactId>calculator</artifactId>
    <version>1.0-SNAPSHOT</version>
    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>5.10.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>3.2.5</version>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-checkstyle-plugin</artifactId>
                <version>3.3.1</version>
                <configuration>
                    <configLocation>checkstyle.xml</configLocation>
                    <failOnViolation>true</failOnViolation>
                </configuration>
                <executions>
                    <execution>
                        <id>validate</id>
                        <phase>validate</phase>
                        <goals>
                            <goal>check</goal>
```

```
            </goals>
          </execution>
        </executions>
      </plugin>
      <plugin>
        <groupId>org.jacoco</groupId>
        <artifactId>jacoco-maven-plugin</artifactId>
        <version>0.8.12</version>
        <executions>
          <execution>
            <goals>
              <goal>prepare-agent</goal>
            </goals>
          </execution>
          <execution>
            <id>report</id>
            <phase>test</phase>
            <goals>
              <goal>report</goal>
            </goals>
          </execution>
          <execution>
            <id>check</id>
            <goals>
              <goal>check</goal>
            </goals>
            <configuration>
              <rules>
                <rule>
                  <element>BRANCH</element>
                  <limits>
                    <limit>
                      <counter>BRANCH</counter>
                      <value>COVEREDRATIO</value>
                      <minimum>1.0</minimum>
                    </limit>
                  </limits>
                </rule>
              </rules>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

Жишээ checkstyle.xml (Google загварын хялбаршуулсан хувилбар):

```
<?xml version="1.0"?>
<!DOCTYPE module PUBLIC "-//Checkstyle//DTD Configuration 1.3//EN"
   "https://checkstyle.sourceforge.io/dtds/configuration_1_3.dtd">
```

```
<module name="Checker">
  <property name="severity" value="error"/>
  <module name="TreeWalker">
    <module name="Indentation"/>
    <module name="MethodLength">
      <property name="max" value="100"/>
    </module>
    <module name="LineLength">
      <property name="max" value="120"/>
    </module>
  </module>
</module>
```

2. Салбарлах болон Feature/Боломж Хөгжүүлэлт

➓ Хөгжүүлэлтийн явцад зориулж develop салбар үүсгэ.

➓ Багийн ажлын урсгалыг дуурайлган develop салбараас **хоёр онцлогын салбар** (жишээ нь, feature/add-multiplication, feature/add-division) үүсгэ.

➓ Салбар бүрт:

➓ src/main/java/labxx/sict/must/edu/mn хавтаст Java класс (жишээ нь, Multiplication.java эсвэл Division.java) нэмж, энгийн функц (жишээ нь, double multiply(double a, double b)) оруул.

➓ src/test/java/labxx/sict/must/edu/mn хавтаст JUnit тест класс (жишээ нь, MultiplicationTest.java) нэмж, хамгийн багадаа хоёр тест хэрэг (жишээ нь, @Test void testMultiplyPositive()) оруул, 100% branch coverage хангасан байх (жишээ нь, бүх if нөхцөл тестлэгдсэн).

➓ Код checkstyle.xml дүрмийг дагаж байгаа эсэхийг шалга (локал дээр mvn checkstyle:check ажиллуул).

➓ Тодорхой мес zwiercежтэй коммит хий (жишээ нь, "Хүмүүсийн функц болон JUnit тест нэмсэн").

➓ Хоёр онцлогын салбарыг develop руу нэгтгэхийн тулд pull request (PR) үүсгэ. PR бүр:

➓ Өөрчлөлтийн тодорхойлолт агуулсан байх ёстой.

➓ GitHub-ийн merge товчийг ашиглан нэгтгэгдэнэ (“Create a merge commit” сонголтыг хэрэглэ).

➓ Checkstyle эсвэл 100% branch coverage хангагдаагүй бол CI шалгалт амжилтгүй болж, merge татгалзагдана.

➓ PR үүсгэхээс өмнө локал дээр mvn test, mvn checkstyle:check, болон mvn jacoco:report ажиллуулж, coverage 100% эсэхийг шалга (target/site/jacoco/index.html дээр харагдана).

Жишээ Java Код (src/main/java/labxx/sict/must/edu/mn/Multiplication.java):

```
package labxx.sict.must.edu.mn;

public class Multiplication {
  public double multiply(double a, double b) {
```

```
        return a * b;
    }
}
```

Жишээ JUnit Тест (src/test/java/labxx/sict/must/edu/mn/MultiplicationTest.java):

```
package labxx.sict.must.edu.mn;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class MultiplicationTest {
    @Test
    void testMultiplyPositive() {
        Multiplication calc = new Multiplication();
        assertEquals(6.0, calc.multiply(2.0, 3.0), "2 * 3 нь 6-тай тэнцүү байх ёстой");
    }

    @Test
    void testMultiplyNegative() {
        Multiplication calc = new Multiplication();
        assertEquals(-6.0, calc.multiply(-2.0, 3.0), "-2 * 3 нь -6-тай тэнцүү байх ёстой");
    }
}
```

3. Хувилбарын Удирдлага

- develop салбараас release/v1.0.0 салбар үүсгэ.

- release/v1.0.0 салбарт v1.0.0 Git tag нэмж, tag-ийг репозитори руу push хий (git tag v1.0.0; git push origin v1.0.0).

- Хотфикс загварчлах:

- release/v1.0.0 салбараас hotfix/v1.0.1 салбар үүсгэ.

- Java классын аль нэгэнд алдаа зас (жишээ нь, Division.java дээр тэгээр хуваахыг зохицуулах нэмэлт оруулах).

- Холбогдох JUnit тестыг шинэчлэн засварыг баталгаажуул, 100% branch coverage хангасан байх.

- Засвар checkstyle.xml дүрмийг дагасан эсэхийг шалга.

- Хотфиксыг release/v1.0.0 руу нэгтгэх PR болон develop руу нэгтгэх өөр PR үүсгэ. CI шалгалт (Checkstyle, 100% coverage) амжилтгүй бол merge татгалзагдана.

- Хотфиксыг v1.0.1 гэж tag хийж, tag-ийг push хий.

- Бүх салбар болон tag-уудыг алсын репозитори руу push хий.

Жишээ Хотфикс (src/main/java/labxx/sict/must/edu/mn/Division.java):

```
package labxx.sict.must.edu.mn;

public class Division {
```

```
    public double divide(double a, double b) {
        if (b == 0) {
            throw new IllegalArgumentException("Тэгээр хуваах боломжгүй");
        }
        return a / b;
    }
}
```

Жишээ Шинэчлэгдсэн Тест (src/test/java/labxx/sict/must/edu/mn/DivisionTest.java):

```
package labxx.sict.must.edu.mn;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertThrows;

public class DivisionTest {
    @Test
    void testDivideNormal() {
        Division calc = new Division();
        assertEquals(2.0, calc.divide(4.0, 2.0), "4 / 2 нь 2-той тэнцүү байх ёстой");
    }

    @Test
    void testDivideByZero() {
        Division calc = new Division();
        assertThrows(IllegalArgumentException.class, () -> calc.divide(4.0, 0.0), "Тэгээр хуваах нь
exception өгчих байх ёстой");
    }
}
```

4. GitHub Actions CI Процесс (25 оноо)

❿ develop болон release/* салбарууд руу хийсэн push болон pull request дээр JUnit тест, Checkstyle шалгалт, болон JaCoCo-ийн 100% branch coverage-ийг автоматжуулсан GitHub Actions процесс үүсгэ.

❿ Репозиторидоо .github/workflows/ci.yml файл үүсгэж, дараахыг оруул:

❿ Java болон Maven тохируулдаг ажил.

❿ Кодыг checkout хийх, JDK 17 тохируулах, mvn checkstyle:check (загварын шалгалтын хувьд), mvn test (JUnit тестүүдийн хувьд), болон mvn jacoco:check (100% branch coverage шалгахад) гүйцэтгэх алхмууд.

❿ Workflow файлыг коммит хийж, CI процесс амжилттай ажилласан эсэхийг шалга (GitHub дээрх “Actions” таб дээр харагдана). Checkstyle эсвэл branch coverage 100% хангагдаагүй бол CI алдаа гарч, merge татгалзагдана.

GitHub Actions Workflow (.github/workflows/ci.yml):

```
name: CI Процесс
on:
  push:
```

```
    branches:
      - develop
      - release/*
  pull_request:
    branches:
      - develop
      - release/*
jobs:
  build-and-test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: JDK 17 Тохируулах
        uses: actions/setup-java@v4
        with:
          java-version: '17'
          distribution: 'temurin'
      - name: Checkstyle Шалгах
        run: mvn checkstyle:check
      - name: JUnit Тест ба JaCoCo Coverage Шалгах
        run: mvn test jacoco:check
      - name: JaCoCo Тайлан Үүсгэх
        run: mvn jacoco:report
```

5. Хамтын Ажиллагаа Харуулах (10 оноо)

➓ Шинэ онцлогын салбар болон develop дээр ижил Java файлын (жишээ нь, src/main/java/labxx/sict/must/edu/mn/Multiplication.java) ижил мөрийг өөрчлөн (жишээ нь, multiply функцын Javadoc коммент өөрчлөх) merge conflict үүсгэ.

➓ Merge conflict-ийг локал дээр эсвэл GitHub-ийн conflict resolution хэрэгслээр шийд.

➓ Репозиторидоо conflict_resolution.md файл үүсгэж, дараахыг баримтжуул:

➓ Conflict-ийг хэрхэн шийдсэн алхмууд.

➓ Хэрэглэсэн Git командууд (жишээ нь, git merge, git add, git commit).

➓ Шийдэгдсэн өөрчлөлтийг PR-ээр коммит хийж, push хий. CI шалгалт (Checkstyle, 100% coverage) амжилтгүй бол merge татгалзагдана.

Жишээ conflict_resolution.md:

```
# Conflict Шийдвэрлэлт


## Conflict-ийн Дэлгэрэнгүй
`feature/add-multiply-validation`-ийг `develop` руу нэгтгэхэд
`src/main/java/labxx/sict/must/edu/mn/Multiplication.java` дээр merge conflict гарсан. Conflict нь
`multiply` функцын Javadoc коммент дээр байсан.


## Шийдвэрлэлтийн Алхмууд
1. Онцлогын салбарт `git merge develop` ажиллуулсан.
```

2. `Multiplication.java` дээрх conflict-ийг тодорхойлсон.

3. Файлыг засварлаж, хоёр салбарын Javadoc комментийг нэгтгэсэн.

4. `git add src/main/java/labxx/sict/must/edu/mn/Multiplication.java` ажиллуулсан.

5. `git commit` ажиллуулж merge-ийг дуусгасан.

6. Салбарыг push хийж, PR үүсгэсэн.

Ирүүлэх Заавар

➓ **Репозитори**: Олон нийтэд нээлттэй GitHub репозиторийн холбоосоо хуваалц. Репозитори дараахыг агуулсан байх ёстой:

➓ Бүх салбарууд (main, develop, feature/*, release/*, hotfix/*).

➓ Tag-ууд (v1.0.0, v1.0.1).

➓ JUnit 5, Maven Surefire Plugin, Maven Checkstyle Plugin, JaCoCo Maven Plugin агуулсан зөв pom.xml.

➓ Төслийн үндсэн хавтаст checkstyle.xml файл.

➓ .github/workflows/ci.yml файл болон амжилттай CI гүйцэтгэлийн нотолгоо (“Actions” таб дээр харагдана).

➓ conflict_resolution.md файл.

➓ main салбарт branch protection rule тохируулагдсан байх.

Зөвлөмж

➓ Тодорхой коммитын мессеж хэрэглэ (жишээ нь, “Тэгээр хуваахыг зохицуулсан division нэмсэн” гэдэг “Шинэчлэлт” гэхээс илүү).

➓ GitHub руу push хийхээс өмнө локал дээр mvn test, mvn checkstyle:check, mvn jacoco:report ажиллуулж, coverage 100% эсэхийг шалга (target/site/jacoco/index.html).

➓ Checkstyle-д зориулж энгийн checkstyle.xml хэрэглэж, локал дээр алдааг засаж CI-д амжилттай болго.

➓ JaCoCo-д 100% branch coverage хангахын тулд бүх if/else нөхцөл, exception-уудыг тестээр хамар.

➓ GitHub Actions-д тест, Checkstyle, эсвэл coverage алдаа гарвал “Actions” таб дээрх логыг шалга.

➓ CI-д нийцтэй байлгахын тулд энгийн JUnit тохиргоо (жишээ нь, @Test-тэй assertEquals) хэрэглэ.

➓ main салбарт branch protection зөв тохируулагдсан эсэхийг “Settings” > “Branches” дээр шалга.

➓ GitHub Actions Docs, Maven Checkstyle Plugin, JUnit 5 User Guide, JaCoCo Docs, Checkstyle Google Style зэрэг эх сурвалжуудыг ашигла.

➓ Conflict-ийг git merge-ээр локал дээр шийдэж, танил болоорой.