Dział III: Testy, diagramy, dokumentacja

Uporządkowanie aplikacji RhymersDemo

1. Utworzono nową gałąź docs:

   git checkout -b docs

```
PS C:\MM\MOJE\pio_git_rhymers> git config --global user.email "mateuszmroz001@gmail.com"
PS C:\MM\MOJE\pio_git_rhymers> git config --global user.name "MattyMroz"
PS C:\MM\MOJE\pio_git_rhymers> git checkout -b docs
Switched to a new branch 'docs'
PS C:\MM\MOJE\pio_git_rhymers>
```

2. Zrefaktoryzowano metodę main w RhymersDemo:

   - Użyto Refactor → Extract Method do wyodrębnienia metody testRhymers

   - Przeniesiono logikę testową do nowej metody statycznej

```
public static void main(String[] args) {
    RhymersFactory factory = new DefaultRhymersFactory();

    testRhymers(factory);

}

private static void testRhymers(RhymersFactory factory) {  1 usage
    DefaultCountingOutRhymer[] rhymers = { factory.getStandardRhymer(), factory.getFalseRhymer(),
            factory.getFIFORhymer(), factory.getHanoiRhymer()};

    for (int i = 1; i < 15; i++)
```

Reorganizacja projektu

1. Przeanalizowano klasy pod kątem modyfikatorów dostępu
2. Zidentyfikowano klasy, które nie muszą być publiczne
3. Usunięto zbędne pliki i zoptymalizowano strukturę projektu
4. Testy działają
   git commit -m "Projekt jest poprawny!" – dodano commit i wypchnięto zmiany

Testy jednostkowe

1. Dodano nowe testy jednostkowe:

   - Wybrano klasę do testowania i napisano testy

```java
package edu.kis.vh.nursery;

import org.junit.Assert;
import org.junit.Test;

public class HanoiRhymerTest {

    @Test
    public void testCountIn() {
        HanoiRhymer rhymer = new HanoiRhymer();

        rhymer.countIn(in:10);
        Assert.assertEquals(expected:10, rhymer.peek());
        Assert.assertEquals(expected:0, rhymer.reportRejected());

        rhymer.countIn(in:5);
        Assert.assertEquals(expected:5, rhymer.peek());
        Assert.assertEquals(expected:0, rhymer.reportRejected());

        rhymer.countIn(in:15);
        Assert.assertEquals(expected:5, rhymer.peek());
        Assert.assertEquals(expected:1, rhymer.reportRejected());

        rhymer.countIn(in:20);
        Assert.assertEquals(expected:5, rhymer.peek());
        Assert.assertEquals(expected:2, rhymer.reportRejected());
    }

    @Test
    public void testCountOut() {
        HanoiRhymer rhymer = new HanoiRhymer();

        rhymer.countIn(in:15);
        rhymer.countIn(in:10);
        rhymer.countIn(in:5);

        Assert.assertEquals(expected:5, rhymer.countOut());
        Assert.assertEquals(expected:10, rhymer.countOut());
        Assert.assertEquals(expected:15, rhymer.countOut());
        Assert.assertTrue(rhymer.isEmpty());
    }

    @Test
    public void testReportRejected() {
        HanoiRhymer rhymer = new HanoiRhymer();

        Assert.assertEquals(expected:0, rhymer.reportRejected());

        rhymer.countIn(in:5);
        Assert.assertEquals(expected:0, rhymer.reportRejected());

        rhymer.countIn(in:10);
        rhymer.countIn(in:15);
        rhymer.countIn(in:20);

        Assert.assertEquals(expected:3, rhymer.reportRejected());
    }
}
```

Dokumentacja

1. Wygenerowano szkielet dokumentacji
2. Przegląd jakości kodu

Zgłoszenie rozwiązania

1. Utworzono gałąź review:

   - Wybrano odpowiedni commit bazowy

   - Utworzono nową gałąź bez własnych commitów

2. Utworzono Pull Request:

   - Z gałęzi master do review

   - Dodano prowadzącego jako reviewera