Hogeschool van Amsterdam

Discrete Mathematics

Assignment 1

Auteurs:
David Soff
Joe Vrolijk

Datum: 16-02-2021

Opdracht 1 – A :

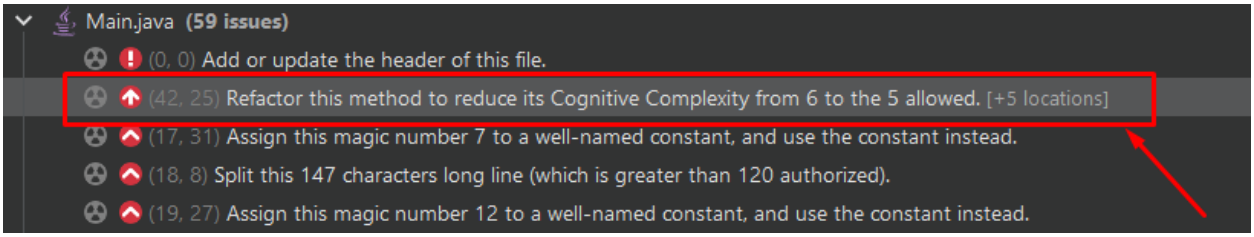
Zie hieronder de originele code die wij voor de opdracht gebruikt hebben:

```java
/**
 * Deze method vraagt de user om een cijfer voor een vak. Deze input wordt gevalideerd en returned met het correcte cijfer.
 * Als men een verkeerde input of range invult dan blijft de method loopen totdat het cijfer valideerd.
 *
 * @param vakNaam Naam van het specificeerde vak.
 * @return returned een gevalideerd cijfer voor een vak.
 */
public static double getUserCijfer(String vakNaam) {
    Scanner input = new Scanner(System.in);
    System.out.print(vakNaam + ": ");
    if (input.hasNextDouble()) {
        double num = input.nextDouble();
        if (num ≥ 0 && num ≤ 10) {
            return num;
        } else {
            System.out.println("Verkeerde invoer. Gebruik een nummer tussen de 0 en 10");
            return -1;
        }
    } else {
        System.out.println("Verkeerde invoer. Gebruik een cijfer!");
        return -1;
    }
}
```

Nadat wij SonarLint hebben geinstalleerd en de ‘threshold’ naar 5 hebben gezet kregen wij het volgende:

```java
public static double getUserCijfer(String vakNaam) {
    Scanner input = new Scanner(System.in);
    System.out.print(vakNaam + ": ");
    1 if (input.hasNextDouble()) {
        double num = input.nextDouble();
        2 if (num ≥ 0 3 && num ≤ 10) {
            return num;
        } 4 else {
            System.out.println("Verkeerde invoer. Gebruik een nummer tussen de 0 en 10");
            return -1;
        }
    } 5 else {
        System.out.println("Verkeerde invoer. Gebruik een cijfer!");
        return -1;
    }
}
```

SonarLint gaf ons de volgende melding:



Voor opdracht 1a – 6 hebben we de logica in een predicate verwerkt. Het resultaat is als volgt:

```java
public static double getUserCijfer(String vakNaam) {
    Scanner input = new Scanner(System.in);
    System.out.print(vakNaam + ": ");

    if (input.hasNextDouble()) {
        double num = input.nextDouble();
        if (isValidGrade().test(num)) {
            return num;
        } else {
            System.out.println("Verkeerde invoer. Gebruik een nummer tussen de 0 en 10");
            return -1;
        }
    } else {
        System.out.println("Verkeerde invoer. Gebruik een cijfer!");
        return -1;
    }
}

public static DoublePredicate isValidGrade() {
    DoublePredicate isAboveLowerBound = grade → grade ⩾ 0;
    DoublePredicate isBelowUpperBound = grade → grade ⩽ 10;

    return isAboveLowerBound
        .and(isBelowUpperBound);
}
```

Hiermee is de cognitieve melding van SonarLint verdwenen en zitten we onder de Cognitieve Complexiteit threshold van 5.

Bij de opdracht om te onderzoeken om we met de “Morgan Regels” de logica kunnen versimpelen hebben wij alle vormen uitgewerkt. Zowel de double negation als de Morgan variant:

```java
public static DoublePredicate isValidGradeDoubleNegation() {
    DoublePredicate isAboveLowerBound = grade → grade ⩾ 0;
    DoublePredicate isBelowUpperBound = grade → grade ⩽ 10;

    return isAboveLowerBound.negate().negate()
        .and(isBelowUpperBound.negate().negate());
}

public static DoublePredicate isValidGradeWithMorganRules() {
    DoublePredicate isAboveLowerBound = grade → grade ⩾ 0;
    DoublePredicate isBelowUpperBound = grade → grade ⩽ 10;

    return (isAboveLowerBound.negate()
        .or(isBelowUpperBound.negate())).negate();
}
```

Gezien de vele negations vinden wij dat de "Morgan regels" en "Double Negation" minder goed leesbaar zijn en hebben wij besloten om deze varianten niet te gebruiken bij het verduidelijken van de logica. Daarbij is het makkelijker om logica te begrijpen als je uitgaat van positieve condities i.p.v. logica te begrijpen met negaties.

Opdracht 1 – B

Zie bijlagen voor de projecten.