C - Take Home

Team Members:

- Lazar Camelia
- Ghilea Ramona
- Ivan Adrian
- Comsa Filip
- Foidas Andrei

Task 1

- **Mystery Guest test smell**
  - We have noticed that all of the test methods use data from external files
  - **Time spent:** 5 minutes.
- **Eager Test smell**
  - In the **test_Dominator_And_Numerator()** function we noticed that there were many other functions being called with regard to different behaviours.
  - **Time spent:** 15 minutes
- **Duplicate Assert test smell**
  - In the test method **testSimplify**() - **getNominator**() and **getDominator**() are tested twice
  - In the test method **test_Denominator**(), both the functions getNominator and getDenominator are tested twice.
  - **Time spent**: 10 minutes
- **Test Code Duplication test smell**
  - Lines 24-25 and 30-31 in the **testSimplify**() function are cloned, and they can be extracted in another function. Same for lines 43-44 and 48-49.
  - **Time spent**: 10 minutes
- We split the work for Task 1 in the following manner:
  - Two persons from the team handled the first two test smells, while the other three focused on the other two

Task 2

- **Mystery Guest test smell**
  - After studying the usage of the files, we have discovered that their usage was not mandatory and we could easily refactor the code to remove the need of having them, and to reduce the latency generated by their usage.
  - **Time spent:** 10 minutes
- **Eager Test smell**
  - We split the **test_Dominator_And_Numerator()** function (which was created at an intermediary part from the test_Dominator() function) into three new tests, checking everything that was previously tested in that function: **test_Numerator()**, **test_Dominator_set()**, **test_Numerator_set()**
  - **Time spent:** 15 minutes
- **Duplicate Assert test smell**
  - **Time spent**: 15 minutes

- **Test Code Duplication test smell**
  - **Time spent**: 10 minutes
- We split the work for Task 2 in the following manner:
  - Two persons from the team handled the first two test smells, while the other three focused on the other two. The sub-team consisting of three people also created the Github Repository where we the code was manipulated.

Refactored Code

```
public class FractionTest {

    private Fraction fc1, fc2, fc3;

    @Test
    public void testSimplify() {
        fc1 = new Fraction(12, 30);
        fc1.Simplify();

        assertEquals(2, fc1.getNumerator());
        assertEquals(5, fc1.getDenominator());
    }

    @Test
    public void testSimplify_Negative_Numerator() {
        fc2 = new Fraction(-25, 7);
        fc2.Simplify();

        assertEquals(-25, fc2.getNumerator());
        assertEquals(7, fc2.getDenominator());
    }

    @Test
    public void test_Denominator() {
        fc1 = new Fraction(12, 30);
        int result = fc1.getDenominator();
        assertTrue("getDenominator() returned " + result + " instead of 30.",
                   result == 30);
    }

    @Test
    public void test_Numerator() {
        fc2 = new Fraction(-25, 7);

        int result2 = fc2.getNumerator();
        assertTrue("getNumerator() returned " + result2 + " instead of -25.",
                   result2 == -25);
    }

    @Test
    public void test_Denominator_set() {
        fc2 = new Fraction(-25, 7);
```

```java
        fc2.setDenominator(1);
        int result2 = fc2.getDenominator();

        assertTrue("getDenominator() returned " + result2 + " instead of 1.",
                    result2 == 1);
    }

    @Test
    public void test_Numerator_set() {
        fc2 = new Fraction(-25, 7);

        fc2.setNumerator(6);
        int result2 = fc2.getNumerator();

        assertTrue("getNumerator() returned " + result2 + " instead of 6.",
                    result2 == 6);
    }

    public Fraction getFractionFromFile(String filepath) throws IOException {
        System.out.println(filepath);
        String[] arrayNumbers = getNumbersFromFile(filepath);
        for (String elem : arrayNumbers)
            System.out.println(elem);

        return new Fraction(Integer.parseInt(arrayNumbers[0]),
                Integer.parseInt(arrayNumbers[1]));
    }

    public String[] getNumbersFromFile(String numbersFile) throws IOException
    {
        int n = 0;
        BufferedReader in = new BufferedReader(new FileReader(numbersFile));
        while ((in.readLine()) != null) {
            n++;
        }
        in.close();

        String[] la=new String[n];
        String s = new String();
        int i = 0;
        in = new BufferedReader(new FileReader(numbersFile));
        while ((s=in.readLine()) != null) {
            la[i] = s;
            i++;
        }
        in.close();
        return la;
    }
}
```