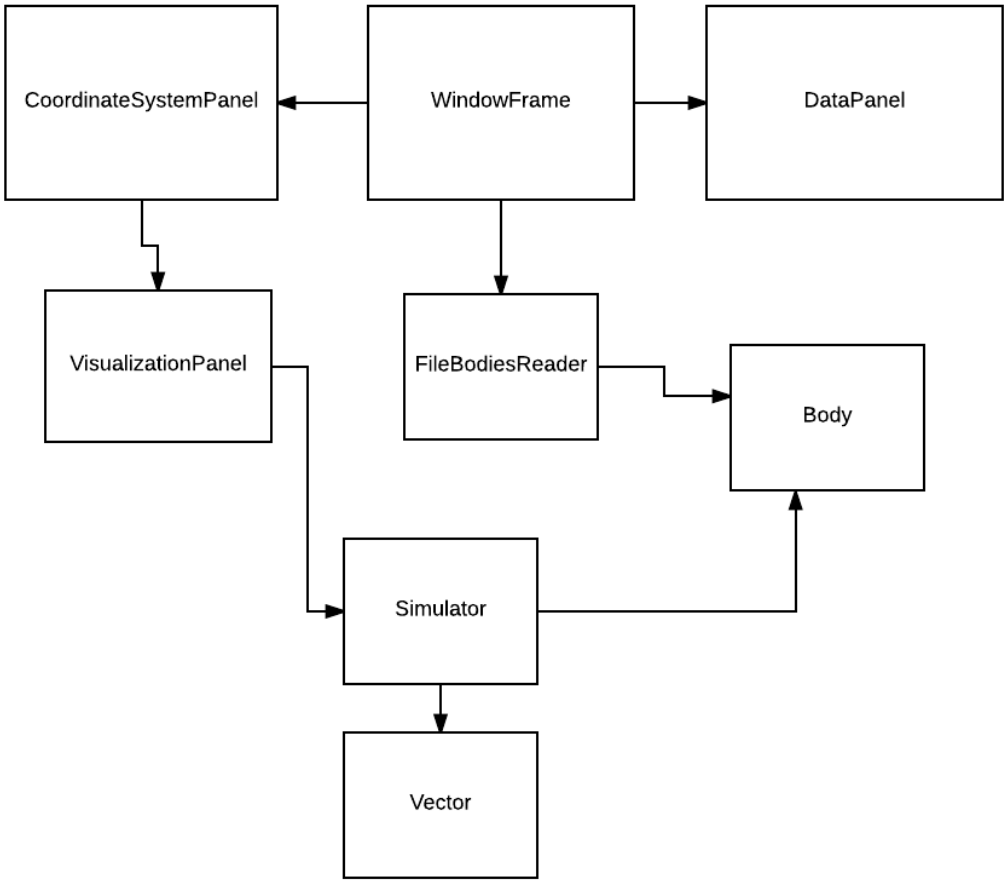
Sprawozdanie

Filip Choromański
Mateusz Karwowski

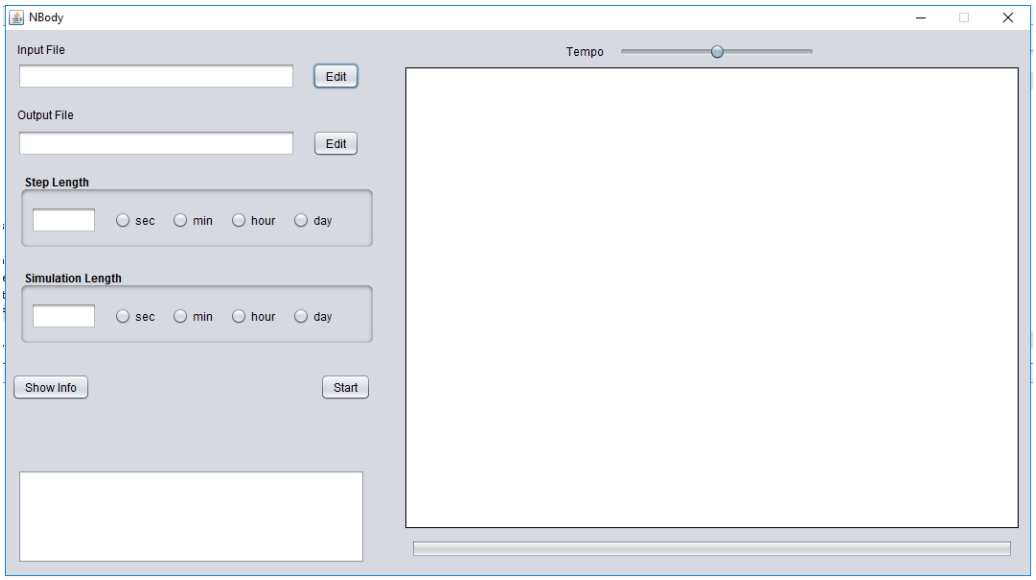
1 Podział programu

Podział programu na moduły zaprezentowany za pomocą diagramu.



2 Jak uruchomić program

Po przygotowaniu pliku z danymi należy uruchomić program. Ukaże nam się okno programu:



Należy wybrać plik z danymi wejściowymi oraz wybrać ustawienia symulacji, następnie nacisnąć przycisk Start.

Program na bieżąco wizualizuje obliczone kolejne położenia podanych ciał. Można też (opcjonalnie) zapisać je do pliku.

3 Testy

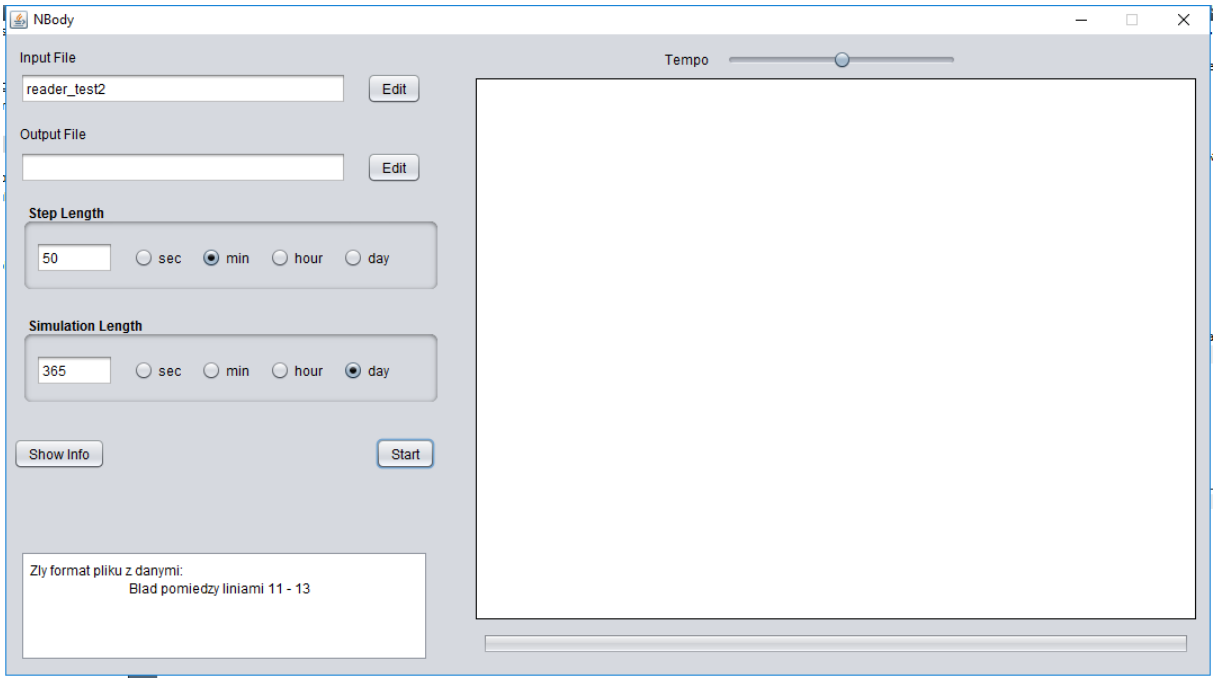
Poszczególne moduły testowaliśmy, tworząc na końcach plików z kodem źródłowym funkcje main, w której wywoływaliśmy funkcje danego modułu, bądź też korzystaliśmy z metody testów jednostkowych.

3.1 FileBodiesReader

Moduł testowaliśmy na poprawnych i błędnych danych.

```
4
124153.12312
12312.123 9854 218652
566532 6265 662656
2535148.59595
5959 9595 9595
5118 5484 21845
1
59565 48454 54545
51544 21481 11845
3
59562 65626
1518 18152
```

Plik z błędnymi danymi (Brakuje współrzędnej z położenia i prędkości czwartego ciała).



Komunikat o błędzie w tym przypadku.

3.2 Vector

W celu testowania tego modułu sporządziliśmy klasę testującą:

```
1  package Logic;
2
3  import static java.lang.Math.sqrt;
4  import org.junit.After;
5  import org.junit.Before;
6  import org.junit.Test;
7  import static org.junit.Assert.*;
8
9  /**
10  *
11  * @author Mateusz
12  */
13 public class VectorTest {
14
15     private Vector myinstance;
16
17     @Before
18     public void setUp() {
19         myinstance = new Vector(3, 5, 6);
20     }
21
22     @After
23     public void tearDown() {
24         myinstance = null;
25     }
26
27     /**
28      * Test of addVectors method, of class Vector.
29      */
30     @Test
31     public void testAddVectors() {
32         System.out.println("addVectors");
33         Vector v = new Vector(2, 4, 5);
```

```
34          Vector instance = myinstance;
35          assertEquals(instance.addVectors(v), new Vector(5, 9, 11));
36      }
37
38      /**
39       * Test of substractVectors method, of class Vector.
40       */
41      @Test
42      public void testSubstractVectors() {
43          System.out.println("substractVectors");
44          Vector v = new Vector(5, 7, 8);
45          Vector instance = myinstance;
46          assertEquals(instance.substractVectors(v), new Vector(-2, -2, -2));
47      }
48
49      /**
50       * Test of multiplyVector method, of class Vector.
51       */
52      @Test
53      public void testMultiplyVector() {
54          System.out.println("multiplyVector");
55          double scalar = 3.0;
56          Vector instance = myinstance;
57          assertEquals(instance.multiplyVector(scalar), new Vector(9, 15, 18));
58      }
59
60      /**
61       * Test of vectorLength method, of class Vector.
62       */
63      @Test
64      public void testVectorLength() {
65          System.out.println("vectorLength");
66          Vector instance = myinstance;
67          assertEquals(instance.vectorLength(), sqrt(70), 0.0);
68      }
69
70  }
```

4 Prezentacja przeprowadzanej symulacji