

28 styczeń 2020

Dokumentacja projektu

Programowanie III

Paweł Pietraszko, IIIF

Spis treści:

1. Opis programu,
 - a) Temat
 - b) Opis działania oraz zrzuty ekranu
2. Opis algorytmu
 - a) Schemat blokowy
 - b) Budowa/struktura programu

1. Opis programu

Temat

Moim zadaniem było napisać program, który sprawdzi, czy podane 2 liczby naturalne są zaprzyjaźnione, tzn. kiedy suma dzielników jednej liczby (z jedynką, bez niej samej, tj. np. dla 24 dzielnikami są liczby: 1, 2, 3, 4, 6 i 12; suma = $1+2+3+4+6+12=28$) jest równa drugiej i odwrotnie - suma dzielników drugiej równa się pierwszej.

Inspiracją do projektu zaczerpnięta została z zestawów zadań algorytmicznych, a konkretniej jest to zadanie 5 z edycji 2013:



POLITECHNIKA ŚLĄSKA

Wydział Matematyki Stosowanej

Studenckie Koło Naukowo-Informatyczne „Link”

ul. Kaszubska 23, 44-100 Gliwice



obszaru do pola całego obszaru. Z tej zależności da się znaleźć (pośrednio lub w przybliżony sposób) wartość r .

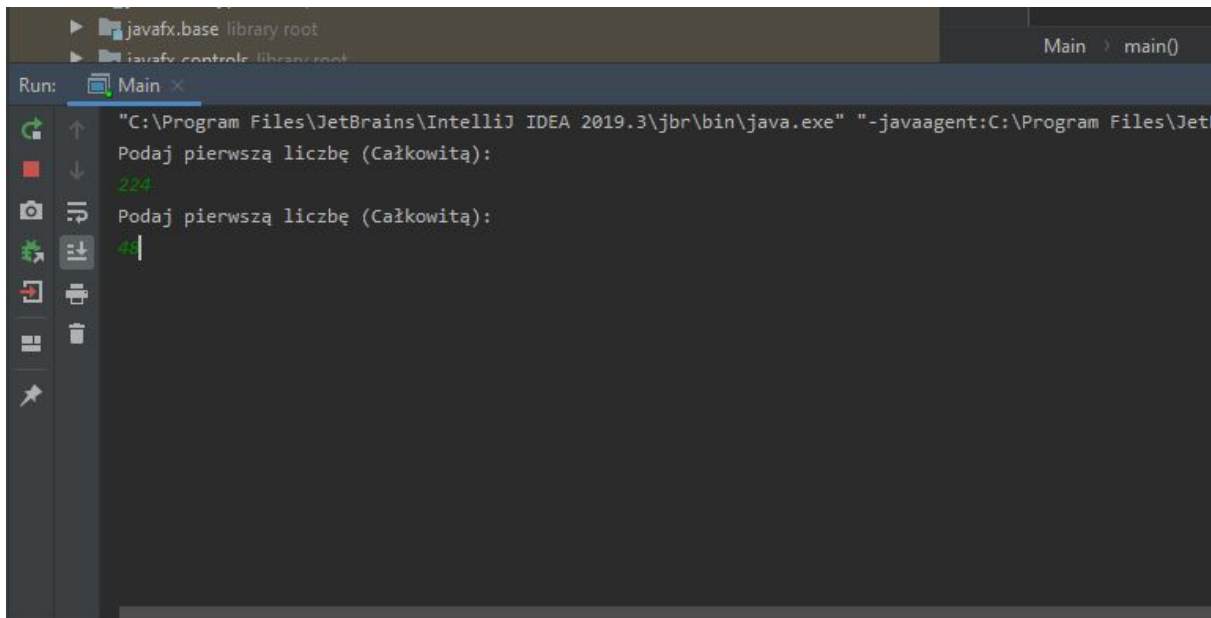
ZADANIE 5 - "LICZBY ZAPRZYJAŹNIONE"

Za Wikipedią: "Liczby zaprzyjaźnione to para różnych liczb naturalnych, takich, że suma dzielników każdej z tych liczb równa się drugiej (nie uwzględniając tych dwóch liczb jako dzielników)." Np. liczba 284 ma dzielniki: 1, 2, 4, 71, 142, których suma daje 220, a liczba 220 ma dzielniki: 1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110, których suma daje 284. Zatem liczby 220 i 284 tworzą parę liczb zaprzyjaźnionych. Należy napisać program, który dla dowolnej pary różnych liczb naturalnych będzie rozstrzygał, czy para ta tworzy liczby zaprzyjaźnione.

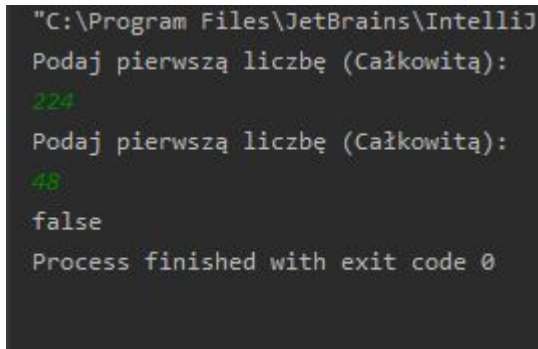
Obsługa programu

Kod pisany był na komputerze z system *Windows 10.0.18363.592*, w programie *IntelliJ IDEA 2019.3*.

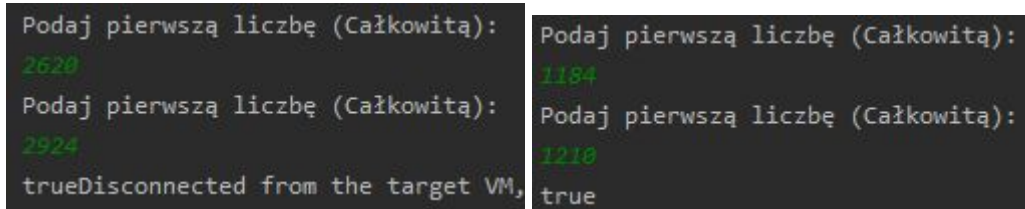
Po uruchomieniu programu zostajemy poproszeni o wprowadzenie 2 liczb całkowitych.



Następnie program zwraca wartość bool'owską: true - liczby podane są zaprzyjaźnione, false - liczby nie są zaprzyjaźnione.



Liczbami zaprzyjaźnionymi, które udało mi się znaleźć, są to pary:(
(220, 284), (1184,1210), (2620, 2924).

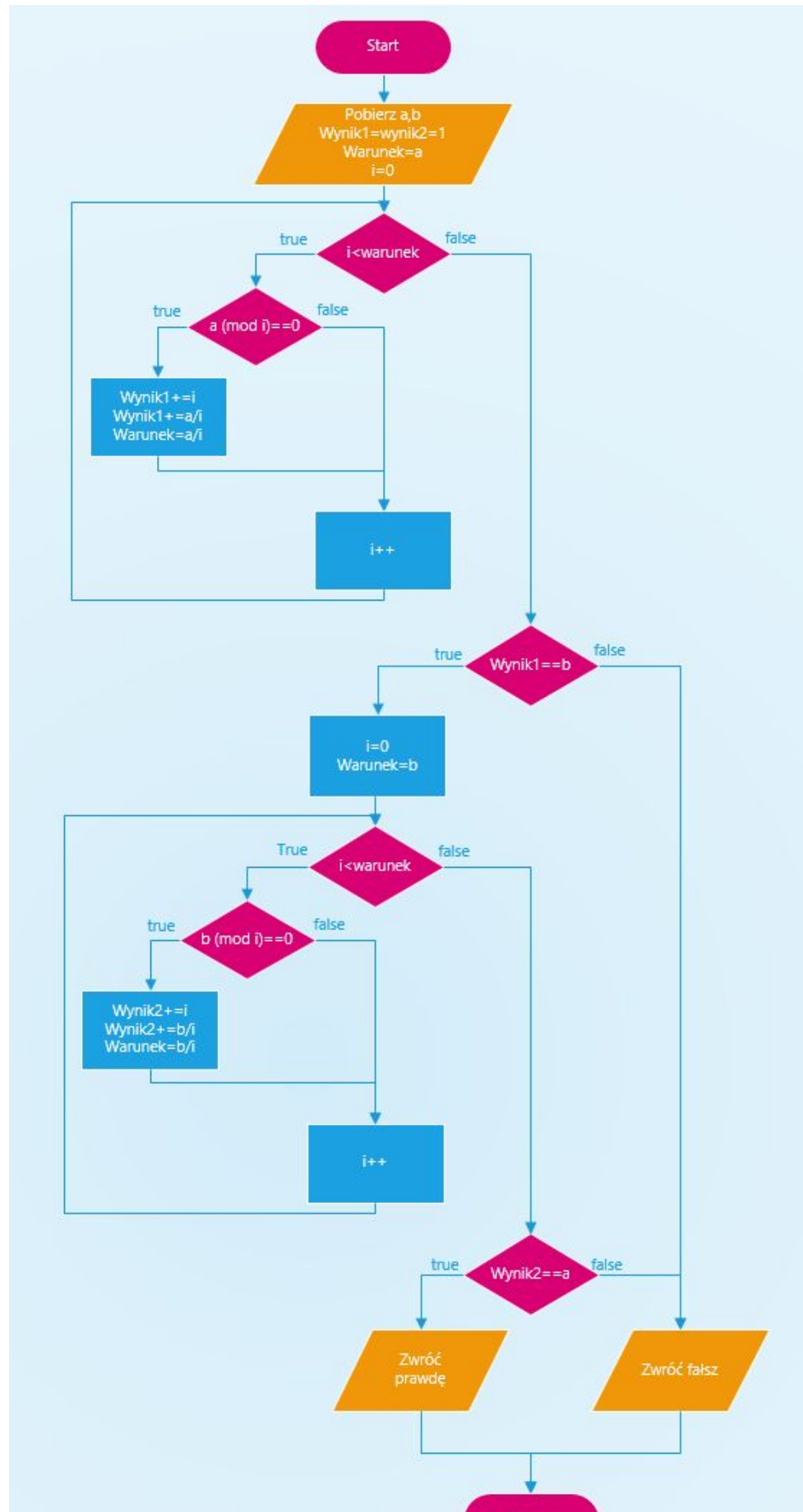


Niestety, "liczby zaprzyjaźnione" najprawdopodobniej nie jest terminem obowiązującym w świecie matematyki, co uniemożliwiło mi wsparcie się internetem. Powyższe liczby zostały

znalezione metodą BruteForce na przedziale [1;5000]. Można zauważyć, że jest ich bardzo mało.

2. Opis algorytmu

Schemat blokowy



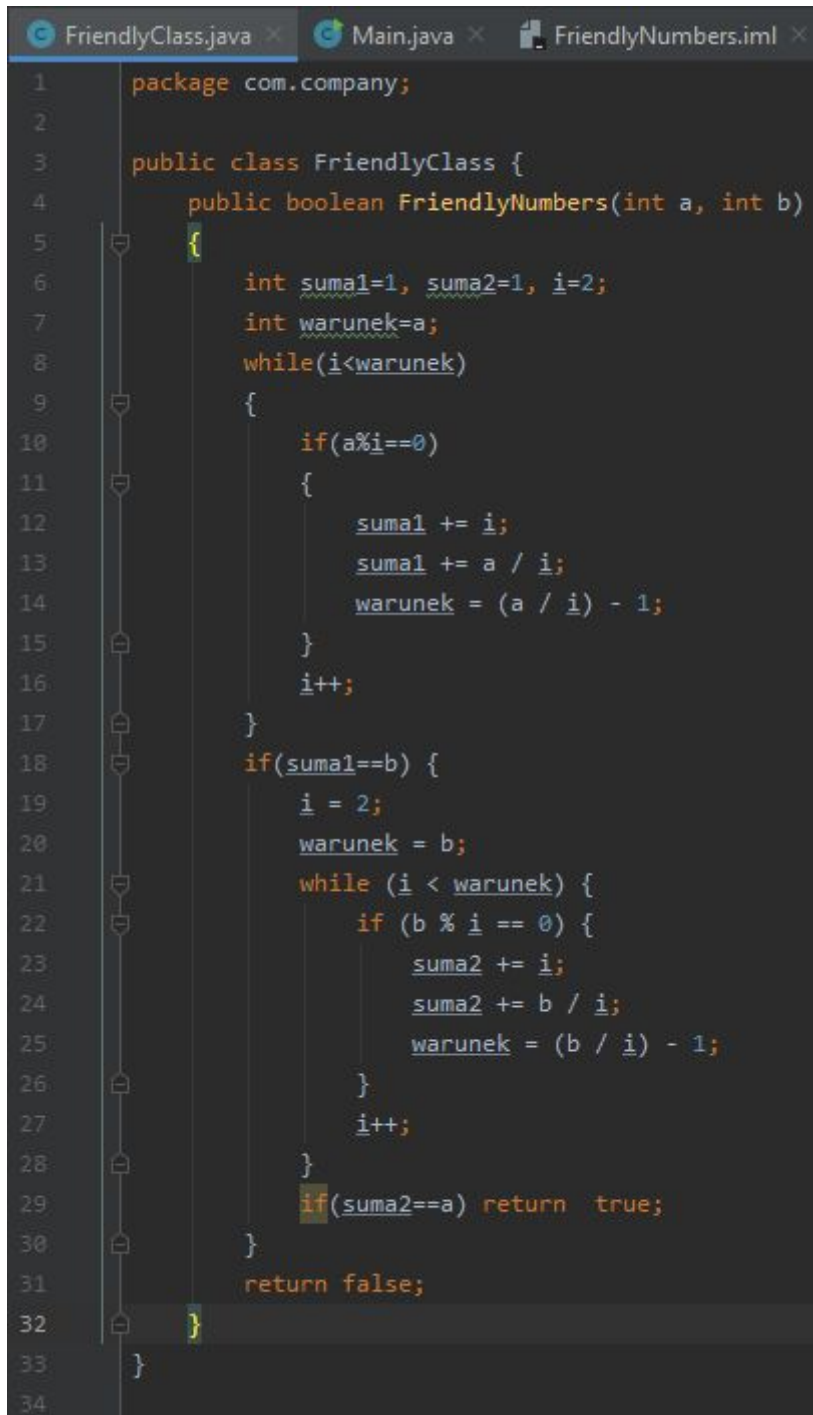
Opis schematu

Jak widać, program w pierwszej kolejności sprawdza sumę dzielników pierwszej liczby i porównuje tę sumę z drugą liczbą. Jeżeli wynik okaże się negatywny, program od razu zwraca false. Dopiero w przypadku, kiedy wynik jest pozytywny, program sprawdza drugą liczbę. Dzięki samemu temu zabiegowi wydajność algorytmu znacząco wzrosła, gdyż wcześniej liczył on obie sumy dzielników a dopiero następnie porównywał z liczbami.

Kolejnym zabiegiem optymalizacyjnym jest szukanie dzielników na podstawie par, tj jeżeli liczba ' i ' dzieli liczbę ' a ', to wynikiem tego działania jest liczba ' a/i ', wtedy wiemy, że liczba ' a ' podzielona przez ' a/i ' zwróci nam ' i '. Automatycznie możemy zmniejszyć zakres, w którym szukamy dzielnika z $a-1$ do $a/i - 1$, czyli w większości przypadków program zwiększa wydajność nawet kilkadziesiąt razy.


Budowa/struktura programu

Funkcja, a dokładniej metoda sprawdzająca, czy znajduje się w klasie FriendlyClass, która została utworzona w osobnym pliku *FriendlyClass.java*.



```
1 package com.company;
2
3 public class FriendlyClass {
4     public boolean FriendlyNumbers(int a, int b)
5     {
6         int suma1=1, suma2=1, i=2;
7         int warunek=a;
8         while(i<warunek)
9         {
10             if(a%i==0)
11             {
12                 suma1 += i;
13                 suma1 += a / i;
14                 warunek = (a / i) - 1;
15             }
16             i++;
17         }
18         if(suma1==b) {
19             i = 2;
20             warunek = b;
21             while (i < warunek) {
22                 if (b % i == 0) {
23                     suma2 += i;
24                     suma2 += b / i;
25                     warunek = (b / i) - 1;
26                 }
27                 i++;
28             }
29             if(suma2==a) return true;
30         }
31         return false;
32     }
33 }
34
```

W głównym pliku, tj *Main.java*, został utworzony obiekt *fr* na podstawie powyższej klasy. Do pobierania zmiennych służy funkcja zawarta w głównym pliku o nazwie *getInt()*, która weryfikuje, czy użytkownik wprowadził poprawne dane, wczytane za pomocą *Scanner'a*. Program jest stosunkowo bardzo prosty, dlatego dużo uwagi poświęciłem optymalizacji i estetyce.



```
1 package com.company;
2
3 import ...
4
5
6 public class Main {
7     public static void main(String []args)
8     {
9         Locale locale = new Locale( language: "pl", country: "PL");
10        FriendlyClass fr = new FriendlyClass();
11        int a,b;
12        System.out.println("Podaj pierwszą liczbę (Całkowitą): ");
13        a=getInt();
14        System.out.println("Podaj pierwszą liczbę (Całkowitą): ");
15        b=getInt();
16        System.out.print(fr.FriendlyNumbers(a,b));
17    }
18
19    static int getInt(){
20        Scanner input = new Scanner(System.in);
21        int a=0;
22
23        try {
24            a = input.nextInt();
25        }
26        catch(Exception e)
27        {
28            System.out.println("podana wartość jest błędna");
29        }
30
31        return a;
32    }
33 }
34
```