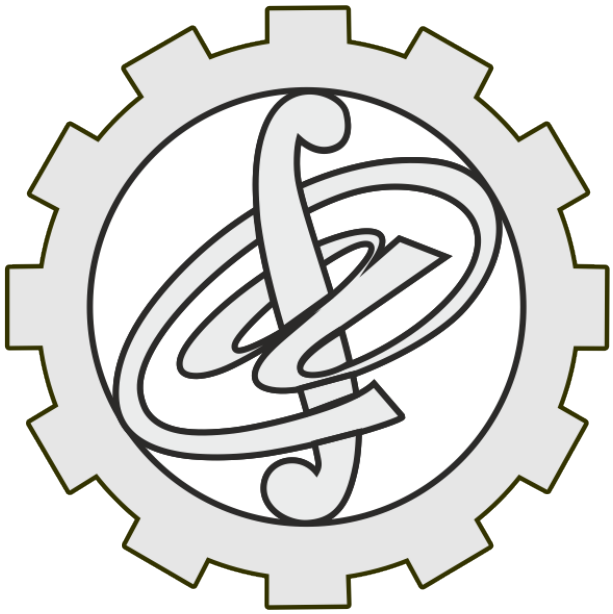
Programowanie III

Projekt Zaliczeniowy - Algorytmion 2011
Zadanie "Monety"

Politechnika Śląska
Wydział Matematyki Stosowanej

Radosław Terelak

Gliwice, 2022



Spis treści

1 Temat projektu

1.1 Treść i problematyka

Mój projekt zaliczeniowy to rozwiązanie zadania **"Monety"** z **Algorytmionu 2011.**

Treść zadania:

Na placu targowym umieszczono **X worków**, w których w każdym z nich znajdowało się **Y złotych monet**. Każda złota moneta waży **3 gramy**. Fałszerz monet podmienił jeden worek, zamieniając w nim wszystkie monety na podrobione. Fałszywa moneta waży **2 gramy**.

Sprzedawca chce znaleźć worek z fałszywymi monetami. Może to zrobić tylko w ten sposób, że wyciąga dowolną ilość monet z każdego worka i waży je złożone razem tylko jeden raz.

Program wczytuje ze zbioru **dane.txt** ilość worków znajdującą się na targu, ilość monet wyciągniętych przez sprzedawcę kolejno ponumerowanych worków oraz wagę wszystkich wyciągniętych monet.

Na podstawie wagi stwierdza, w który worku znajdują się fałszywe monety i zapisuje numer worka z fałszywymi monetami do pliku **wynik.txt** albo w przypadku nie możliwości określenia numeru worka wpisuje komunikat **"dokonaj wyciągnięcia monet jeszcze raz".**

Wejście: Pierwszy wiersz zawiera liczbę całkowitą określającą **ilość worków** i zakończoną znakiem nowej linii. Druga linia zawiera **n liczb całkowitych** i każda z nich jest ilością wyciągniętych monet przez sprzedawcę z ponumerowanych worków od 1 do X. W ostatnim wierszu znajduje się **waga** wszystkich wyciągniętych monet, prawdziwych i fałszywych.

Wyjście: Liczba typu całkowitego określająca numer worka z fałszywymi monetami albo napis "dokonaj wyciągnięcia monet jeszcze raz".

Problematyka:

Problem ukazany w zadaniu to określenie fałszywości worka na podstawie jego wagi. Aby go rozwiązać należy na początku policzyć wagę, gdyby wszystkie wyciągnięte monety były prawdziwe. Pozwoli to określić czy istnieje fałszywy worek. Następnie należy ważyć monety z każdego kolejnego worka wagą monety fałszywej i dodając wagę pozostałych monet o wadze monety złotej. Jeśli suma tych wag jest równa wartości zapisanej w pliku, worek fałszywy został znaleziony.

2 Dane

2.1 Dane wejściowe

Jako dane wejściowe przygotowałem 9 plików. Pierwsze 3 zawierają fałszywy worek. Kolejny plik zawiera dane, w którym nie istnieje fałszywy worek. Następne zawierają błędy, a ostatni posiada taki zestaw danych, który uniemożliwia określenie worka fałszywego.

2.2 Dane wyjściowe

Plik wynikowy zostanie utworzony jeśli:
- dane znajdujące się w pliku wejściowym są poprawne,
- można określić, który worek jest fałszywy,
- można określić, że nie istnieje worek fałszywy.

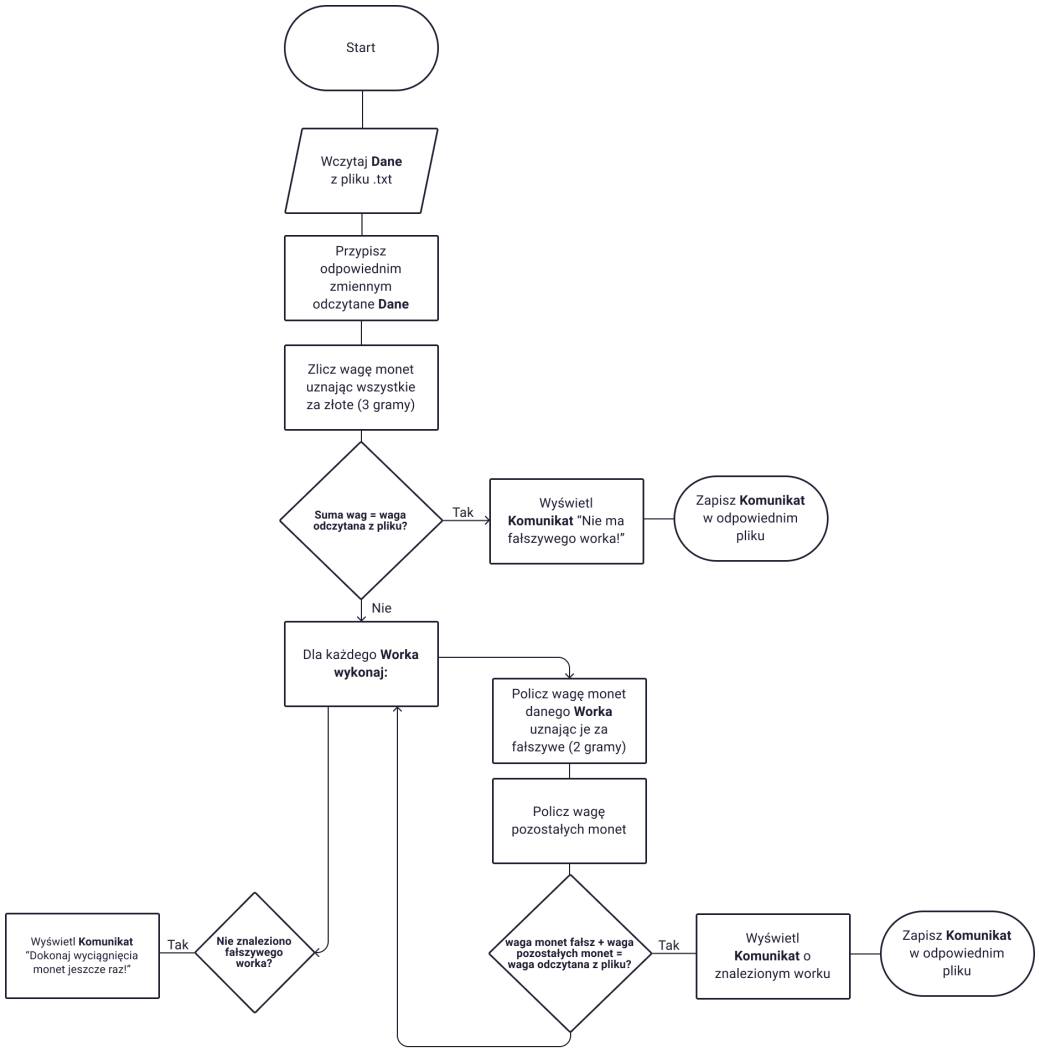
W pliku wynikowym znajduje się odpowiedni komunikat.
Jeśli dane są poprawne, ale nie można określić worka fałszywego na ekranie wyświetli się komunikat proszący o ponowne wyciągnięcie monet.

3 Algorytm

3.1 Opis słowny

1. Wczytaj dane z pliku .txt.
2. Jeśli dane są zgodne: odpowiednim zmiennym przypisz wartości z pliku.
3. Sprawdź czy w podanych danych zawiera się worek fałszywy. Jeśli nie: wyświetl odpowiedni komunikat i zapisz wynik.
4. Dla każdego worka:
   - Policz wagę monet danego worka uznając je za fałszywe.
   - Policz wagę pozostałych monet.
   - Porównaj sumę tych wartości z wagą odczytaną z pliku.
5. Jeśli znaleziono worek fałszywy: zapisz odpowiedni komunikat do pliku wynikowego.
6. Jeśli nie udało się znaleźć worka fałszywego: wyświetl komunikat aby dokonać wyciągnięcia monet jeszcze raz.

3.2 Schemat blokowy



3.3 Zastosowane funkcje i struktury w programie

W pliku źródłowym znajdują się 3 funkcje.

Funkcja wczytajDane:

Ta funkcja przeszukuje folder wejściowy w poszukiwaniu plików .txt. Następnie otwiera każdy plik i odczytane wartości przypisuje do odpowiednich zmiennych. Wypisuje na ekran odczytane dane. Sprawdza poprawność danych, jeśli znajdzie błąd to wyświetli odpowiedni komunikat. Sprawdza również, czy ilość worków jest równa ilości razy wyciągnięcia monet.

Funkcja sprawdzDane:

Jest to funkcja analizująca wczytane dane i szukająca worka fałszywego. Na początku sprawdza, czy w ogóle worek fałszywy występuje porównując wczytaną wagę łączną monet z rzeczywistą wagą monet przyjmując, że każda jest złota. Jeśli może wystąpić sprawdza każdy kolejny worek i porównuje wartości wagowe. Po analizie danych wyświetla odpowiedni komunikat. Jeśli znaleziono worek fałszywy, komunikat zostaje zapisany w odpowiednim pliku wynikowym.

Funkcja zapiszWynik:

Ta funkcja tworzy odpowiedni plik w folderze wynikowym, którego numer odpowiada numerowi pliku wejściowego. Zapisuje w nim komunikat przekazany przez funkcję **sprawdzDane.** Wyświetla informację o pomyślnym, bądź nieudanym zapisie danych.

3.4 Kod źródłowy

```
1  // Projekt zaliczeniowy Radoslaw Terelak 4G
2  // Algorytmion 2011 - Zadanie 1 "Monety"
3
4  package projekt_terelak;
5
6  import java.util.*;
7  import java.io.*;
8
9  public class Projekt_Terelak {
10
11     public static void wczytajDane() throws FileNotFoundException,
           IOException{
12         System.out.println("Wczytuje dane..");
13         File folder_in = new File("IN");
14         Scanner scan = new Scanner(System.in);
15
16         FilenameFilter textFilter = new FilenameFilter() {
17             public boolean accept(File dir, String name) {
18                 return name.toLowerCase().endsWith(".txt");
19             }
20         };
21
22         File[] dane = folder_in.listFiles(textFilter);
23         for (File plik : dane) {
24             System.out.println("\nOtwieram plik: " + plik.
                   getCanonicalPath());
25             scan = new Scanner(plik);
26
27             try{
28                 char nr_pliku = plik.getName().charAt(5);
29                 int ile_workow = Integer.parseInt(scan.nextLine());
30                 String[] monety = scan.nextLine().split(";");
31                 int waga = Integer.parseInt(scan.nextLine());
32
33                 System.out.println("Nr pliku: " + nr_pliku + " Liczba
                       workow: " + ile_workow + " Waga: " + waga);
34                 System.out.print("Ilosc wyciagnietych monet: ");
35                 for (String moneta : monety) {
36                     System.out.print(moneta + ", ");
37                 }
38                 if(monety.length == ile_workow){
39                     sprawdzDane(ile_workow, monety, waga, nr_pliku);
40                 }
41                 else{
42                     System.out.print("\nPodano zle dane! \n");
```

```
43                  }
44              }
45              catch(Exception e){
46                  System.out.print("Podano zle dane! " + e + "\n");
47              }
48          }
49      }
50
51      public static void sprawdzDane(int ile_workow, String[] monety,
            int waga, char nr_pliku) throws FileNotFoundException {
52          try{
53              System.out.println("\nSprawdzam dane..");
54              int suma_wag = 0;
55              boolean znaleziono = false;
56
57              for (String moneta : monety) {
58                  if(Integer.parseInt(moneta) <= 0){
59                      System.out.println("Podano nieprawidlowe dane!");
60                      return;
61                  }
62                  else{
63                      suma_wag += Integer.parseInt(moneta) * 3;
64                  }
65              }
66
67              if(suma_wag == waga){
68                  System.out.println("Nie ma worka z falszywymi
                        monetami!");
69                  zapiszWynik("Nie ma worka z falszywymi monetami!",
                        nr_pliku);
70                  return;
71              }
72
73              for(int i=0; i<ile_workow; i++){
74                  int waga_falsz = Integer.parseInt(monety[i]) * 2;
75                  int waga_pozostale = 0;
76                  int nr_worka = i + 1;
77
78                  for(int j=0; j<ile_workow; j++){
79                      if(j != i){
80                          waga_pozostale += Integer.parseInt(monety[j])
                                * 3;
81                      }
82                  }
83                  if(waga_falsz + waga_pozostale == waga){
84                      znaleziono = true;
85                      System.out.println("Falszywym jest worek o
                            numerze: " + nr_worka);
```

```
86                      zapiszWynik("Falszywym␣jest␣worek␣o␣numerze:␣" +
                           nr_worka, nr_pliku);
87                      return;
88                  }
89              }
90              if(znaleziono == false){
91                  System.out.println("Dokonaj␣wyciagniecia␣monet␣
                       jeszcze␣raz!");
92              }
93          }
94          catch(Exception e){
95              System.out.println("Podano␣nieprawidlowe␣dane!␣" + e);
96          }
97      }
98
99      public static void zapiszWynik(String komunikat, char nr_pliku)
           throws FileNotFoundException{
100         try{
101             System.out.println("Wynik..");
102             new File("OUT").mkdir();
103             String plik_wynikowy = "wynik_" + nr_pliku + ".txt";
104             File wynik = new File("OUT\\" + plik_wynikowy);
105             FileWriter zapisz = new FileWriter("OUT\\" +
                   plik_wynikowy);
106             wynik.createNewFile();
107             zapisz.write(komunikat);
108             zapisz.close();
109             System.out.println("Wynik␣zapisano␣w␣pliku:␣" + wynik.
                   getAbsolutePath());
110         }
111         catch(IOException e){
112             System.out.println("\nNie␣udalo␣sie␣zapisac␣wyniku␣do␣
                   pliku␣:(");
113             System.out.println(e);
114         }
115     }
116
117     public static void main(String[] args) throws
           FileNotFoundException, IOException{
118         wczytajDane();
119     }
120 }
```

4 Test na poprawność działania programu

4.1 Zastosowane zabezpieczenia

Na przestrzeni całego programu zadbałem o to by wyeliminować podawanie niepoprawnych danych. W każdej funkcji znajduje się **try** i **catch** walidujący zgodność danych i zachodzących procesów.

```
1 if(monety.length == ile_workow){
2     sprawdzDane(ile_workow, monety, waga, nr_pliku);
3 }
4 else{
5     System.out.print("\nPodano zle dane! \n");
6 }
```

W funkcji **wczytajDane** weryfikuję, czy ilość podanych worków zgadza się z ilością wyciągnięc monet z worków.

```
1 for (String moneta : monety) {
2     if(Integer.parseInt(moneta) <= 0){
3         System.out.println("Podano nieprawidlowe dane!");
4         return;
5     }
6     else{
7         suma_wag += Integer.parseInt(moneta) * 3;
8     }
9 }
```

W funkcji **sprawdzDane** weryfikuję, czy wczytana każda wartość wyciągniętych monet jest dodatnia.

4.2 Analiza wyników

Nr pliku	Dane wejściowe	Wynik
1	10 1;2;3;4;5;6;7;8;9;10 155 Waga gdyby wszystkie monety były złote: 165	Faszywym jest worek o numerze: 10
2	10 1;2;3;4;5;6;7;8;9;10 163 Waga gdyby wszystkie monety były złote: 165	Faszywym jest worek o numerze: 2
3	3 33;44;88 462 Waga gdyby wszystkie monety były złote: 495	Faszywym jest worek o numerze: 1
4	7 5;3;4;5;1;1;4 69 Waga gdyby wszystkie monety były złote: 69	Nie ma worka z faszywymi monetami!
5	5 33;44;88 462 Błąd ilości worków względem podanych wyciągnieć	-
6	7 5;-3;4;6;1;5;4 83 Błąd użycia -3 jako ilość wyciągniętych monet	-
7	7 5;3;4;6;0;5;4 75 Błąd użycia 0 jako ilość wyciągniętych monet	-
8	7 5;X;4;6;5;2;4 73 Błąd znaku 'X' zamiast cyfry	-
9	7 5;1;4;6;5;2;4 150 Błąd za dużej wagi	-

5 Podsumowanie

5.1 Wnioski i wrażenia

Samo zadanie od strony logicznej nie stworzyło mi wielu trudności. Dość szybko zrozumiałem problem i przeszedłem do implementacji. Stopniowo eliminowałem błędy i dodawałem dokładniejsze zabezpieczenia. Od początku kod podzieliłem na kilka funkcji, co zwiększyło przejrzystość i ułatwiło stworzenie działającego programu. Z pewnością dzięki stworzeniu tego projektu utrwaliłem wiedzę z języka Java oraz nauczyłem się nowych rzeczy.