Faliński Emil **| Grupa czwartek 7:30 | ISX 2.5**

Temat programu: Zamiana liczb rzymskich na arabskie oraz arabskich na rzymskie.

Dane:

Danymi będą liczby zapisane w systemie rzymskim, które następnie zgodnie z algorytmem programu zostaną przekonwertowane na system dziesiętny. Drugą możliwością którą program będzie oferował to zamiana wprowadzonych danych w postaci liczb dziesiętnych które zostaną zamienione na liczby zapisane w systemie rzymskim.

Użytkownik programu będzie mógł poruszać się po wyświetlonym menu poprzez wpisanie określonych cyfr przyporządkowanym określonym funkcjom. Innymi danymi będą ścieżki do plików wprowadzane również przez korzystającego z programu.

Wyniki:

Wynikami będą:

- Liczby zapisane w systemie *RZYMSKIM* [o typie danych String] ; jeśli będziemy chcieli zamienić liczbę z systemu dziesiętnego na rzymski,
- Liczby zapisane w systemie *DZIESIĘTNYM* [o typie danych int]; jeśli będziemy chcieli zamienić liczbę z systemu rzymskiego na dziesiętny.
- Pliki z danymi przekonwertowanymi (pliki tekstowe).

Algorytm rozwiązania:

Własności rzymskiego systemu zapisywania liczb:

- Nie pozwala na zapis ułamków.
- W zapisie rzymskim nie istnieje cyfra 0 [zero].
- W zapisie liczby wykorzystywane jest 7 znaków. **I, V, X, L, C, D, M.**
- Nie można znaków jedności (I), dziesiątek (X), setek (C) oraz tysięcy (M) używać więcej niż 3 razy. Znaki (V)-pięć, (L)-pięćdziesiąt, (D)-pięćset można użyć tylko raz w zapisie.
- Znaki które można odejmować od znaków o większej wartości to **I, X, C,** ale tylko jeśli znak stojący po prawej stronie (ten większy), nie jest od niej większy więcej niż 10 razy.
- Zakres liczb jakie mogą być zapisane w systemie rzymskim to od 1 do 3999 w systemie dziesiętnym.

Algorytm :

Musimy zadeklarować dwie tablice do których wpiszemy w 1) musi być typu String i zawierać kolejne znaki w systemie rzymskim czyli M, CM, D, CD, C, XC, L, XL, X, IX, V, IV, I. W drugiej tablicy wpiszemy odpowiadające pierwszej tablicy wartości systemu dziesiętnego czyli 1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1, tablica będzie typu Integer(całkowite liczby). Kolejnym punktem programu będzie wpisanie przez użytkownika liczby zapisanej w systemie rzymskim lub arabskim. Następnie program musi sprawdzić czy podane przez użytkownika dane są prawidłowe np. zgodnie z własnościami systemu rzymskiego zakres liczb możliwych do zapisania to od 1 do 3999.

- **Rzymskie→Dziesiętne.**

Zaczynamy badanie wprowadzonej liczby rzymskiej przesuwając się od lewej strony do prawej dodając do siebie kolejne składniki z tym ,że szczególną uwagę należy zwrócić na pozycje w których liczba 100 [C] poprzedza liczbę 500 [D] lub 1000 [M].
Sprawdzamy zgodność badanego znaku i symbolu rzymskiego poczynając od największej wartości. Dopóki nie przeanalizowaliśmy wszystkich znaków liczby do przekonwertowania.

Jeżeli aktualnie badany znak zgodny jest z aktualnym symbolem to dodaj do wyniku wartość tego symbolu, do zmiennej utworzonej wcześniej typu Integer „startindex" z wartością domyślną 0 dodaj długość sprawdzanych znaków (zgodnie z tablicą typu String rzymskie) .

W przeciwnym razie, jeżeli aktualnie rozpatrywany symbol jest postaci 10xx i w liczbie do przekonwertowania pozostały znaki znak do analizy to sprawdź czy dwa kolejne znaki pasują do wzorca postaci 900. Jeżeli tak, to dodaj wartość wzorca postaci 9xx do wyniku, przejdź do analizy kolejnego znaku po wzorcu 900 i dodaj do zmiennej „startindex" długość znaków z tablicy rzymskie z odpowiedniej komórki(czytanej). Przejdź do rozpatrywania kolejnego symbolu (o wartości mniejszej).

W przeciwnym razie, jeżeli aktualnie rozpatrywany symbol jest postaci 500 i w liczbie do przekonwertowania pozostało więcej niż jeden znak do analizy to sprawdź czy dwa kolejne znaki pasują do wzorca postaci 400. Jeżeli tak, to dodaj wartość wzorca postaci 400 do wyniku, do zmiennej „startindex" długość znaków z tablicy rzymskie z odpowiedniej komórki(czytanej), przejdź do analizy kolejnego znaku po wzorcu 400. Przejdź do rozpatrywania kolejnego symbolu (o wartości mniejszej).

W przeciwnym wypadku przejdź do rozpatrywania kolejnego symbolu (o wartości mniejszej).

- **Dziesiętne→Rzymskie.**

Sprawdzamy możliwość dodania symbolu rzymskiego poczynając od największej wartości. Dopóki zadana liczba jest większa od zera:

Jeżeli zadana liczba jest większa bądź równa, wartości aktualnie rozpatrywanego symbolu to dodaj ten symbol do wyniku, a od liczby odejmij jego wartość.

W przeciwnym wypadku jeżeli aktualnie rozpatrywany symbol jest postaci 1000, to rozpatrz możliwość dodania symbolu 900 - jeżeli liczba jest większa bądź równa 900, to dodaj reprezentację 900 do wyniku, a jego wartość odejmij od liczby, którą konwertujemy. Przejdź do rozpatrywania kolejnego symbolu (o wartości mniejszej)

W przeciwnym wypadku jeżeli aktualnie rozpatrywany symbol jest postaci 500, to rozpatrz możliwość dodania symbolu 400 - jeżeli liczba jest większa bądź równa 400, to dodaj reprezentację 400 do wyniku, a jego wartość odejmij od liczby, którą konwertujemy. Przejdź do rozpatrywania kolejnego symbolu (o wartości mniejszej)

W przeciwnym wypadku przejdź do rozpatrywania kolejnego symbolu (o wartości mniejszej).

Przykład:

W celu lepszego zobrazowania przykładu wykorzystam zadeklarowane przeze mnie tablice:

```
private final String [] tab_rzymskie = {"M" , "CM", "D" , "CD", "C", "XC", "L", "XL",
"X", "IX", "V", "IV", "I" };

private final int [] tab_arabskie =    {1000, 900 , 500 , 400 , 100, 90  , 50 ,  40 ,
10 ,   9 ,  5 ,  4  ,  1  };
```

a) Zamiana przykładowej liczby arabskiej np. **1296**. Jako wzór posłużę się kodem który już napisałem:

```
private String doRzymskiej (1296)
 {
       String wynik_rzymska="";
       for (int i=0; i<tab_arabskie.length; i++)
       {
             while (tab_arabskie[i]<=arabska)
             {
                   wynik_rzymska+=tab_rzymskie[i];
                   arabska-=tab_arabskie[i];
             }
       }
       return wynik_rzymska;
             }
```

Pętla 'for' będzie się przemieszczała po kolejnych komórkach tablicy z arabskimi liczbami (tab_arabskie) posegregowanymi malejąco ponieważ zaczynamy czytanie od największej liczby i schodzimy do co raz mniejszych. Zadeklarowaliśmy zmienną typu Integer i=0 która pozwoli nam przechodzić do kolejnych komórek tablicy. Zwiększamy 'i' aż do momentu gdy przeszukamy całą tablice liczb arabskich. Następnie w pętli while sprawdzamy czy aktualnie czytana liczba z tablicy arabskie [od i] jest mniejsza bądź równa liczbie konwertowanej. Jeśli jest to do zadeklarowanej wcześniej zmiennej typu String dodajemy zawartość komórki (o tym samym indeksie co aktualnie sprawdzany) z tablicy z rzymskimi odpowiednikami liczb arabskich. Następnie odejmujemy od naszej konwertowanej liczby to co zamieniliśmy. Przykład:

1. Mamy liczbę arabską 1296, zaczynamy przeszukiwanie kolejnych komórek tablicy arabskie. Zmienna i przyjmuje wartość 0 przechodzimy do pętli while (tab_arabskie[0]≤1296) otrzymujemy **prawdę** ponieważ 1000≤1296 więc do zmiennej „wynik_rzymska" dodajemy 'M' oraz odejmujemy odpowiednik 'M' z tablicy arabskie czyli 1000. Nasza liczba arabska jest równa aktualnie 296. Zostajemy jeszcze przy i=0. Sprawdzamy czy while(1000≤296)? **Fałsz**.
2. Przechodzimy do kolejnej iteracji i=1, kolejną liczbą jest 900 która jest większa od 296 więc przechodzimy dalej.
3. Zmienna i=2, while (500≤296) **fałsz**, przechodzimy dalej.
4. Zmienna i=3, while (400≤296) **fałsz**, przechodzimy dalej.
5. Zmienna i=4, while (100≤296) **prawda,** a więc zatrzymujemy się żeby wykonać wnętrze pętli while. Zgodnie z „i" równym aktualnie 4. Dodajemy do „wynik_rzymska" literkę 'C' (bo tab_rzymskie[4] to 'C') i odejmujemy od 296 sto. Wynik_rzymska to aktualnie „MC" a arabska to 196. Jesteśmy jeszcze w pętli while ponieważ tab_arabskie[4] czyli 100 nadal jest mniejsze od arabskiej 196. Więc do wynik_rzymska dodajemy kolejny znak 'C' i od arabskiej odejmujemy 100. Wynik_rzymska to aktualnie „MCC" a arabska to 96.

6. Przechodzimy do instrukcji for ponieważ w pętli while 100 nie jest mniejsze od 96. Zmienna i=5 przechodzimy do while tab_arabskie[5]=90 a 90 jest mniejsze od 96 więc do „wynik_rzymska" dodajemy „XC" i od arabskiej odejmujemy 90. Wynik_rzymska to aktualnie „MCCXC" a arabska to 6. While (90≤6) **fałsz** więc przechodzimy do pętli for oraz zmiennej 'i' zwiększonej o 1 czyli i=6.
7. Zmienna i=6. Następnie przesuwamy się do pętli while gdzie 50 ma być mniejsze od 6, nie jest więc w pętli for do 'i' dodajemy 1.
8. Zmienna i=7, while (40≤6) **fałsz**, przechodzimy dalej.
9. Zmienna i=8, while (10≤6) **fałsz**, przechodzimy dalej.
10. Zmienna i=9, while (9≤6) **fałsz**, przechodzimy dalej.
11. Zmienna i=10 , 5 jest mniejsze od 6 zatrzymujemy się aby wykonać zawartość pętli while. Do wynik_rzymska dodajemy odpowiedni znak z tablicy tab_rzymskie[10] czyli V, od arabskiej równej 6 odejmujemy wartość z komórki dziesiątej tablicy arabskie czyli 6-5=1. Wynik_rzymska to aktualnie „MCCXCV" a arabska to 1. Według kodu musimy jeszcze sprawdzić czy while(5≤1) **fałsz**.
12. Kolejna iteracja i=11, while (4≤1) pętla while wyrzuca **fałsz** przechodzimy do kolejnej iteracji.
13. Zmienna i=12, while (1≤1) więc zatrzymujemy się aby wykonać zawartość pętli while czyli do wynik_rzymska dodajemy odpowiedni znak z tablicy rzymskie 'I', a od aktualnej arabskiej odejmujemy odpowiednią wartość z tablicy arabskie 1-1=0. Tak doszliśmy już do końca ponieważ tab_arabskie.length wynosi trzynaście a zmienna 'i' ma być mniejsza od 13. 'I' aktualnie wynosi 13 więc 13 nie jest mniejsze od 13 tak zakończyliśmy iteracje w pętli for. Teraz wyprowadzamy wynik Wynik_rzymska to **MCCXCVI**.

b) Zamiana przykładowej liczby rzymskiej np. **MMCDXV**. Jako wzór posłużę się kodem który już napisałem:

```
private final String [] tab_rzymskie = {"M" , "CM", "D" , "CD", "C", "XC", "L",
"XL", "X", "IX", "V", "IV", "I" };

private final int [] tab_arabskie =    {1000, 900 , 500 , 400 , 100, 90  , 50 ,
40 , 10 ,   9 ,  5 ,  4  ,  1  };

private int doArabskiej(MMCDXV)
      {
            int wynik_arabska=0;
            int startindex=0;

            for (int i=0; i<tab_rzymskie.length; i++)
            {
             while (rzymska.startsWith(tab_rzymskie[i],startindex))
                  {
                        wynik_arabska+=tab_arabskie[i];
                        startindex+=tab_rzymskie[i].length();                }
            }

            return wynik_arabska;
      }
```

Zadeklarowałem zmienną wynik_arabska która będzie wynikiem naszej konwersji równą na razie zero. Drugą zmienną która nam pomoże jest „startindex" który też domyślnie jest równy zero. Startindex jest to zmienna typu Integer od której będziemy zaczynali czytać zadaną liczbę rzymską zapisaną w formacie String. Tab_rzymskie.length w naszym przypadku jest równe 13 bo ilość komórek tablicy to 13. A więc zaczynamy w pętli 'for' jest wykonywana inkrementacja zmiennej 'i'. Pętla „for" będzie wykonywana do

momentu aż 'i' będzie równe 12. Rozpisze jeszcze indeksy znakowe naszej zadanej liczby zapisanej w formacie String „MMCDXV" : M=0, M=1, C=2, D=3, X=4, V=5.

1. W pętli for (i=0; 0<13 (**prawda**); i++) przechodzimy do pętli 'while'. Rzymska.startsWith to metoda która sprawdza czy zadana przez nas liczba zapisana w systemie rzymskim zaczyna się od danej zawartości tablicy tab_rzymska[0] i miejsce od którego zaczyna poszukiwania określone jest przez „startindex" aktualnie równy zero czyli zaczyna od pierwszego znaku ze zmiennej String „MMCDXV" czyli „M", startsWith wyrzuca typ boolean czyli „**true**" albo „**false**". Rzymska zaczyna się od 'M' i tab_rzymskie[0]= 'M' więc while(**true**), wchodzimy do środka pętli while. Do „wynik_arabska" dodajemy wartość kryjącą się pod komórką zerową od „tab_arabskie" czyli 1000. Teraz do „startindex" dodajemy ilość znaków kryjących się pod komórką aktualną czyli zerową. Działamy w ten sposób aby przesunąć początek nowego czytania o tyle znaków ile odczytaliśmy (tyle znaków ile jest w komórce tab_rzymskie[od aktualnego i] wtedy i tylko wtedy dodamy do „startindex" ilość znaków jeśli znaki z komórki pasują do znaków odczytanych). Czyli w gruncie rzeczy ta operacja mówi nam czy odczytaliśmy jeden znak czy więcej. Jeśli odczytaliśmy jeden znak(i jest on zgodny z zawartością komórki tablicy) to przesuwamy się o jeden znak do przodu (żeby nie odczytywać znów tego samego znaku tylko stojący za nim (następny) lub w przypadku gdy w komórce tablicy jest więcej a widzimy że maksymalnie są dwa znaki to przesuniemy czytanie o dwa znaki które już odczytaliśmy. Jeżeli odczytaliśmy dwa znaki (np. CM=900) to do aktualnego „startindex" dodajemy 2 czyli tyle ile odczytaliśmy znaków. Nadal jesteśmy w pętli „while" ponieważ Rzymska czyli „MMCDXV" zaczyna się od znaku z indeksem startindex=1 również od M [„i" się nie zmieniło nadal jest 0] więc do „wynik_arabska" dodajemy wartość z komórki 0 tablicy arabskie czyli 1000. Aktualny „wynik_arabska" to 2000. Stratindex zwiększamy o 1 jest równy 2. Sprawdzamy czy prawdą jest że kolejnym odczytanym znakiem będzie 'M'. Nie jest to prawdą ponieważ kolejny znak z naszej liczby Rzymskiej jest 'C'.

2. W pętli for (i=1; 1<13 (**prawda**); i++) przechodzimy do pętli „while".
while (rzymska.startsWith(tab_rzymskie[1],startindex=2)) czyli czy CM będzie pasowała do znaków zaczynających się od indeksu 2 ze zmiennej String „Rzymska". „CM" nie jest równe „CD", fałsz. Przechodzimy dalej.

3. W pętli for (i=2; 2<13 (**prawda**); i++) przechodzimy do pętli „while".
while (rzymska.startsWith(tab_rzymskie[2],startindex=2)) czyli czy D jest kolejnym znakiem odczytanym ze zmiennej String „Rzymska" zaczynając od indeksu 2. 'D' nie jest równe 'C', **fałsz**. Przechodzimy dalej.

4. W pętli for (i=3; 3<13 (**prawda**); i++) przechodzimy do pętli „while".
while (rzymska.startsWith(tab_rzymskie[3],startindex=2)) czyli czy „CD" są kolejnymi znakami odczytanymi ze zmiennej String „Rzymska" zaczynając od indeksu 2. „CD" jest równe „CD", **prawda**. Wchodzimy do pętli while. Do „wynik_arabska" dodajemy wartość komórki tab_arabskie[i=3] czyli 400. Wynik_arabska to aktualnie 2400. Odczytaliśmy dwa znaki więc będziemy czytali naszą rzymską od znaku o indeksie o 2 większym niż aktualnie, więc startindex+=2 → startindex=4. Sprawdzamy czy kolejne znaki zaczynając od 4 ze zmiennej String „Rzymska" zawierają „CD"? **Fałsz** nie zawierają bo „CD" nie jest tym samym co „XV". Przechodzimy dalej.

5. W pętli for (i=4; 4<13 (**prawda**); i++) przechodzimy do pętli „while".
while (rzymska.startsWith(tab_rzymskie[4],startindex=4)) czyli czy 'C' jest kolejnym znakiem odczytanym ze zmiennej String „Rzymska" zaczynając od indeksu 4. 'C' nie jest równe 'X', **fałsz**. Przechodzimy dalej.

6. W pętli for (i=5; 5<13 (**prawda**); i++) przechodzimy do pętli „while".
while (rzymska.startsWith(tab_rzymskie[5],startindex=4)) czyli czy „XC" są kolejnymi znakami odczytanymi ze zmiennej String „Rzymska" zaczynając od indeksu 4. „XC" nie jest równe „XV", **fałsz**. Dalej.

7. W pętli for (i=6; 6<13 (**prawda**); i++) przechodzimy do pętli „while".
while (rzymska.startsWith(tab_rzymskie[6],startindex=4)) czyli czy 'L' jest kolejnym znakiem odczytanym ze zmiennej String „Rzymska" zaczynając od indeksu 4. 'L' nie jest równe 'X', **fałsz**. Dalej.

8. W pętli for (i=7; 7<13 (**prawda**); i++) przechodzimy do pętli „while".
while (rzymska.startsWith(tab_rzymskie[7],startindex=4)) czyli czy „XL" są kolejnymi znakami odczytanymi ze String'a rzymska zaczynając od indeksu 4. „XL" nie jest równe „XV" , **fałsz**. Dalej.

9. W pętli for (i=8; 8<13 (**prawda**); i++) przechodzimy do while.
while (rzymska.startsWith(tab_rzymskie[8],startindex=4)) czyli czy 'X' jest kolejnym znakiem odczytanym ze zmiennej String „Rzymska" zaczynając od indeksu 4.
'X' jest równe 'X' **prawda**. Wchodzimy do pętli while. Do wynik_arabska dodajemy wartość komórki tab_arabskie[i=8] czyli 10. Wynik_arabska to aktualnie 2410. Odczytaliśmy jeden znak więc będziemy czytali naszą rzymską od znaku o indeksie o 1 większym niż aktualnie, więc startindex+=1 → startindex=5. Sprawdzamy czy kolejny znak zaczynając od 5 ze zmiennej String „Rzymska" zawiera 'X'? **Fałsz** nie zawiera bo 'X' nie jest tym samym co 'I'.

10. W pętli for (i=9; 9<13 (**prawda**); i++) przechodzimy do pętli „while".
while (rzymska.startsWith(tab_rzymskie[9],startindex=5)) czyli czy „IX" są kolejnymi znakami odczytanymi ze zmiennej String „Rzymska" zaczynając od indeksu 5. „IX" nie jest równe 'V', **fałsz**. Dalej.

11. W pętli for (i=10; 10<13 (prawda); i++) przechodzimy do pętli „while".
while (rzymska.startsWith(tab_rzymskie[10],startindex=5)) czyli czy 'V' jest kolejnym znakiem odczytanym ze zmiennej String „Rzymska" zaczynając od indeksu 5. 'V' jest równe 'V' prawda. Wchodzimy do pętli while. Do wynik_arabska dodajemy wartość komórki tab_arabskie[i=10] czyli 5. Wynik_arabska to aktualnie 2415. Odczytaliśmy jeden znak więc będziemy czytali naszą „Rzymską" od znaku o indeksie o 1 większym niż aktualnie] więc startindex+=1→ startindex=6. Sprawdzamy czy kolejny znak zaczynając od 6 ze zmiennej typu String „Rzymska" zawiera 'V'? **Fałsz** nie zawiera bo nic nie jest tym samym co 'V'. Dalej.

12. Wnętrze pętli for wykona się jeszcze dwukrotnie ale nic nie zmieni w naszym wyniku ponieważ skończyły nam się już znaki w String „Rzymska". Pętla while będzie wyrzucała false i jej wnętrze nie będzie wykonywane także po konwersji wynik_arabska jest równa **2415**.

Opis programu i jego funkcji:

Program składa się z czterech klas:

- **Head** – Główna klasa z metodą public static void main(String[] args). Zarządza całym projektem.
- **Laborer** – Klasa której metody używane są do podstawowych operacji programu (klasa robocza).
- **Files** – Klasa stworzona do operacji na plikach (zapisu, tworzenia, otwierania, czytania plików)
- **Display** – Klasa stworzona do wyświetlania informacji oraz do pobierania od użytkownika danych.

Menu:

Użytkownik po włączeniu programu otrzymuje informacje wypisaną w konsoli z menu w której ma do wyboru kilka opcji. Wyboru między opcjami dokonuje poprzez wpisanie numeru opcji i zatwierdzeniu Enterem. Jeśli wpisze inną wartość niż przewidziano program zostaje zamknięty.

1) *Zamień liczbę Arabską lub Rzymską.*

Użytkownik wybierając opcje pierwszą otrzyma informacje ,że może wpisać w konsole zarówno liczbę zapisaną w systemie rzymskim jak i w systemie arabskim. Po wpisaniu liczby i jeśli liczba spełnia wymagania (np. zakres arabskiej to od 1 do 3999 ponieważ w systemie rzymskim nie możemy użyć więcej niż 3 razy znaków M, C, X, I czyli znaków odpowiadających arabskim 1000, 100, 10, 1) to zostanie wypisana liczba przekonwertowana z adnotacją w jakim systemie jest zapisana.

2) *Dokonaj konwersji kilku liczb i zapisz wyniki do pliku.*

Użytkownik w tym przypadku może wpisać w konsole kilka liczb dowolnie arabska/ rzymska i zapisać je do odpowiedniego pliku. Po wybraniu opcji 2 z menu zostanie wyświetlona informacja o tym jak wpisać ścieżkę do pliku. Użytkownik może wpisać albo po prostu samą nazwę pliku z rozszerzeniem np. dane.txt (tylko plik zostanie zapisany wtedy wprost na dysku C:\\) albo ścieżkę do pliku np. Documents and Settings\\dane.txt (również i w tym przypadku użytkownik nie musi wpisywać odpowiedniego dysku ponieważ domyślnie jest ustawiony dysk C:\\). Po wpisaniu odpowiedniej ścieżki do pliku użytkownik otrzyma informacje o tym jak wpisywać liczby czyli automatycznie będzie wznawiana opcja wpisu ,aż do momentu w którym użytkownik zdecyduje się na zatrzymanie poprzez wpis do konsoli „exit”. Po tym otrzyma informacje gdzie plik się zapisał i ,że zapis się powiódł lub gdy wystąpił błąd również otrzyma taką informacje.

3) Odczytaj liczby z pliku, dokonaj konwersji i wypisz je na konsole.

Opcja trzecia umożliwia odczyt z pliku zapisanego na dysku C:\\ w odpowiednim formacie i wypisanie liczb po konwersji na konsolę. Liczby powinny zostać zapisane w pliku w oddzielnych linijkach np:
xxx
1235
795
Liczby zostaną odczytane przekonwertowane przez program na odpowiednio rzymskie lub arabskie. Możemy wpisać w plik dowolne liczby bez różnicy czy to rzymska czy arabska program zamieni nam na przeciwną. W tym przypadku podobnie jak w punkcie drugim użytkownik może wpisać albo samą nazwę + rozszerzenie np. dane.txt lub ścieżkę do pliku należy pamiętać ,że używamy dwóch backslash'y do oddzielania kolejnych folderów lub plików. Różnicą między punktem drugim jest tutaj fakt ,że nie tworzymy nowego pliku przez program a po prostu go odczytujemy także użytkownik powinien wcześniej przygotować plik z danymi do odczytu. Po poprawnym wpisani program zacznie prace wypisze nam na konsole aktualnie czytane dane i dane po konwersji.

4) Odczytaj liczby z pliku, dokonaj konwersji i zapisz do pliku wyniki.

Użytkownik w tym przypadku może dokonać konwersji z pliku do pliku czyli podaje ścieżkę lub nazwę pliku podobnie jak miało to miejsce w wyżej wymienionych podpunktach oraz otrzymuje informacje o tym ,że plik z wynikami zostanie zapisany w tym samym folderze/dysku co plik odczytany z tym ,że jego nazwa będzie zawierała „-wyniki" np. plik który chcemy odczytać nazywa się dane.txt zapisane dane utworzą się w pliku o nazwie dane-wyniki.txt. Po podaniu poprawnej ścieżki do pliku dostaniemy informacje o aktualnie wczytanej liczbie co umożliwia nam wczesną weryfikacje czy cały proces przebiega poprawnie. Na koniec otrzymujemy informacje o sukcesie lub porażce zapisu pliku, lecz jeśli odpowiednio zapisaliśmy plik do odczytu (w kolejnych liniiach kolejne liczby) nie powinno być problemów z zapisem.

5) Wpisz 'exit' lub '5' jeśli chcesz zakończyć działanie programu.

Jeśli chcemy zakończyć działanie programu wpisujemy po prostu exit lub 5 (jeśli wpiszemy inną wartość niż cyfry z menu program i tak zostanie zamknięty).

Inne funkcje programu i opis ciekawych metod :

Program oferuje także możliwość wyboru między kontynuacją a zakończeniem działania po wyborze z menu opcji i zakończeniu wybranego procesu.
Przykład:
Wybraliśmy pierwszą opcje czyli zamianę liczby rzymskiej lub arabskiej wpisaliśmy xxx po konwersji 30. Otrzymujemy „okienko" z wyborem czy dalej chcemy kontynuować program (przeniesie nas do wyboru opcji z menu) lub zakończeniu działania programu.

Program używa dwóch kluczowych metod do konwersji liczb arabskich i rzymskich są to:

- **private int** doArabskiej(String rzymska)
- **private** String doRzymskiej (**int** arabska)

Jak program zamienia liczby zostało opisane w przykładach zamiany.

Ale ciekawe jest to jak program dowiaduje się o tym czy aktualnie wpisana lub czytana liczba jest arabską czy rzymską. Za ten proces odpowiada metoda:

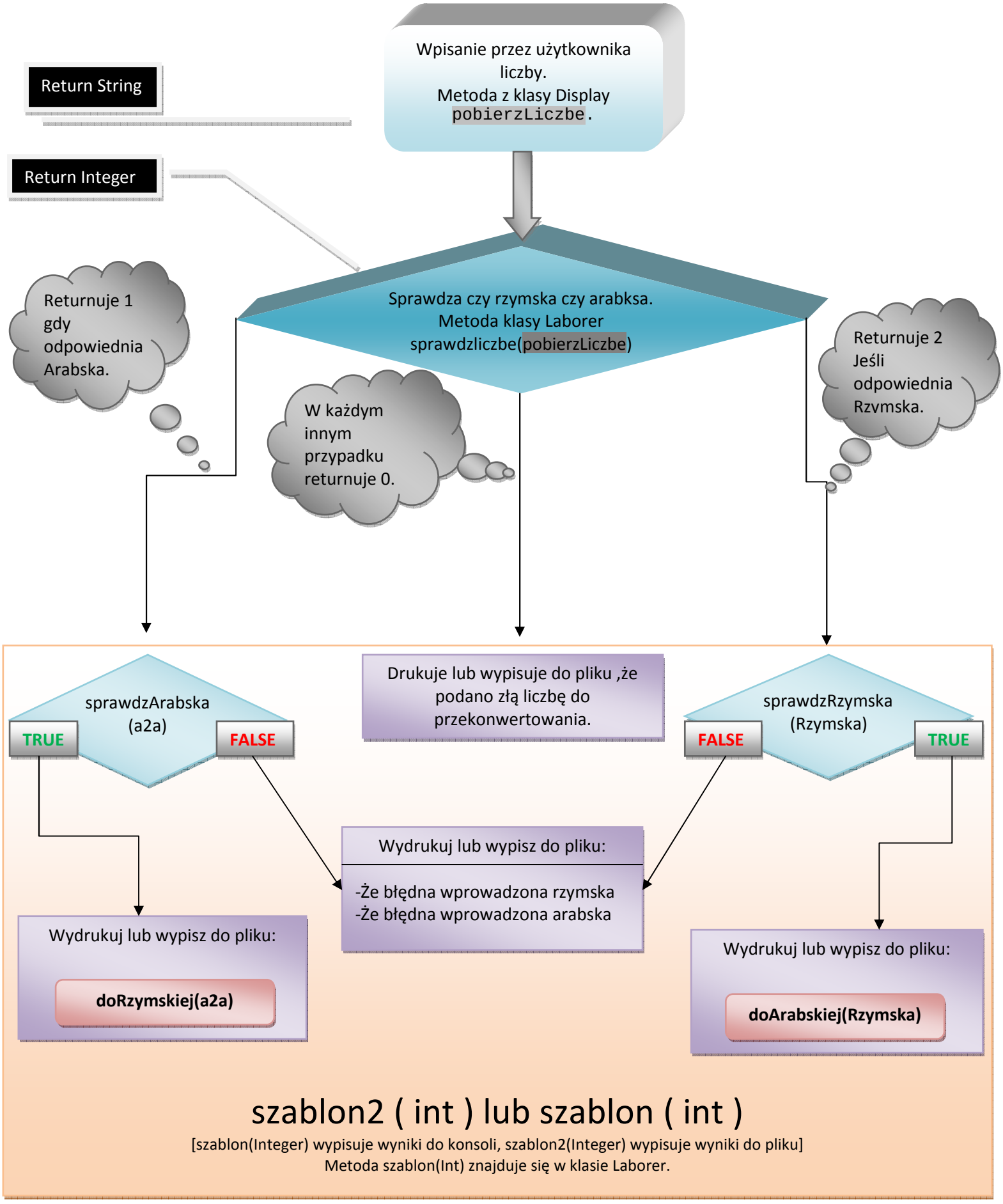
```
public int sprawdzliczbe(String L)       {

Arabska=""; Rzymska=""; a2a=0;
for (int i=0; i<L.length(); i++)
{
if ( 47<L.codePointAt(i)&& L.codePointAt(i)<58){
      Arabska+=L.charAt(i); Arabska=Arabska.trim();  }
if (L.charAt(i)=='X' || L.charAt(i)=='C'  || L.charAt(i)=='D' ||
L.charAt(i)=='M'|| L.charAt(i)=='V'|| L.charAt(i)=='L' || L.charAt(i)=='I')
{ Rzymska+=L.charAt(i); Rzymska=Rzymska.trim(); }
}

if (L.length()==Arabska.length() && L.length()!=0)
{
      a2a=Integer.parseInt(Arabska);
      return 1;
}
if (L.length()==Rzymska.length() && L.length()!=0)
{
      return 2;
}
else return 0;
      }
```

Zadeklarowałem dwie zmienne Arabską i Rzymską trzecią zajmiemy się później.
Arabska będzie odpowiadać za liczby arabskie a Rzymska odpowiednio za liczby zapisane w systemie rzymskim. W pętli for jeśli aktualnie czytany znak jest cyfrą to dodajemy ten znak do Arabskiej (na razie o typie String) wykorzystałem tutaj kodowanie UTF w którym znaki zapisane są w postaci kodu. Cyfry jakie nas interesują zawierają się między kodem (47 a 58) [nie musze wypisywać po kolei cyfr jakie są tylko określam przedział w jakim się znajdują]. Z drugiej strony jeśli aktualnie czytany znak jest literą która może być użyta w systemie rzymskim (I, V, X, L, C, D, M) to dodajemy ją do zmiennej typu String Rzymska[tutaj już musimy wypisać jakie znaki mogą być użyte]. Po wykonaniu całej pętli sprawdzamy czy długość Arabskiej lub Rzymskiej odpowiada długości danej wprowadzonej przez użytkownika lub przeczytanej z pliku, wtedy i tylko wtedy gdy Arabska jest złożona z samych cyfr i długość wpisanego lub czytanego tekstu jest równa długości wprowadzonej danej **returnujemy 1** ale również zamieniamy cyfry zapisane w typie String na liczbę w formacie Int. Czyli wracamy do zmiennej a2a która będzie od teraz zawierała cyfry zamienione z Arabskiej(String) na Integer czyli całkowite.
W przypadku gdy długość zmiennej Rzymska jest równa długości wprowadzonej danej **returnujemy 2.** Nie ma możliwości żeby arabska i rzymska były równe sobie i równe wprowadzonej danej. Jedyną taką możliwością byłoby wprowadzenie niczego czyli zatwierdzenie enterem bez wprowadzania danych lub pusta linia czytanych danych ale na taki wypadek ubezpieczyliśmy się przez komendę L.length()!=0. Czyli jeśli długość Arabskiej lub Rzymskiej równa się długości czytanej lub wprowadzonej danej a ponadto nie jest ona bez znakowa czyli o długości zero dopiero wtedy dostajemy return o wartości 1 lub 2. W każdym innym przypadku **return 0**.

Przedstawię teraz cykl przechodzenia liczby od wpisania przez użytkownika w konsole lub czytania z pliku aż do 'wypirintowania' na ekran lub zapisu do pliku.



szablon2 (int) lub szablon (int)

[szablon(Integer) wypisuje wyniki do konsoli, szablon2(Integer) wypisuje wyniki do pliku]
Metoda szablon(Int) znajduje się w klasie Laborer.