

## Laboratorium 12

### Zadanie obowiązkowe

#### Konto oszczędnościowe

1. Napisz program obsługi kont bankowych. W tym celu, utwórz interfejs `IKonto`, posiadający metody do wpłat (metoda `Wplata`, z argumentem `kwota`) i wypłat (metoda `Wypłata`, z argumentem `kwota`) oraz odczytu (`Bilans`) stanu konta.  
Implementująca powyższy interfejs klasa `KontoOszczednosciowe` powinna posiadać prywatne pole `bilans` oraz metody wynikające z implementacji interfejsu. Metoda `Wypłata` powinna zakazywać wypłat kwot wyższych od stanu konta. Rozwiąż problem za pomocą wyjątku (standardowego, bądź niestandardowego).  
Dodatkowo, zaimplementuj metodę `PolaczDwa`, umożliwiającą połączenie dwóch wskazanych kont bankowych (obejmujące likwidację tego z nich, na którym znajduje się mniejsza kwota pieniędzy; na czym będzie polegać likwidacja konta?).  
Następnie, utwórz dla klasy `KontoOszczednosciowe` jej podklasę `MaleKonto`, umożliwiającą realizację wypłat tylko do wskazanej wysokości. W celu realizacji zadania, przesłoń metodę `Wypłata`.  
Zademonstruj działanie wszystkich wymienionych metod w utworzonej przez siebie klasie `Bank`.

### Zadanie obowiązkowe

#### Dwa ciągi – wyjątki

2. Na standardowym wejściu, w kolejnych dwóch wierszach, zapisane są dwa ciągi znaków, odpowiednio ciąg `a` i ciąg `b`, takie że  $\text{długość}(a) < \text{długość}(b)$ . Skonstruuj tablicę liczb całkowitych, o wymiarze  $10 * 2$ . Zapisz w kolejnych wierszach tej tablicy pozycje wszystkich wystąpień ciągu `a` wewnątrz ciągu `b`, w postaci:  
[indeks\_pierwszego\_znaku\_ciagu\_a\_w\_ciagu\_b, indeks\_ostatniego\_znaku\_ciagu\_a\_w\_ciagu\_b].  
Zastosuj numerację znaków zaczynającą się od 1.  
Zaproponuj własne wyjątki do opisu potencjalnych błędów w programie. Dokonaj ich zgłoszenia i obsługi we wszystkich niezbędnych miejscach programu.

#### Przykład1

##### Dane wejściowe:

aba  
ababakjabsabagab

##### Dane wyjściowe:

1	3
3	5
11	13

## Przykład2

### Dane wejściowe:

anbbngdfaa

bdbnf

### Dane wyjściowe:

Operacja niewykonalna -  $\text{długość}(a) > \text{długość}(b)$

## Zadanie obowiązkowe

### Stos – obsługa wyjątków

3. Zdefiniuj w języku Java podstawowe operacje do działań na stosie (**top**, **pop**, **push**).  
Wykorzystaj je w programie wypełniającym stos liczb całkowitych liczbami naturalnymi wprowadzonymi ze standardowego wejścia. Opróżnij zbudowany stos, wyprowadzając na wyjście wszystkie i tylko te spośród zdejmowanych liczb, które są podzielne przez 2.  
W programie zdefiniuj obsługę sytuacji wyjątkowych, powstających w trakcie jego działania. Skorzystaj z wyjątków własnych (niestandardowych). Obsłuż je w najprostszy możliwy sposób: zaniechaj obliczeń i dokonaj bezpiecznego zamknięcia programu. Pamiętaj o konieczności użycia klauzuli **throws**.

## Zadanie obowiązkowe

### Stos – implementacja interfejsu i obsługa wyjątków

4. Zdefiniuj interfejs **Stos** z podstawowymi funkcjami do obsługi stosu (**top()**, **pop()**, **push(int el)**) oraz stałą **r**, oznaczającą pojemność stosu.  
Zaimplementuj ten interfejs w klasach: **Stos\_Tablica** (stos przechowywany w klasycznej tablicy, o pojemności zadanej w interfejsie) oraz **Stos\_Lista** (stos przechowywany i obsługiwany w klasie wbudowanej **ArrayList**).  
W obu klasach użyj dokończeniowej metody **finalize**, która specjalnym komunikatem poinformuje o likwidacji obiektu, a wcześniej wyprowadzi zawartość stosu w obiekcie.  
Do zgłoszenia wyjątków powstających w wyniku działań na stosie użyj wyjątków własnych **stos\_pusty** i **stos\_pelny**. Rozdziel miejsca zgłoszenia i obsługi tych wyjątków. Pamiętaj o konieczności użycia klauzuli **throws**. Oprócz właściwych sekcji obsługi wyjątków, zdefiniuj sekcję **finally**, która poinformuje specjalnym komunikatem o wyjściu z obszaru potencjalnych zgłoszeń wyjątków.  
W każdej z klas umieść pole statyczne **ile\_obiektow**, które będzie przechowywać bieżącą liczbę obiektów danego typu. Umieść we właściwych miejscach instrukcje odpowiedzialne za modyfikację tego pola.  
Interfejs **Stos** i klasy **Stos\_Tablica** i **Stos\_Lista** umieść w odrębnym pakiecie **Stosy**.  
Utwórz jeden obiekt typu **Stos\_Tablica** (**Ob1**) i dwa obiekty typu **Stos\_lista** (**Ob2**, **Ob3**). Wykonaj na tych obiektach działania (**PU**, **PO**, **TO**) wskazane w pliku wejściowym **plik1.txt**. Na koniec, wyprowadź wartości przechowywane w polach statycznych **ile\_obiektow** obu klas. Wszystkie informacje wyprowadź do pliku wyjściowego **plik2.txt**.

### **Przykład**

**Dane wejściowe (dla  $r = 5$ ):**

```
PU OB1 2 //push do stosu-tablicy
PU OB1 4
PU OB1 3
PU OB1 7
PU OB1 8
PU OB1 6
PO OB2 //pop z pierwszego stosu-listy
TO OB3 //top z drugiego stosu-listy
PU OB3 9 //push do drugiego stosu-listy
PU OB3 1
PU OB3 4
PU OB3 3
PU OB3 7
PO OB3 //pop z drugiego stosu-listy
TO OB3
```

**Dane wyjściowe:**

```
Stos OB1 pełny!
Stos OB2 pusty!
Stos OB3 pusty!
Szczytowy element OB3: 3
```

### **Zadanie obowiązkowe**

5. PW/MC lab. 12 zad. 2

### **Zadanie dla chętnych**

6. PW/MC lab. 12 zad. 3 i zad. 5