Beata Jankowska
Instytut Informatyki PP

Laboratorium 12

Szanowni Państwo,

bardzo przepraszam za zwłokę w publikacji materiałów pomocniczych i zadań z laboratorium nr 12

1. JPO_lab_12a _zad1

Konto oszczędnościowe

2. Napisz program obsługi kont bankowych. W tym celu, utwórz interfejs `IKonto`, posiadający metody do wpłat (metoda `Wplata`, z argumentem `kwota`) i wypłat (metoda `Wyplata`, 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 `Wyplata` 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; jak zamierzasz zrealizować likwidację konta?)
   Następnie, utwórz dla klasy `KontoOszczednosciowe` jej podklasę `MaleKonto`, umożliwiające realizację wypłat tylko do wskazanej wysokości. W celu realizacji zadania, przesłoń metodę `Wyplata`.
   Zademonstruj działanie wszystkich wymienionych metod w utworzonej przez siebie klasie `Bank`.

Dwa ciągi – wyjątki

3. Na standardowym wejściu, w kolejnych dwóch wierszach, zapisane są dwa ciągi znaków, odpowiednio ciąg `a` i ciąg `b`, takie że długość(`a`) < 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_ciągu_b`,` indeks_ostatniego_znaku_ciagu_a_w_ciągu_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
```

Beata Jankowska
Instytut Informatyki PP

Dane wyjściowe:

```
1       3
3       5
11      13
```

Przykład2

Dane wejściowe:

```
anbbngdfaa
bbdnf
```

Dane wyjściowe:

```
Operacja niewykonalna - długosc(a) > długosc(b)
```

Stos – obsługa wyjątków

4. 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`.

Stos – implementacja interfejsu i obsługa wyjątków

5. 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`.

Beata Jankowska
Instytut Informatyki PP

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` (dane w pliku plik1.txt) :

```
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 (dane w pliku plik2.txt):

```
Stos OB1 pelny!
Stos OB2 pusty!
Stos OB3 pusty!
Szczytowy element OB3: 3
Obiekty Stos_tablica: 1
Obiekty Stos_Lista: 2
```

Zadania dodatkowe

6. JPO_lab_12b_zad1
7. JPO_lab_12b _zad3
8. JPO_lab_12b_zad5