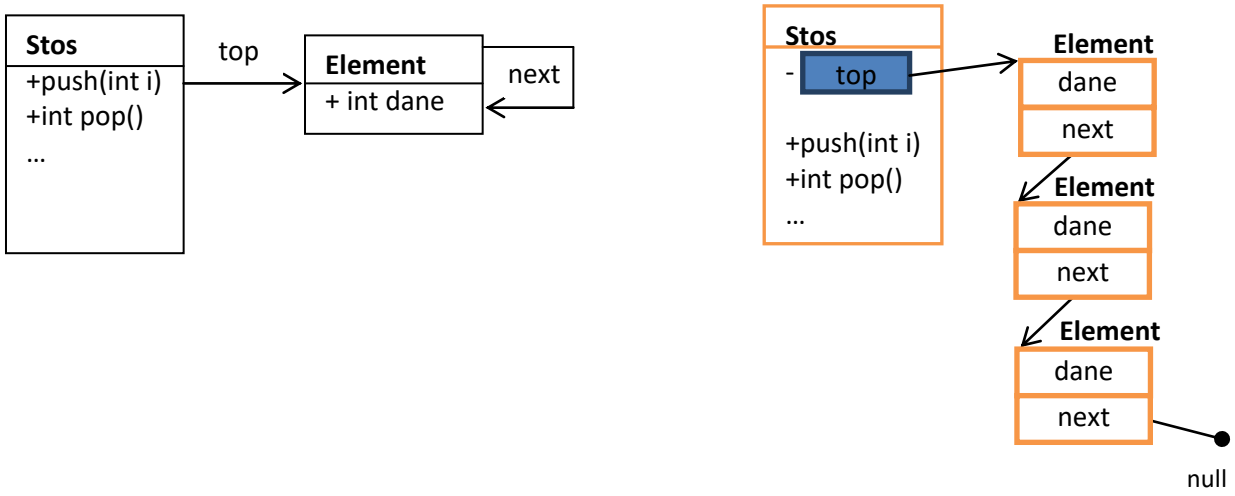
Algorytmy i złożoność obliczeniowa

Laboratorium 3: Stosy. Proste algorytmy wykorzystujące stosy.

1. **Koncepcja stosu.**
   Stos jest jednym z rodzajów struktur danych. Kolejne elementy (dane) są układane na wierzchołku stosu. Usunięty może być tylko element z wierzchołka stosu. Dobrym przykładem obrazującym działanie stosu jest stos książek na biurku. Możemy dołożyć kolejną książkę kładąc ją na wierzchołku, po czym to właśnie nowo dołożona książka staje się wierzchołkiem stosu. Jeśli chcemy z takiego stosu wziąć książkę, to możemy wziąć tylko książkę, która leży na wierzchu. Jeśli chcemy sięgnąć po inną książkę, musimy kolejno zdejmować książki z wierzchołka.

2. **Operacje na stosie.**
   push – dołożenie elementu na stos
   pop – ściągnięcie elementu ze stosu (i zwrócenie jego wartości)
   isEmpty – sprawdzenie, czy stos jest pusty
   Często implementacje stosu dostarczają dodatkowych operacji, jak np.
   peek – sprawdzenie wartości elementu na wierzchołku stosu bez jego zdejmowania

3. **Diagram klas oraz schemat odwołań referencji poszczególnych elementów stosu.**



4. **Interfejs IStos, klasa Element – stos przechowujący liczby całkowite.**

```
public interface IStos {
    public void push(int i);  // dodaje i na stos
    public int pop();         // zdejmuje element, zwraca wartość z wierzchołka
    public int peek();        // zwraca wartość z wierzchołka stosu
    public boolean isEmpty(); // sprawdza, czy stos jest pusty
    public void print();      // wypisuje na ekran zawartość stosu
    public void clear();      // usuwa wszystkie elementy ze stosu
}

public class Element{           // element stosu
    public int dane;            // wyjątkowo, dla uproszczenia, pola publiczne
    public Element next;        // referencja do następnego elementu stosu
}
```

5. **Zadania.**
   1. Utwórz klasę **Stos** implementującą interfejs **IStos** jako strukturę wiązaną w oparciu o klasę **Element**. Napisz program testujący zaimplementowane metody (np. włożenie kilku elementów na stos, wypisanie zawartości stosu, zdjęcie elementu z wierzchołka, wypisanie zawartości stosu itp.).
      Wskazówki:
      - w klasie Stos utwórz prywatne pole typu Element (np. o nazwie top);
      - sprawdź, czy stos jest niepusty zanim zdejmiesz element ze stosu;
      - wypisanie stosu (print) nie może zmieniać zawartości stosu

      [4p]

      Komentarz: Na rysunku zaprezentowano sposób wykorzystania instancji klas Stos i Element. Zwróć uwagę, że obiekt typu Stos przechowuje tylko jedną referencję do obiektu typu Element (wierzchołek stosu - *top*), a nie wszystkie elementy znajdujące się na stosie. W szczególności gdy stos jest pusty prawdziwy jest warunek `top == null`. Elementy stosu (jak wszystkie obiekty Java) są alokowane na stosie po wywołaniu konstruktora (np. `Element nowy = new Element()`). Każdy obiekt typu Element w polu `dane` przechowuje liczbę całkowitą, a w polu `next` referencję do następnego elementu (tego, który jest tuż pod spodem), lub wartość null jeśli dany element jest na samym dole stosu (jeśli jest ostatni).

   2. Napisać program wykorzystujący stos, który sprawdza czy w podanym wyrażeniu nawiasy są prawidłowo zagnieżdżone. Np. wyrażenie "(( ) (( )))" jest prawidłowe, a ")(" oraz "(()" są nieprawidłowe. **[2p]**
      Wskazówka: Umieszczaj element na stosie w momencie napotkania '(', a zdejmuj element kiedy napotkasz ')'
   3. Napisać program wykorzystujący stos, który sprawdza czy podane słowo jest palindromem. **[2p]**
      Wskazówka: Użyj stosu do odwrócenia słowa.
   4. Napisać program czytający wyrażenia arytmetyczne w notacji postfiksowej (Odwrotna Notacja Polska) i obliczający ich wartość z wykorzystaniem stosu. **[2p]**
      Wskazówka: Po napotkaniu liczby umieść ją na stosie, po napotkaniu operatora zdejmij dwie liczby ze stosu, wykonaj działanie i umieść wynik na stosie.
      Przykładowe wyrażenie w ONP: 5 2 + 7 * odpowiada (5+2)*7. Wynik 49.

Praca domowa:

1. Napisać program czytający z pliku wyrażenia arytmetyczne zbudowanie z wykorzystaniem operatorów + - * / () i zamieniający je na wyrażenia w notacji postfiksowej z wykorzystaniem stosu.
2. Zaimplementować klasę Stos jako klasę generyczną, aby można było przechowywać na stosie obiekty dowolnego typu.
3. Rozwiązać zadania 2-4 wykorzystując klasę java.util.Stack lub implementację interfejsu java.util.Deque (np. LinkedList).

Na następnych zajęciach:

Listy. Kolejki. Operacje na listach i kolejkach.