Téma 15: ADT - Zásobník

Fronta i zásobník jsou lineární spojové seznamy. Oba dva mají strukturu tvořenou **elementy**, které na sebe ukazují, proto musíme nejdříve vytvořit element, který je pro oba lineární seznamy stejný.

```
public class Element {
     private int value;
     private Element element;
     private Element(int value) {
          this.value = value;
     }
     public Element getElement() {
          return element;
     }
     public void setElement(Element element) {
          this.element = element;
     }
     public int getValue() {
          return value;
     }
}
```

Element udržuje hodnotu daného elementu int value, který se nastaví při vytvoření elementu. Proměnná typu Element je ukazatel na další Element, pro práci s elementy zásobníku a fronty se používají gettery a settery, pomocí getterů získávám hodnotu elementu a pomocí setterů nastavuji hodnotu elementů.

Zásobník je lineární spojový seznam typu LIFO – Last In First Out, což znamená, že prvek který byl do fronty přidán jako poslední, je z fronty odebrán jako první. Je to stejné jako zásobník u pistole.

Pokud chci správně implementovat zásobník, musím si uchovávat ukazatel na vrchol zásobníku, třeba v proměnné Element top. Ideální je ukazovat na vrchol, jelikož na něj přidávám a z něj také odebírám.

Pokud chci do zásobníku přidat nový element, jednoduše metodě (add) předám hodnotu nového elementu. Poté v metodě vytvořím nový element, kterému nastavím jeho ukazatel na aktuální top. Poté aktuálnímu topu vytvořím referenci na nově vytvořený prvek.

Odebírá se pomocí metody pop, prvně zkontroluji, zda zásobník není prázdný, tím že by top ukazoval na null. Poté si uložím do proměnné hodnotu z elementu, na který míří aktuální top.

Poté nastavím topu referenci na prvek, na který aktuální top ukazuje. Poté vrátím hodnotu předchozího topu.

```
public class Stack {
private Element top = null;
public void add(int value) {
     Element e = new Element(value);
     e.setElement(top);
     top = e;
}
public boolean isEmpty() {
     if(top == null) {
     return true;
```

```
      }
      return false;
}
public Integer remove() {
      if(isEmpty()) {
            return null;
      } else {
            int value = top.getValue();
            top = top.getElement();
            return value;
            }
      }
}
```