16. ADT – Fronta (princip, realizace, základní operace)

Architektura, kterou je možno realizovat pomocí lineárního spojového seznamu typu **FIFO** (First In First Out). První prvek, který se do fronty vloží se z ní i jako první odebere.

Nejprve se zvolí, odkud se budou odebírat prvky a kam se budou ukládat. Ideální stav je aby vršek fronty sloužil k odebírání prvků z fronty. Spodek fronty bude tedy sloužit k přidávání prvků do pole

Základní metody pro práci s frontou:

Konstuktor

Vytvoří prázdnou frontu (first, last ukazují na null)

isEmpty()

Test na prázdnost; pokud se top rovná null → fronta je prázdná

first()

Vrátí referenci na první prvek fronty.

last()

Vrátí referenci na poslední prvek fronty.

add()

Přidá další prvek na konec fronty.

Pokud je fronta prázdná, tak při přidání nového prvku je first právě ten nový prvek, last je nový prvek taktéž. Pokud fronta není prázdná, tak poslednímu prvku se nastaví (**setNext()**) přidávaný prvek a opět se nastaví last jako přidávaný prvek.

remove()

Odebere první prvek z fronty.

Pokud je fronta prázdná, tak nelze nic odebírat → vyhodí se výjimku. Poté se vytvoří nový dočasný Element, do kterého se uloží první prvek fronty. Do prvního prvku se uloží **first.getNext()**, Poté se provede kontrola, zda se první prvek nerovná null → jestli ano, tak se poslední prvek taky nastaví na null. Následně se vrátí dočasný element, který byl na začátku vytvořen.

Výhody

Nekonečný

Nevýhody

Složitější realizace

Použití

BFS

```
public class Queue {
     private Element first; private Element last;
     public Queue() {
        this.first = null;
        this.last = null;
     }
     public boolean isEmpty() {
        if((this.first == null) && (this.last == null)) {
            return true;
        } else {
            return false;
        }
     }
     public Element first() {
          if(this.isEmpty()) {
               throw new NullPointerException("prázdno");
          } else {
               return this.first;
          }
     }
     public Element last() {
          if(this.isEmpty()) {
               throw new NullPointerException("prázdno");
          } else {
               return this.last;
          }
     }
     public void add(Element e) {
          if(this.isEmpty()) {
               this.first = e;
          } else {
               this.last.setNext(e);
          }
          this.last = e;
    }
```

```
    public Element remove() {
        if(this.isEmpty()) {
            throw new NullPointerException("prázdno");
        } else {
            Element tmp = this.first;
            this.first = this.first.getNext();
            if(this.first == null) this.last = null;
            return tmp;
        }
    }
}
```