

Zadat úkol skupině č. 1

Zadat úkol skupině č. 2

## Kontext

---

Pole a spojový seznam jsou základní datové struktury, kterým musíme porozumět, stejně jako operacím s nimi a jejich časové složitosti. Pomocí pole a seznamu můžeme rovněž implementovat některé další struktury, například zásobník a frontu.

## Zadání

---

### 1. Pole

---

Dokončete implementaci třídy **BooleanArray** ze [třetího semináře](#). Důležité metody (`[]` a `[]=`) máte k dispozici, úkolem je doplnit obsah ostatních metod. Dejte pozor na dodržení rozhraní metod, správné návratové hodnoty a správné vyvolání výjimek. Korektnost svého kódu si můžete ověřit pomocí dodaného testu. (Pozor, test nemusí postihovat úplně všechny myslitelné situace. Jinými slovy sice platí, že pokud jím implementace neprojde, je chybná, ale nemusí nezbytně platit, že pokud jím projde, je stoprocentně správná.)

**Výstup:** soubor `boolean_array.rb`

## 2. Jednosměrný seznam

---

Vytvořte implementaci třídy **SinglyLinkedList**, která je obdobou třídy **LinkedList** ze [čtvrtého semináře](#) a která implementuje jednosměrný spojový seznam. V příloze naleznete kostru této třídy (především seznam metod, které je potřeba naimplementovat) a opět test, který vám pomůže ověřit si správnost svého kódu. Stejně jako v případě **BooleanArray** dejte pozor na správné návratové hodnoty a vyvolání výjimek. Výjimky vyvolejte ve velmi podobných situacích jako v případě **BooleanArray**, jaké přesné výjimky kdy vyvolat usudte sami podle **BooleanArray**.

**Upozornění:** *myslete na to, že implementace by měla být nejenom korektní, ale rovněž i efektivní. Některé metody může být vhodné v případě jednosměrného seznamu implementovat jinak než přímou analogií stejných metod na obousměrném seznamu.*

**Výstup:** soubor `singly_linked_list.rb`

## 3. Zásobník

---

Vytvořte implementaci tříd **SinglyListStack** a **BooleanArrayStack**. Obě budou reprezentovat [zásobník](#) (tedy budou implementovat metody `push` a `pop`), první z nich pro implementaci použije třídu **SinglyLinkedList**, druhá použije třídu **BooleanArray**.

V obou případech nezapomeňte na potřebu řešení podtečení zásobníku (zavolání operace `pop` na prázdném zásobníku), v takové situaci vyvolejte výjimku předpřipravené třídy **StackEmptyException**. V případě **BooleanArrayStack** je rovněž potřeba řešit přetečení zásobníku (zavolání operace `push` na plném zásobníku), v takové situaci vyvolejte výjimku třídy **StackFullException**.

K oběma třídám sami vytvořte vhodné unit testy ověřující korektnost implementace.

**Výstup:** oba zásobníky v souboru `stack.rb`, testy v souboru `stack_test.rb`

## 4. Fronta

---

Vytvořte implementaci tříd **SinglyListQueue** a **BooleanArrayQueue**. Obě budou reprezentovat [frontu](#) (tedy budou implementovat

metody `queue` a `dequeue`).

První z nich pro implementaci použije třídu **SinglyLinkedList**, zde opět není potřeba řešit případ plné fronty (vždy můžeme přidat nový prvek). Druhá použije třídu **BooleanArray**, ovšem pozor, implementace si potom musí vystačit s metodami `[]` a `[]=` za pomoci udržování dvou indexů označujících začátek a konec fronty způsobem popsáným na přednášce (jinými slovy implementace nesmí používat metody jako `<<` nebo `delete_at`). V případě přeplnění fronty **BooleanArrayQueue** dojde k výjimce třídy **QueueFullException**. Při obou implementacích vyvolejte výjimku třídy **QueueEmptyException** v případě operace `dequeue` na prázdné frontě.

K oběma třídám opět sami vytvořte vhodné unit testy ověřující korektnost implementace.

**Výstup:** obě fronty v souboru `queue.rb`, testy v souboru `queue_test.rb`

## Všeobecné poznámky

---

### Způsob odevzdání

- všech šest požadovaných souborů uložte do adresáře `prijmeni_jmeno_du1`
- adresář zabalte do souboru `prijmeni_jmeno_du1.zip` nebo `prijmeni_jmeno_du1.7z`
- v +4U artefaktu s odevzdáním vyplňte na začátek políčka **Název** řetězec **DU1**
- Nedodržení tohoto způsobu uložení bude mít za následek snížení počtu bodů, případně nemožnost úkol opravit.

Pro shrnutí, očekává se, že odevzdáte:

- Efektivní implementaci třídy **BooleanArray**, která rozhodně projde dodanými unit testy.
- Efektivní implementaci třídy **SinglyLinkedList**, která rozhodně projde dodanými unit testy.
- Efektivní implementaci tříd **SinglyListStack** a **BooleanArrayStack**.
- Vhodné unit testy ke třídám **SinglyListStack** a **BooleanArrayStack**.
- Efektivní implementaci tříd **SinglyListQueue** a **BooleanArrayQueue**.
- Vhodné unit testy ke třídám **SinglyListQueue** a **BooleanArrayQueue**.

## Řešení



---

Řešení posílejte tlačítkem níže do termínu, který máte uveden v úkolu nad tímto artefaktem. Jakékoliv později odevzdané řešení bude automaticky hodnoceno 0 body.

Odeslat řešení domácího úkolu

## Zdroje

---

-  ● ALG Datové struktury
-  ● ALG Bitové operace, hashování