3️⃣

Blatt 3

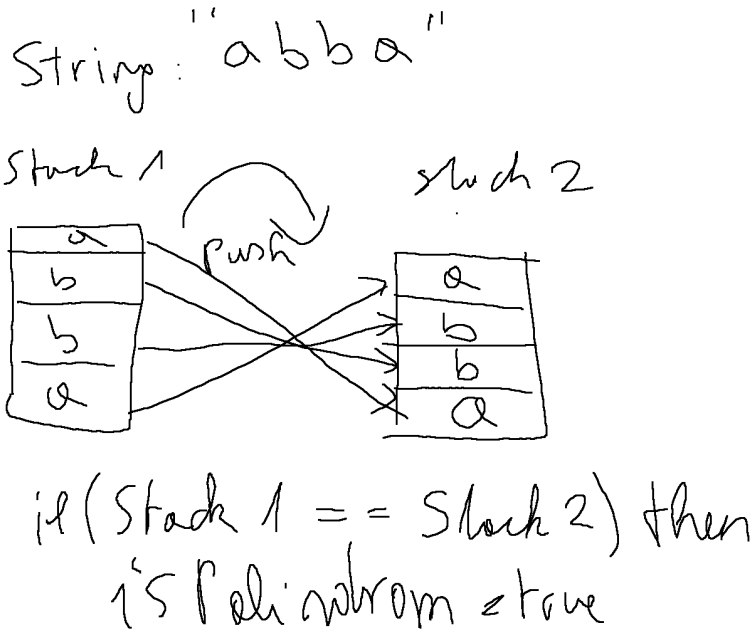
Aufgabe 1

1.

Wenn man zwei stack verwendet und diese dann so initialisiert:
Datei ist auch hochgeladen worden!

```
import java.util.Stack;

public class PalindromTest {
    public static void main(String[] args) {
        String inputtedString = "Able was I, ere I saw Elba";
        String inputtedString_c = inputtedString.toUpperCase();
        inputtedString_c = inputtedString_c.replaceAll("\\W","");
        Stack<Character> stack = new Stack<>();

        for (int i = 0; i < inputtedString_c.length(); i++) {
            stack.push(inputtedString_c.charAt(i));
        }

        String reversedString = "";

        while (!stack.isEmpty()){
            reversedString = reversedString+stack.pop();
        }
        if (inputtedString_c.equals(reversedString)){
            System.out.println("The inputted String is a palindrome");
        }
        else {
            System.out.println("The inputted String is not a palindrome");
        }
    }
}
```

Der String wird rückwärts auf einen anderen Stacks gepusht und dann anschleißend aufeinander verglichen.



2.

> Ticked boxes are Palindroms!

- [x] ~~Abba~~
- [x] ~~Rentner~~
- [ ] Angela
- [x] ~~Lagerregal~~
- [ ] Algorithmen und Datenstrukturen
- [x] ~~Saippuakivikauppias~~
- [x] ~~Die Liebe ist Sieger, stets rege ist sie bei Leid~~
- [ ] Ist das die Lösung?
- [x] ~~Geist ziert Leben, Mut hegt Siege, Beileid tragt belegbare Reue, Neid dient nie, nun eint Neid die neuerer, abgelebt gart die Liebe, geist geht, umnebelt reizt Sieg.~~

3. Implementierungen mit den Stack sind sehr effizient da sie nach dem LIFO-Prinzip gehen (”Last in first Out”). Nachteil dabei ist, dass man nichts in den Stack legen kann. Auch beim nehmen eines Elements kann nur das oberste Element runter gepusht werden. $\mathcal{O}(n) = n + n = 2n$

Vorteil dieser Methode ist, dass sehr schnell zu Beginn gleich überprüft wird ob der gegebene String ein Palindrom ist.
Nachteil dieser Methode ist, dass man bei einem langen String alles pushen muss, wenn der Fehler jetzt eher zu Begin des Strings liegt.

Aufgabe 2

1.

▼ linked List - **SLOWEST**

▼ Für jeden durchlauf muss eine neue “Node” erstellt werden. Neue Nodes müssen immer hinten drangehängt werden. Zudem muss man immer vom “Head” ausgehen um auf ein bestimmtes Element zuzugreifen

▼ cyclic Array - **2ND SLOWWEST**

▼ eine Double Ended Queue kann vorne und hinten neue Elemente dranhängen, aber der Hauptgrund ist die unbekannte Größe des Speichers. Wenn ein Array mit Größe 10, dann auf einmal 20 Elemente bekommt, dann muss dann in ein neues Array kopiert werden und noch dazu muss die Größe vom neuen Array angepsst werden.

▼ cyclic Array with preallocated memory - **FASTEST**

▼ Eine Double Ended Queue mit maximaler Größe ist deshalb schneller da eine fixe Größe bekannt ist und das Program muss nicht entscheiden was es machen soll, falls mehr Elemente hinzugefügt werden.

2.

▼ **verkettete Liste**

Vorteile:

- braucht keine größe bei der Initialisierung
- kann dynamisch wachsen, vergrößern und verkleinern während der Laufzeit ist somit kein problem

Nachteile:

- kann sehr viel Speicher benötigen
- es hat ein komplizierteres Handling als bei Arrays, man kann nicht direkt auf eine Element an einer bestimmten Stelle zugreifen

- dies führt zu einem höheren Fehlerpotenzial und die Geschwindigkeit leidet ebenfalls darunter

▼ **Double-Ended Queue**

Vorteile:

- Die Double-Ended Queue erlaubt es uns Vorne und Hinten Elemente einfügen und auch zu löschen
- es hat einen zyklischen Ablauf

Nachteil:

- hat keine maximale Größe, weshalb es theoretisch immer weiter wachsen könnte

▼ **Double-Ended Queue mit maximaler Größe**

Vorteile:

- sehr schnell im Vergleich zu denn anderen Datenstrukturen
- fixierter Speicher, kann nicht größer werden

Nachteile:

- man muss einen neuen Array erzeugen, falls man doch mehr Speicher braucht

Aufgabe 3

1.

Double - Ended Queue, da immer wieder Threads vorne eingefügt werden. Es darf aber keine maximale Größe haben da ein Prozess möglicherweise noch größer werden kann, da dieser Thread von anderen Prozessen klauen darf und kann.

2.

a) $10 - 3 = 7 \Rightarrow 25 - 7 = 18$, wenn z.B zu Beginn drei Mal gepopt wird, bevor überhaupt gepusht wird

$25 - 10 = 15$, wenn die 3-mal null vom `top` abhängig sind

b) $32 - 10 = 22$, wenn man von `first` ausgeht

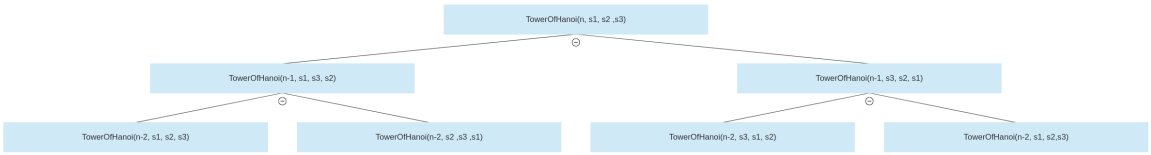
$32 - 15 = 17$, wenn man von den `dequeue` ausgeht

3.

a)	enqueue(5)	keine Rückgabe
b)	enqueue(1)	keine Rückgabe
c)	print(first())	5
d)	print(dequeue())	5
e)	enqueue(2)	keine Rückgabe
f)	print(dequeue())	1
g)	print(dequeue())	2
h)	print(isEmpty())	true

Aufgabe 4

1.



2.

TowerOfHanoi hat eine Komplexität von $2^n - 1$ und ein $\mathcal{O}(2^n)$, welches man anhand dieses Beispiels gut sehen kann:

$3\ disks : 2^3 - 1 = 7$

1. Disk 1 auf Stapel 2
2. Disk 2 auf Stapel 3
3. Disk 1 auf Stapel 3
4. Disk 3 auf Stapel 2
5. Disk 1 auf Stapel 1
6. Disk 2 auf Stapel 2
7. Disk 1 auf Stapel 2

Done.