hier ein paar Fragen:
Wann genau verwendet man denn eine Array Lists und woran kann man es in einem Programm erkennen?
Wie definiert man den Konstruktor einer Klasse und wann ist es notwendig einen Konstrukor zu verwenden?
Wann benutzt man parse und wie funktioniert es?
Wir werden uns noch weitere Fragen bis zum Sondertutoret überlegen und diese dann nachreichen.
Wir würden uns natürlich freuen wenn du paar kleine Fragen schon vor dem Sondertutorat beantworten könntest, natürlich nur wenn bei dir passt.

- Eine ArrayList ist wie eine normale Array, außer das die Anzahl der Elemente nicht fest ist. Eine ArrayList kann beliebig verlängert und gekürzt werden: https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html#add(int,%20E)

- Jede Klasse hat einen Konstruktor. Der Konstruktor ist eine besondere art Methode. Diese Methode (Konstruktor) wird so genannt wie die Klasse selbst und sie wird bei der Erzeugung eines Objektes ausgeführt. Wenn du keinen Konstruktor brauchst (z.B. weil du in deiner Klasse keine Variablen zum initialisieren hast), musst du den Konstruktor auch nicht hin schreiben. Der Compiler generiert aber trotzdem einen Leeren Konstruktor mit "public" zugriff.

- Parse bedeutet auf Deutsch ungefähr "durchsuchen" oder "durchlesen": https://translate.google.com/#auto/de/parse

  Diese Frage ist sehr allgemein.
  Ich werde mal das Beispiel aus stackexchange nehmen: http://stackoverflow.com/questions/787735/what-is-parse-parsing
  Hier wird ein String "geparsed" bzw. durchsucht um integers zu finden:
  int i = `Integer.parseInt("This is one long string and it includes 42");`

  Nun sollte i == 42 gelten.

  In diesem Fall ist parseInt eine Methode von der Klasse Integer: https://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html#parseInt(java.lang.String)

  Es gibt jedoch auch ein Interface zum parsen von XML Dateien.
  https://docs.oracle.com/javase/7/docs/api/org/xml/sax/Parser.html#parse(org.xml.sax.InputSource)

  Bestimmt gibt es noch viele mehr Methoden und Klassen die irgendetwas durchlaufen...

im Kapitel 7 wird die Klasse ArrayList definiert und erklärt. Da verstehe ich soweit alles gut. Jedoch wird dieser Klasse eine Methode Iterator() zugewiesen. Verstehe ich ebenfalls. Jedoch wird auf Folie 7.32 gesagt : Um eine breite Anwendbarkeit realisieren zu können, müssen Klassen wie ArrayList oder Iterator diese Flexibilität haben.

Ist Iterator nun eine Klasse oder eine Methode von ArrayList?

Iterator ist ein Interface bzw. eine Schnittstelle. Wenn es mehrere Klassen gibt die ähnliche eigenschaften haben, kann es sinnvoll sein eine Schnittstelle zu definieren die alle Klassen mit dieser Eigenschaft umfasst. Wenn eine Klasse diese Schnittstelle implementieren will, muss sie die Eigenschaften (zB. Methoden) haben die von dieser Schnittstelle vorgegeben sind.

Siehe: https://en.wikipedia.org/wiki/Interface_(Java)

Und gegen Ende der Übungen wurde die Interface auch behandelt...

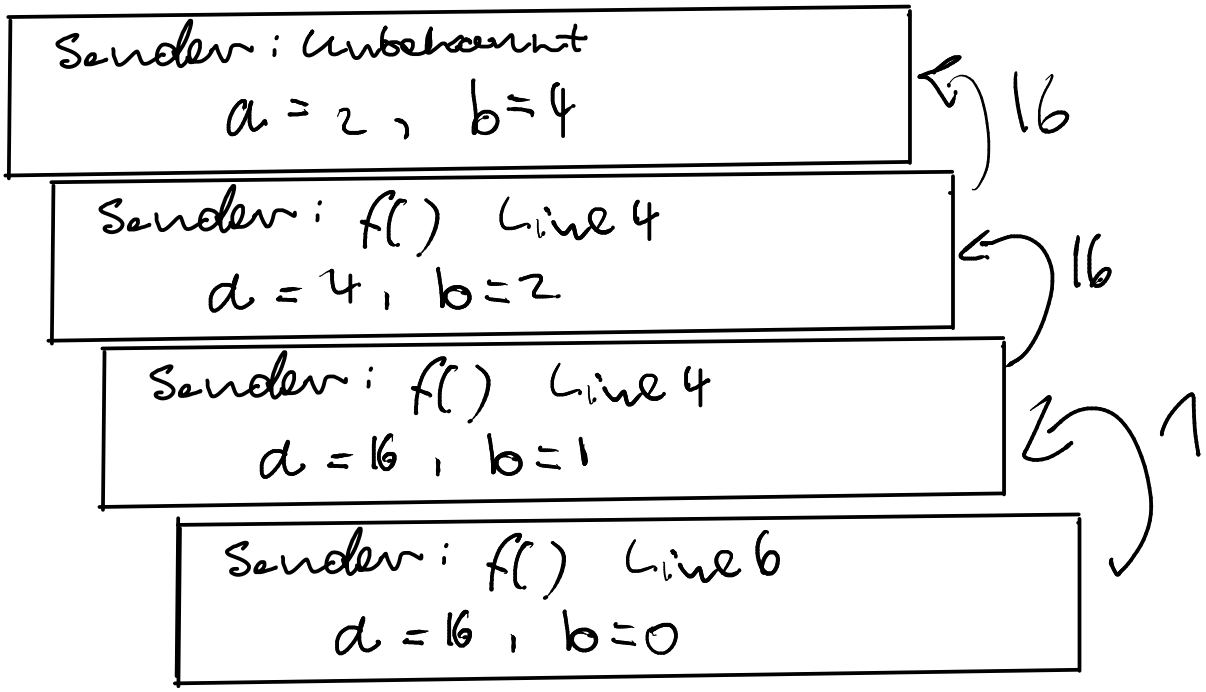
Aufgabe 4 **2+8 Punkte**

Gegeben sei folgendes Java-Programm:

```
public static int f(int a, int b) {
    if(b == 0) {
      return 1;                    //Zeile 1
    }
    if ((b % 2) == 0) {
      return f (a * a, b / 2);     //Zeile 4
    }
    return a*f (a, b - 1);         //Zeile 6
  }
```

(a) Was wird bei dem Aufruf $f(2,4)$ zurückgegeben?

(b) Zeichnen Sie die Activation Records für den Aufruf $f(2,4)$ zu dem Zeitpunkt, an dem die maximale Rekursionstiefe erreicht ist.



Aufgabe 13.8

Entscheiden Sie, ob die folgenden Aussagen richtig oder falsch sind.

	Richtig	Falsch
Die Menge der Algorithmen ist abzählbar.	☒	☐
Jede Funktion lässt sich durch einen Algorithmus realisieren.	☐	☒
Jeder Algorithmus beschreibt eine Funktion.	☒	☐
Die Frage, ob zwei Programme auf die gleiche Eingabe stets die gleiche Ausgabe liefern, ist entscheidbar.	☐	☒
Ein partiell korrektes Programm kann nie ein falsches Ergebnis zurückgeben.	☒	☐
Es gibt Programme, für die gezeigt werden kann, dass sie total korrekt sind.	☒	☐

Aufgabe 13.6

Eine Warteschlange (engl. „Queue“) ist eine Datenstruktur, die Objekte nach dem FIFO-Prinzip (First In – First Out) verwaltet, d.h. dass Objekte genau in der Reihenfolge ausgegeben werden, in der sie in die Warteschlange eingefügt werden. In dieser Aufgabe soll eine Warteschlange mit einer einfach verketteten Liste implementiert werden.

```
public class Queue {
  public Queue() {...}
  public void push(Object o) {...}
  public Object pop() {...}
  ...
  protected QueueNode head;
  protected QueueNode tail;
}

public class QueueNode {
  ...
  private Object content;
  ...
}
```

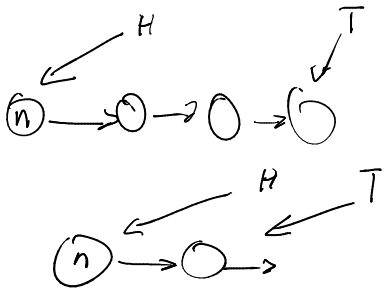
1. Geben Sie die vollständige Klassendefinition eines Speicherelements `QueueNode` mit geeigneten Zugriffsmethoden sowie typischem Konstruktor an und implementieren Sie diese.

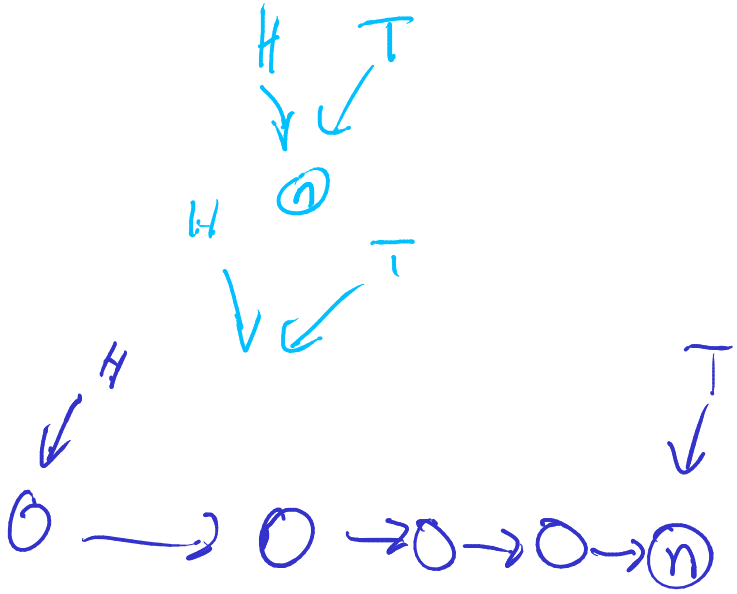
2. Implementieren Sie den Konstruktor sowie die Methoden `push()` und `pop()` der Klasse `Queue`. Die Methode `push()` fügt dem Speicher ein Element hinzu. Die Methode `pop()` liefert das nächste abzuarbeitende Element zurück und löscht dieses aus der Warteschlange.

3. Geben Sie den Aufwand für ihre Methoden in Abhängigkeit der Anzahl an gespeicherten Objekten an.

```
public class Queue {
	public Queue() {
		this.head = null;
		this.tail = null;
	}

	public void push(Object o) {
		QueueNode n = new QueueNode(o);
		n.setNext(this.head);
		this.head = n;

		if(this.tail == null) {
			this.tail = this.head;
		}
	}

	public QueueNode pop() {
		if(this.tail == null) {
			return null;
		}

		QueueNode n = this.tail;
		QueueNode m = this.head;

		if(this.tail == this.head) {
			this.tail = null;
			this.head = null;
			return n;
		}

		while(((m.next()).next() != null)) {
			m = m.next();
		}

		this.tail = m;
	}
}
```

```
public class QueueNode {
	public next() {
		return this.next;
	}

	public setNext(Object n) {
		this.next = n;
	}

	private Object content;
	private Object next;
}
```





$$\sum_{i=0}^{n-1} \sum_{j=0}^{i} 1 = \sum_{i=0}^{n} i = \frac{n(n-1)}{2}$$