Programmieren und Software-Engineering II Übung 15

Name:	Klasse:	Datum:

Lernziele:

- Objektorientierte Programmierung
- Vererbung
- Dynamische Datenstrukturen

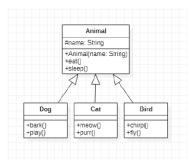
Aufgabe 0: Testverbesserung

Aufgabe 1: Geometrische Figuren (Ex_15_01_RectCubPyr)

Erstelle für das Beispiel aus der Theorieeinheit ein Java-Projekt für die Klassen Rectangle, Cuboid und Pyramid und geeignete Testfälle

Aufgabe 2: Animals (Ex_15_02_Animals)

Erstelle ein Java-Programm für nachfolgendes UML-Klassen-Diagramm und führe die angegebenen Testfälle aus. Überlege dir für jede Zeile was im Hintergrund passiert und welche Methode welcher Klasse aufgerufen wird. Schreibe deine Erkenntnisse in einen Kommentar zu jeder Zeile im Hauptprogramm.



```
public class Main {
   public static void main(String[] args) {
        // Create a Dog object
        Dog dog = new Dog("Rex");
        dog.eat();
        dog.play();
        dog.play();
        dog.play();
        // Create a Cat object
        Cat cat = new Cat("Garfield");
        cat.eat();
        cat.mew();
        cat.purr();
        cat.sleep();

        // Create a Bird object
        Bird bird = new Bird("Tweety");
        bird.eat();
        bird.chirp();
        bird.sleep();
    }
}
```

Rex is eating. Rex is barking. Rex is playing. Rex is sleeping. Garfield is eating. Garfield is meowing. Garfield is purring. Garfield is purring. Tweety is eating. Tweety is cating. Tweety is flying. Tweety is flying. Tweety is sleeping.

Aufgabe 3: LinkedList (Ex_15_03_LinkedList)

Erstelle eine einfach verkettete Liste, welche die untenstehenden Methoden implementiert. Jeder Knoten kennt seinen Nachfolger. Neue Elemente in der Liste werden hinten angehängt. Die Daten eines Knoten sind vom Typ Integer. Erstelle dazu eine Klasse LinkedList, um die Liste abzubilden und die Referenz zum ersten Knoten zu speichern und eine Klasse Node, die einen Knoten repräsentiert, mit der Referenz zu den Daten und einer Referenz zum nächsten Knoten. Implementiere folgende Methoden:

```
- isEmpty()
- contains (Integer data)
- getElement (int index)
- add(Integer data)
- insert(Integer data, int index)
- remove(int index)
- addAll(LinkedList 1)
- toString()
```

Du kannst dir auch folgendes Video zu verketteten Listen ansehen: https://studyflix.de/informatik/verkettete-liste-1433

[FREIWILLIG] Erstelle eine doppelt verkettete Liste (DoubleLinkedList) nach dem gleichen Prinzip. Dabei wird zusätzlich zum Nachfolger auch der Vorgänger in jedem Knoten gespeichert.