5. Übung zur Veranstaltung
Proinformatik: Objektorientierte Programmierung

Freie Universität Berlin
Fachbereich Mathematik und Informatik
Institut für Informatik, SoSe 2013
Prof. Dr. Marco Block-Berlitz

1. Die Funktion `addVektoren` soll derart erweitert werden (test-driven), dass Speicherüberläufe identifiziert und als Exception zurückgeliefert werden sollen.

2. Betrachten Sie das folgende Codefragment:

```
int[][] a = {{2,4,6,8}, {1,2,3}, {3,4,5}};
int[][] b = a;
int[][] c = (int[][]) a.clone();
c[2]      = a[1];
c[2][1]   = 6;
b[2][2]   = 7;

for (int i=0; i<a.length; i++)
   a[i][i]++;
```

   Welche Werte haben die Ausdrucke: `(a[1]==c[1])`, `(b[2] == c[2])`, `(a == c)`, `b[2][2]`, `c[1][1]` und `c[2][2]` nach der Ausführung und warum?

3. Wir haben ein Interface `Haustier` gegeben:

```
public interface Haustier {
   public String getName();
   public int getAlter();
   public String getBezeichnung();
   public String getTierstimme();
}
```

   Eine Klasse `Hund` implementiert die Methoden:

```
public class Hund implements Haustier {
   private String name;
   private int alter;

   public Hund(String name, int alter) {
      this.name  = name;
      this.alter = alter;
   }

   @Override
   public String getName() {
      return name;
   }

   @Override
   public int getAlter() {
      return alter;
   }

   @Override
   public String getBezeichnung() {
      return "Hund";
   }

   @Override
   public String getTierstimme() {
      return "wuff";
   }
}
```

Ein `Haustierhalter` kennt nur das Interface `Haustier`:

```
public class Haustierhalter {
   private Haustier meinHaustier;

   public Haustierhalter() {
      meinHaustier = null;
   }

   public void neuesHaustier(Haustier haustier) {
      meinHaustier = haustier;
   }

   public String getHaustierbezeichnung() {
      return meinHaustier.getBezeichnung();
   }
}
```

Ein kleines Testprogramm soll den Zusammenhang beider Klassen zeigen:

```
public class HaustierHaushalt {
   public static void main(String[] args) {
      Haustierhalter heinz = new Haustierhalter();
      Hund rambo = new Hund("Rambo", 3);
      heinz.neuesHaustier(rambo);
      System.out.println("Haustier von Heinz: "+
         heinz.getHaustierbezeichnung());
```

```
    }
}
```

Machen Sie sich mit den Klassen vertraut und erweitern Sie dieses Projekt um die folgende Funktionalität:

(a) Ein Haustierhalter darf mehrere Haustiere halten. Ein neues Hautier soll dabei über die Methode `neuesHaustier(Haustier h)` hinzugefügt werden können.

(b) Erweitern Sie die Menge der Haustiere um drei Ihrer Lieblingshaustiere.

(c) Haustiere leben leider nicht ewig, mal abgesehen von Tamagotchis. Erweitern Sie den Haustierhalter in der Art, dass er Haustiere auch wieder abgeben kann.

4. Arbeiten Sie sich in die Themen **Stack** und **Warteschlange** in Java ein. Versuchen Sie einen Stack und eine Warteschlange mit einem Array zu realisieren (kleiner Hinweis: bei der Implementierung der Warteschlange sollten Sie ein zyklisches Array simulieren). Sie können dabei voraussetzen, dass die Einträge in die Datenstrukturen nur vom Datentyp `int` sind.