JAVA-ÜBUNGSAUFGABEN INTERFACES

1)

Lösen Sie die 1. Aufgabe der Übungen zur Polymorphie (Bauernhof) mit Hilfe entsprechender selbst erstellter Interfaces.
Was ist das bessere Design (mit oder ohne Interfaces)?

2) Aufgabe

Zusätzlich zur vorigen Aufgabe spielt sich folgendes auf dem Bauernhof ab.
Auf dem Bauernhof gibt es noch das Nutztier Krokodil (ein Krokodil liefert z.B. Leder und Fleisch).
Für die Kinder im Bauernhof spielen einige Nutztiere die Rolle eines Freunds:
Alle Hennen und alle Kühe, aber kein Krokodil sind Freunde der Kinder, weil sie z.B. mit diesen Tieren spielen können.
Bei Bekannten befinden sich einige Freunde der Kinder zur Pflege. Sie werden im Folgenden mit Pflegefreunde bezeichnet.
Legen Sie dazu in main(..) eine Menge von 3 Pflegefreunden in dem (nicht dynamischen) Array pflegefreunde[] an.
Alle Freunde der Kinder (aber nicht das Krokodil) geben den Kindern Geschenke.

2.1)

Die Klassen Kuh und Henne müssen die Methode printGeschenkl(...) enthalten.

2.2)

Für die Freunde müssen, wie für alle Nutztiere, der mittlere Tierwert, der maximale Tierwert und sein Index berechnet werden können.
d) **getGroesstesGeschenk**(...)
Berechnet das größte Geschenk (d.h. mit dem größten Wert) eines Arrays von Freunden
Kuh: Milchleistung / 2
Henne: Legeleistung * 3

2.3)

Die Methode
printAllFreunde(...)
gibt folgende Informationen eines Arrays von Freunden auf dem Bildschirm aus:
Alle Attribute, Tierwerte und Geschenke eines jeden Freundes,
den Mittelwert aller Tierwerte,
den maximalen Tierwert des Arrays,
den Index des maximalen Tierwerts des Arrays,
das größte Geschenk (d.h. mit dem größten Wert) eines Arrays von Freunden

2.4)

Kann man diese Aufgabe ohne Interfaces lösen?

3) Aufgabe

Zu der Sprache Java gehört (genauer im Java-API) die Klasse `Arrays` (siehe Java-API)
Diese besitzt die Methode `sort(Object[] a)`
Mit dieser Methode lässt sich ein Feld sortieren.
(Die Klasse `Object` ist Oberklasse jeder Klasse in Java. Sie existiert automatisch und braucht deswegen nicht erstellt zu werden).

Zitat aus der Java-API
"All elements in the array must implement the Comparable interface."

Bemerkung:
Es gibt noch eine weitere (überladene) Methode sort, mit der man ein Array sortieren kann:
sort(T[] a, Comparator<? super T> c)
Hier muß also zusätzlich noch zum Feld als Parameter eine Referenz auf ein Interface Comparator übergeben werden (also auf ein Objekt, das das Interface Comparator implementiert).

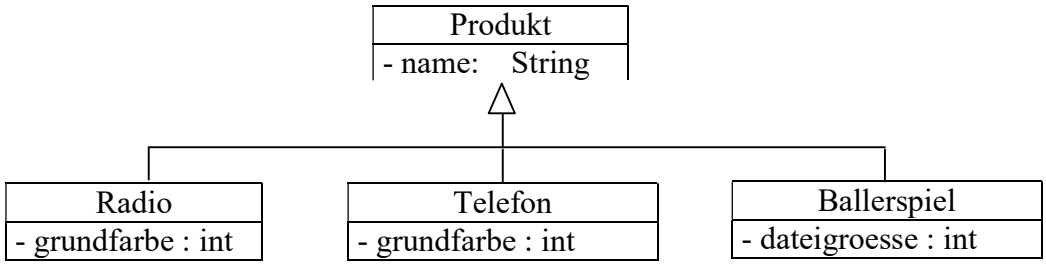
a) Sortieren Sie mit Hilfe der Methode sort(...) ein Feld von Girokonten (oder irgendeinem anderen selbstgebastelten Datentyp, wie z.B. Autos, Hennen bzw. Kühen usw.)

b) Benutzen Sie die andere Version (zusätzliche Übergabe eines Interfaces), die o.g. Felder zu sortieren (siehe Bemerkungen).

Bemerkung:
Aus allen folgenden UML-Diagrammen soll nicht hervorgehen, ob es sich um abstrakte oder "normale" Klassen handelt.

4) Aufgabe

Gegeben ist das folgende abgespeckte, **fest vorgegebene** UML-Digramm (Firma stellt verschiedene Produkte her). Die Klassenhierarchie darf also **nicht** verändert werden.



a) Erzeugen Sie aus dem folgenden UML-Diagramm die entsprechenden Klassen mit den jeweiligen Konstruktoren, set- und get-Methoden. Keine weiteren Attribute einfügen!
Wichtig: Jede Grundfarbe wird durch einen Integer-Wert dargestellt, wie z.B:
-100: weiß 1:gelb, usw.
Umgekehrt entspricht jedem Integer-Wert eine Grundfarbe!
Bitte keine “Umrechnungstabelle“ der Zahlenwerte in tatsächliche Grundfarben erstellen!
Hier ist eine Grundfarbe ein Zahlenwert, mehr nicht!

b) Das Programm soll multipersonal entwickelt werden: Zu Testzwecken sollen die Entwickler der einzelnen Klassen gezwungen werden, die Methode
void printBeschreibung()
zu entwickeln, die die Namen der Attribute mit den zugehörigen Werten auf dem Bildschirm ausgibt.
In dem Feld "produkte" sollen verschiedene Produkte (Radio, Telefon, Ballerspiel) abgespeichert werden.
Danach sollen die Werte der Attribute mit der Methode printBeschreibung() ausgegeben werden. Machen Sie folgendes in der Methode main()
b1) Erzeugen Sie ein Radio, Telefon, Ballerspiel.
b2) Speichern Sie diese 3 Produkte in dem Feld.
b3) Geben Sie mit Hilfe einer Schleife und der Methode printBeschreibung() die Eigenschaften der jeweiligen Produkte auf dem Bildschirm aus.

c)
Beachten Sie:
Ein Radio und ein Telefon sind Gegenstände und haben deshalb eine Grundfarbe.
Dagegen besitzt eine Software, wie ein Ballerspiel keine Grundfarbe.
Um sich von anderen Firmen abzuheben, wird der Name des Produkts (Schriftzug, der sich auf dem Produkt befindet) mit einer individuellen Farbe, der sogenannten Individualfarbe, versehen. Diese Farbe soll nur vom Produktnamen und der Grundfarbe abhängen (z.B. Länge des Produktnamens + grundfarbe).

In dem Feld "gegenstaende" sollen neue, sich in der Entwicklung befindliche Radios und Telefone abgespeichert werden (konkret: das blaue (Kodierung für blau selbst wählen) Radio mit dem Namen "RadioHoliday" und das gelbe (Kodierung für gelb selbst wählen) Telefon mit dem Namen "Urlausbtelefon") und dann mit Hilfe einer Schleife und der Methode getIndividualfarbe() die Individualfarbe des jeweiligen Schriftzugs auf dem Bildschirm ausgegeben werden.

Es gibt mehrere Möglichkeiten, die Methode getIndividualfarbe() unterzubringen:
c1) Was wäre der Nachteil der Lösung, in der man die Methode getIndividualfarbe() in der Klasse Produkt implementiert?
Realisieren Sie dies durch ein Programm.

c2) Was wäre der Nachteil der Lösung, in der man die Methode getIndividualfarbe() in der Klasse Produkt als abstrakte Methode angibt?
Realisieren Sie dies durch ein Programm.

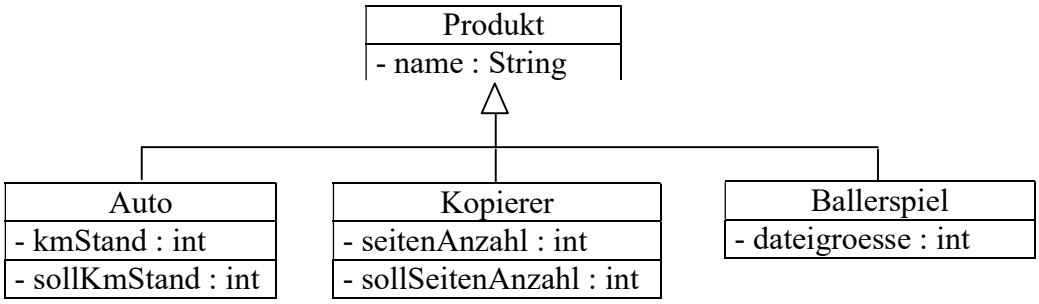
c3) Beurteilen Sie die folgende Lösung:
Die Methode getIndividualfarbe() wird nur (und sonst nirgends) in die Klasse Radio und Telefon implementiert.
Realisieren Sie dies durch ein Programm.

c4) Was ist die beste Lösung ?
Realisieren Sie dies durch ein Programm.

d) (nicht einfach: es muß ein cast ClassCastException oder instanceof benutzt werden)
Geben Sie mit Hilfe einer Schleife und den Methoden printBeschreibung() und getIndividualfarbe() die Eigenschaften des Feldes "produkte" auf dem Bildschirm aus (einschließlich der Individualfarben der Telefone und Radios).

5) Aufgabe

Gegeben ist das folgende abgespeckte, **fest vorgegebene** UML-Digramm (Firma stellt verschiedene Produkte her). Die Klassenhierarchie darf also **nicht** verändert werden.



a) Erzeugen Sie aus dem folgenden UML-Diagramm die entsprechenden Klassen mit den jeweiligen Konstruktoren, set- und get-Methoden. Keine weiteren Attribute einfügen!
Autos und Kopierer müssen zu bestimmten Zeiten zur Inspektion (Wartung). Dies hängt von den gefahrenen Kilometern bzw. der Anzahl kopierter Seiten ab.

b) Das Programm soll multipersonal entwickelt werden: Zu Testzwecken sollen die Entwickler der einzelnen Klassen gezwungen werden, die Methode
void printBeschreibung()
zu entwickeln, die die Namen der Attribute mit den zugehörigen Werten auf dem Bildschirm ausgibt.
In dem Feld "produkte" sollen verschiedene Produkte (Auto, Kopierer, Ballerspiel) abgespeichert werden.
Danach sollen die Werte der Attribute mit der Methode printBeschreibung() ausgegeben werden. Machen Sie folgendes in der Methode main()
b1) Erzeugen Sie ein Auto, Kopierer, Ballerspiel.
b2) Speichern Sie diese 3 Produkte in dem Feld.
b3) Geben Sie mit Hilfe einer Schleife und der Methode printBeschreibung() die Eigenschaften der jeweiligen Produkte auf dem Bildschirm aus.

c)
Beachten Sie:
Im Gegensatz zu einem Ballerspiel müssen Autos und Kopierer regelmäßig zur Inspektion gebracht werden.

In dem Feld "gebrauchtWaren" sollen folgende gebrauchte Autos und Kopierer abgespeichert werden:
Auto: "VW-Käfer-1"; Km-Stand 13000 Km ; nächste Inspektion bei 9000 Km
Kopierer: "MultiStar1", gedruckte Seiten: 40000 Seiten ; nächste Inspektion bei 10000

Dann soll mit Hilfe einer Schleife und der Methode wartungsbericht() auf dem Bildschirm ausgegeben werden, ob die gebrauchte Ware zur Inspektion muß oder nicht bzw. um wieviel Seiten bzw. Kilometer überzogen wurde.
Man könnte zusätzlich auch noch eine Methode wartungsbedarf() erstellen, die die Differenz zwischen ist dem Ist- und Sollwert zurückgibt.

Es gibt mehrere Möglichkeiten, die Methode wartungsbericht() bzw. wartungsbedarf() unterzubringen:
c1) Was wäre der Nachteil der Lösung, in der man diese Methode in der Klasse Produkt implementiert?
Realisieren Sie dies durch ein Programm.

c2) Was wäre der Nachteil der Lösung, in der man diese Methoden in der Klasse Produkt als abstrakte Methode angibt?
Realisieren Sie dies durch ein Programm.

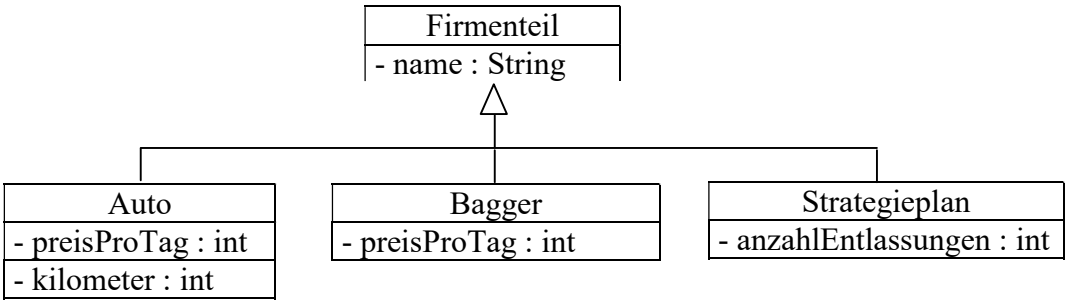
c3) Beurteilen Sie die folgende Lösung:
Diese Methoden nur (und sonst nirgends) in die Klasse Auto und Kopierer implementieren.
Realisieren Sie dies durch ein Programm.

c4) Was ist die beste Lösung ?
Realisieren Sie dies durch ein Programm.

d) (nicht einfach: es muß ein cast ClassCastException oder instanceof benutzt werden)
Geben Sie mit Hilfe einer Schleife und den Methoden printBeschreibung() und - falls es sich um ein Auto oder einen Kopierer handelt - wartungsbericht() die Eigenschaften der jeweiligen Produkte des Feldes "produkte" auf dem Bildschirm aus.

6) Aufgabe

Gegeben ist das folgende abgespeckte, **fest vorgegebene** UML-Digramm (Firma besteht aus verschiedenen Teilen). Die Klassenhierarchie darf also **nicht** verändert werden.



a) Erzeugen Sie aus dem folgenden UML-Diagramm die entsprechenden Klassen mit den jeweiligen Konstruktoren, set- und get-Methoden. Keine weiteren Attribute einfügen!
Autos und Bagger können (im Gegensatz zu einem Strategieplan: der ist geheim!!) vermietet werden. Die Miete hängt von den gefahrenen Kilometern (Preis pro Kilometer: 0,5 Euro als konstanter Wert, der nicht verändert werden kann) bzw. der Anzahl der Tage ab, die das Auto gemietet wurde.

b) Das Programm soll multipersonal entwickelt werden: Zu Testzwecken sollen die Entwickler der einzelnen Klassen gezwungen werden, die Methode
void printBeschreibung()
zu entwickeln, die die Namen der Attribute mit den zugehörigen Werten auf dem Bildschirm ausgibt.
In einem Feld sollen verschiedene Firmenteile (Auto, Bagger, StrategiePlan) abgespeichert werden.
Danach sollen die Werte der Attribute mit der Methode printBeschreibung() ausgegeben werden. Machen Sie folgendes in der Methode main()
b1) Erzeugen Sie ein Auto, Bagger, Strategieplan.
b2) Speichern Sie diese 3 Firmenteile in dem Feld.
b3) Geben Sie mit Hilfe einer Schleife und der Methode printBeschreibung() die Eigenschaften der jeweiligen Firmenteile auf dem Bildschirm aus.

c)
Beachten Sie:
Im Gegensatz zu einem Strategieplan können Autos und Bagger vermietet werden.

In dem Feld "vermietungen" sollen folgende Autos und Bagger abgespeichert werden:
Auto: "VW-Käfer-1"; Preis pro Tag: 60 ; gefahrene Kilometer: 3000
Bagger: "Bagwahnr1", Preis pro Tag: 300:

Dann soll mit Hilfe einer Schleife und der Methode getMietbericht(int tage) die entsprechenden Informationen auf dem Bildschirm ausgegeben werden.
Man könnte zusätzlich auch noch eine Methode getMiete(int tage) erstellen, die die Miete (in Euro) zurückgibt.

Es gibt mehrere Möglichkeiten, die Methode getMietbericht(int tage) bzw. getMiete(int tage) unterzubringen:
c1) Was wäre der Nachteil der Lösung, in der man diese Methode in der Klasse Firmenteil implementiert?
Realisieren Sie dies durch ein Programm.

c2) Was wäre der Nachteil der Lösung, in der man diese Methoden in der Klasse Firmenteil als abstrakte Methode angibt?
Realisieren Sie dies durch ein Programm.

c3) Beurteilen Sie die folgende Lösung:
Diese Methoden nur (und sonst nirgends) in die Klasse Auto und Kopierer implementieren.
Realisieren Sie dies durch ein Programm.

c4) Was ist die beste Lösung ?
Realisieren Sie dies durch ein Programm.

d)
d1) (nicht einfach: es muß ein cast ClassCastException oder instanceof benutzt werden)
Geben Sie mit Hilfe einer Schleife und den Methoden printBeschreibung() und - falls es sich um ein Auto oder einen Bagger handelt - getMietbericht(int tage) die Eigenschaften der jeweiligen Firmenteile auf dem Bildschirm aus.

d2) Berechen Sie die Summe aller Mieteinnahmen und geben diese auf dem Bildschirm aus.

e) Schreiben Sie eine Methode vergleiche, die zwei Gebrauchtwaren miteinander vergleicht und die Gebrauchtware zurückgibt, mit weniger Miete einbringt.

7) Aufgabe

a) Erstellen Sie die Klasse Girokonto mit den Attributen inhaber, kontonummer, kontostand und den zugehörigen set- und get-Methoden.

b) Erstellen Sie 5 Objekte dieser Klasse und speichern diese in dem Feld girokonten ab.

c) Nun sollen die Girokonten, d.h. diese o.g. Feld sortiert werden.
Das könnte man natürlich - mit einem nicht unerträglichen Aufwand - selbst machen.
Allerdings könnte man auch die Hilfe der Java-Standardklasse Arrays (siehe Java-Dokumentation: Arrays, Comparable) verwenden.
Diese besitzt die statische Methode sort(...) mit einem bzw. 2 Parametern.

1. Möglichkeit:
Arrays.sort(girokonten);
Woher weiß die Methode sort, nach welchem Kriterium sie die Girokonten miteinander vergleichen soll?
Dazu muß die Klasse Girokonten gezwungen werden, eine Methode zu implementieren, die zwei Objekte miteinander vergleicht. Die Methode heißt
public int compareTo(Object k)
und zur Implementierung wird man dadurch gezwungen, daß die Klasse Girokonto das Interface Comparable implementiert.

Bemerkung:
Der Parameter k muß noch in ein Girokonto gecastet werden.

2. Möglichkeit:
Arrays.sort(girokonten, inhaberVergleicher);
Der 2. Parameter der Methode ist Objekt einer Klasse, die das Interface Comparator implementiert. Dieses Interface zwint die, zu dem Objekt inhaberVergleicher zugehörigen Klasse, die Methode
public int compare(Object k1, Object k2)
zu implementieren.

8) Aufgabe (Listen, Mengen, Permutationen)

(Eine gute Übungsaufgabe, um Listen zu manipulieren.)
Zur Information:
Für n = 4
wird folgende Ursprungsliste (wie beim Kilometerzähler) erzeugt:

```
(1,1,1,1); (1,1,1,2); (1,1,1,3); ... ; (4,4,4,4)
```

In dieser Liste befinden sich 4-er Paare, die sich nur in der Anordnung
ihrer Reihenfolge unterscheiden (Permutationen) und deren Elemente alle verschieden sind , wie z.B:

```
(1,2,3,4); (1,2,4,3); ... ; (4,3,2,1);
```

A1)
Filtern Sie diese Permutationen aus der Ursprungsliste und speichern diese in einer anderen Liste ab.

A2) (schwierig)
Lösen Sie die Aufgabe für beliebiges n>1

Tipp:
Wenn man ein 4-er Paar in eine ArrayList abspeichert, kann man diese Liste in eine Menge umwandeln. Da eine Menge keine Duplikate enthält, kommt kein Element mehrfach vor.
Dann kann man diese Menge wieder in eine ArrayList umwandeln.
Ist die Länge dieser ArrayList gleich 4, dann ist das 4-er Paar eine Permutation.

9) Aufgabe (Listen, Mengen, Permutationen)

(Eine gute Übungsaufgabe, um Listen zu manipulieren.)
Zur Information:
Für n = 4
wird folgende Ursprungsliste (wie beim Kilometerzähler) erzeugt:

```
(1,1,1,1); (1,1,1,2); (1,1,1,3); ... ; (4,4,4,4)
```

In dieser Liste befinden sich 4-er Paare, die sich nur in der Anordnung
ihrer Reihenfolge unterscheiden, wie z.B:

```
(1,1,1,2); (1,1,2,1); (1,2,1,1); (2,1,1,1)
```

oder

```
(1,2,3,4); (1,2,4,3); (1,3,2,4);  (1,3,4,2) ... (4,3,2,1)
```

A1)
Alle Duplikate sollen entfernt und das Ergebnis in einer neuen Liste gespeichert werden.
Das Ergebnis sieht dann wie folgt aus:

```
(1,1,1,1); (1,1,1,2); (1,1,1,3); (1,1,1,4); (1,1,2,2); (1,1,2,3);
(1,1,2,4); (1,1,3,3); (1,1,3,4); (1,1,4,4); (1,2,2,2); (1,2,2,3);
(1,2,2,4); (1,2,3,3); (1,2,3,4); (1,2,4,4); (1,3,3,3); (1,3,3,4);
(1,3,4,4); (1,4,4,4); (2,2,2,2); (2,2,2,3); (2,2,2,4); (2,2,3,3);
(2,2,3,4); (2,2,4,4); (2,3,3,3); (2,3,3,4); (2,3,4,4); (2,4,4,4);
(3,3,3,3); (3,3,3,4); (3,3,4,4); (3,4,4,4); (4,4,4,4);
```

A2) (schwierig)
Lösen Sie die Aufgabe für beliebiges n>1

I) Tipps:
T1) Sortieren Sie in der Ursprungsliste (ArrayList) jedes 4-er Paar in aufsteigender Reihenfolge. Damit wird z.B: aus (4,3,2,1) das 4-er Paar (1,2,3,4)

T2) Durch die vorige Sortierung kommen dann mehrere gleiche 4-er Paare in der Ursprungsliste vor. Beseitigen Sie die Duplikate dadurch, daß Sie die Ursprungsliste in eine Menge (HashSet) verwandeln (in einer Menge kommt jedes Element genau einmal vor).

T3) Wandeln Sie die Menge wieder in eine ArrayList um. Aus optischen Gründen muß diese dann noch lexikografisch sortiert werden.
Benutzen Sie dazu die Methode sort der Klasse
Beispiel:

```
Collections.sort(arraylist);
```

10) Aufgabe (Rätsel der Woche 26.4.20)

Zwei Mathematikerinnen begegnen sich zufällig auf der Straße.
"Wie alt sind eigentlich deine vier Kinder inzwischen?", fragt die erste.
"Die Summe der Alter aller 4 Kinder ergibt 15 Jahre, und das Produkt ergibt die Hausnummer hinter dir", antwortet die zweite.
Die erste Mathematikerin überlegt eine Weile und sagt dann:
"Das reicht mir nicht, da brauche ich doch noch mehr Infos."
Darauf sagt die zweite Frau: "Also Zwillinge hab ich ja keine."
"Jetzt weiß ich Bescheid", entgegnet die erste.
Wie alt sind die vier Kinder der zweiten Mathematikerin?
Hinweis: Das Alter wird in ganzen Zahlen angegeben.

Hinweise zur programmtechnischen Lösung:
A1) Erstellen Sie die Klasse Feld5, deren Attribut ein Array der Länge 5 ist.
In diesem Feld wird eine mögliche Alterskombination (der 4 Kinder) mit Gesamtalter 15 und dem Gesamtaltersprodukt der 4 Kinder gespeichert, wie z.B:
1,2,3,4,24
1,2,3,4 sind die Altersangaben der 4 Kinder und 24 ist das Produkt der 4 Altersangaben.

A2) Erstellen Sie die ArrayList:
ArrayList<Feld5> arraylistMitSumme_15;
In dieser müssen alle möglichen Kombinationen mit Summe 15 gespeichert werden, wie z.B:
1,2,1,3,6
1,1,1,1,1

A3) Löschen Sie in arraylistMitSumme_15 alle mehrfachen Alterskombinationen heraus, wie z.B:
1,2,3,4,24
4,3,2,1,24
4,3,1,2,24
usw.

A4) Sortieren Sie die verbleibende Liste nach dem Produkt der Alterskombinationen

11) Simulationen

Es sollen verschiedene Simulationen implementiert werden. Dazu gehören u.a:
a) Logistische Gleichung (Verhulst), siehe Wikipedia

$a_{n+1} = a_n \cdot (1-a_n) \cdot k$ wobei $0 \leq a_n \leq 1$ und $r \geq 0$

Mit dieser Simulation soll die Entwicklung der Anzahl der Kranken (an wird in Prozent angegeben) in einer Epidemie modelliert werden.

b) Random Walk
Die Bewegung eines Betrunkenen (wissenschaftlich: random walk) soll simuliert werden, indem sich dieser nur über Zufall in x- und y-Richtung fortbewegt:
Mit jeweils der Wahrscheinlichkeit 0,25 einen Schritt nach oben, nach unten, nach rechts und nach links.

Die Anzahl der Simulationsschritte muß bei jeder Simulation gespeichert werden.
genauso wie die Werte, die bis dahin durchlaufen werden.

In einer Klasse Simulationsverwaltung können verschiedenene Simulationen gespeichert werden (z.B. eine für einen Random Walk und eine für eine Epidemie).
Es muß sichergestellt werden, dass in jeder Simulation eine Methode print() existiert, die die simulierten Werte einer Simulation (bis zur Anzahl der Simulationsschritte) auf dem Bildschirm ausgibt.

Außerdem müssen die Simulationen noch sortiert werden (nach der Anzahl der Simulationsschritte).