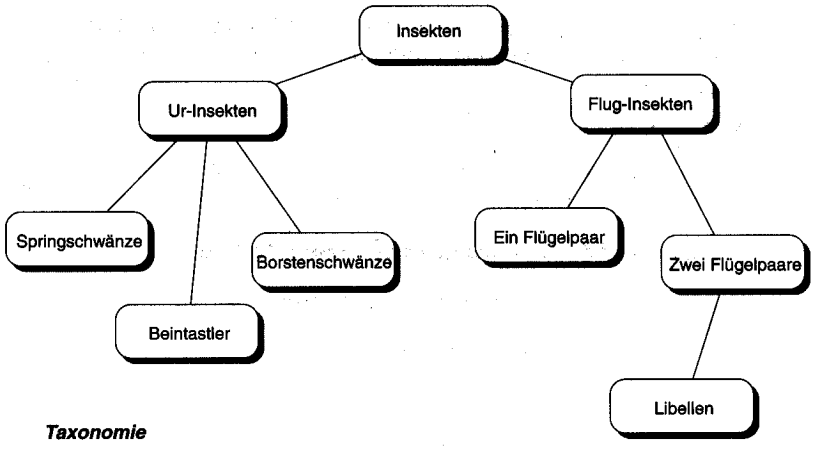
Übungen Kapitel 6

6.1 () Fragen (Papieraufgabe) *(Objektorientierung Ideen)*

- Was sind die zentralen Ideen der Objektorientierung?
- Erklären Sie die Historie der Objektorientierung.
- Sie sind Entwicklungsleiter eines Software Hauses und müssen Ihr Management von den Vorteilen der objektorientierten Denkweise im Vergleich zur prozeduralen Denkweise überzeugen. Wie machen Sie das?
- „Unter Taxonomie [griech.], 1) versteht man in der Zoologie und Botanik als Zweig der Systematik die Wissenschaft und Lehre von dem praktischen Vorgehen bei der Einordnung der Organismen in systematische Kategorien." (aus Meyers Lexikon)). Erklären Sie die Vererbung – ein zentrales Merkmal der Objektorientierung – und übertragen Sie die Idee einer Taxonomie auf die Vererbung. Überlegen Sie sich ein Beispiel, an dem Sie die Vererbung erklären.



Taxonomie

6.2 (**) **Punkte und Dreiecke** *(Klasse, Methoden, Konstruktor)*

a. Entwickeln Sie eine Klasse `Point`, die einen Punkt im kartesischen Koordinatensystem repräsentiert:

- **`Point(double x, double y)`**
  Konstruktor für den Punkt (x,y)
- **`double x()`**
  Liefert die x-Koordinate dieses Punktes
- **`double y()`**
  Liefert die y-Koordinate dieses Punktes
- **`String toString()`**
  Liefert die Koordinaten des Punktes als String zurück
  (z.B. `Point p = new Point(4,10);`
  `System.out.println(p)` ist die verkürzte Schreibweise für
  `System.out.println(p.toString())` und gibt `x: 4 y: 10` aus.)

b. Entwickeln Sie eine Klasse `Triangle`, die Dreiecke im kartesischen Koordinatensystem repräsentiert. Folgende Methoden soll die Klasse zur Verfügung stellen:

- **`Triangle(Point a, Point b, Point c)`**
  Konstruktor für ein Dreieck mit den Punkten a, b und c.

- **`Getter`** Methoden für die Punkte

- **`double perimeter()`**
  Liefert den Umfang des Dreiecks

- **`double area()`**
  Liefert die Fläche des Dreiecks

- **`Point lowerLeft()`**
  Liefert die linke untere Ecke des umschreibenden Rechtecks dieses Dreiecks. Das umschreibende Reckteck ist das kleinste achsenparallele Rechteck, das dieses Dreieck vollständig enthält.

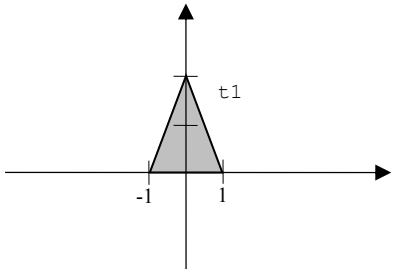
- **`Point upperRight ()`**
  Liefert die rechte obere Ecke des umschreibenden Rechtecks dieses Dreiecks.

- **`Triangle moved(double dx, double dy)`**
  Erzeugt ein horizontal um `dx` und vertikal um `dy` verschobenes Dreieck. Die Methode lässt dieses Dreieck unverändert und liefert das verschobene Dreieck als neues Objekt.

- **`Triangle zoomed(double f)`**
  Erzeugt ein um den Faktor f gestrecktes Dreieck. Zentrum der Streckung ist der Koordinatenursprung. Die Methode lässt dieses Dreieck unverändert und liefert das gestreckte Dreieck als neues Objekt.

- **`Triangle zoomed (Point p, double f)`**
  Erzeugt ein um den Faktor `f` gestrecktes Dreieck. Zentrum der Streckung ist der

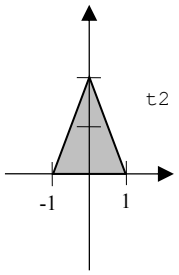
Punkt `p`. Die Methode lässt dieses Dreieck unverändert und liefert das gestreckte Dreieck als neues Objekt.

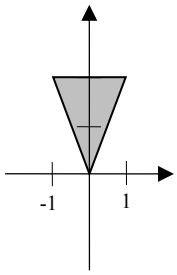


Das folgende Programmstück erzeugt ein Dreieck `t1` und gibt dessen Umfang und Fläche aus:

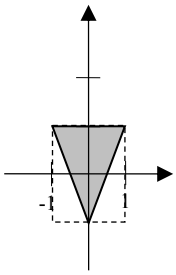
```
Triangle t1 = new Triangle(   new Point(-1,0),
                              new Point(0,2),
                              new Point(1,0));
System.out.println(t1.perimeter());     // 6.47…
System.out.println(t1.area());          // 2.0
```

Das nächste Programmstück erzeugt ein Dreieck `t2`, dreht `t2` um 180 Grad und verschiebt den Mittelpunkt in den Ursprung.





zoomed(new Point(0,1),-1)



moved(0,-1)

```
Triangle t2 = t1;

t2 = t2.zoomed(new Point(0,1),-1);
t2 = t2.moved(0,-1);

//Ausgabe des umschliessenden Recktecks
System.out.println(t2.lowerLeft());    // -1 -1
System.out.println(t2.upperRight());   //  1  1
```

Lösungshinweis: Benutzen Sie die erforderlichen mathematischen Funktionen aus der `Math` Bibliothek.

6.3 (*) Uhr *(Objekt/Klasse)*

Gegeben ist die folgende Klasse `Time` sowie eine Klasse `TimeTest`, die die Klasse Time testen soll:..\...\CodeBeispiele\Time.

a. Arbeiten Sie sich in die Klasse `Time` mit überladenen Konstruktoren ein und lassen das Testprogramm laufen.

b. Ergänzen Sie die Klasse `Time` um die Methoden

- **`public void incrementMinute()`**
  Zählt die aktuelle Zeit um eine Minute hoch

- **`public void incrementHour()`**
  Zählt die aktuelle Zeit um eine Stunde hoch

- **`public void tick()`**
  Zählt die aktuelle Zeit bei jedem Aufruf um eine Sekunde hoch

Testen Sie diese neuen Methoden, in dem Sie diese in der Testklasse `TimeTest` entsprechend aufrufen und `tick()` in einer Endlosschleife aufrufen und mit `System.out` die aktuelle Zeit auf dem Bildschirm ausgeben lassen.

6.4 Datum (*) *(Overloading und Datenkapselung)*

Schreiben Sie eine Klasse **`Date`** mit den Instanzvariablen **`month, day`** und **`year,`** die alle vom Datentyp **`int`** sind.
a. Implementieren Sie drei überladene Konstruktoren um **`Date`** Objekte zu erzeugen, die folgende Datumsformate unterstützt:

- **`Date(int month, int day, int year)`**
  Erzeugt ein Datum, wobei **`month`** den Wertebereich [1..12], **`day`** den Wertebereich [1..31] und **`year`** den Wertebereich [-2000, 3000] hat.
  Fehlerbehandlung dürfen Sie in diesem Fall ausnahmsweise weglassen.

- **`Date(String month, int day, int year)`**
  Erzeugt ein Datum, wobei als Monate der Monatsnamen übergeben wird z.B**.**
  `Date d = new Date("Februar",14, 2009);`
  Fehlerbehandlung dürfen Sie in diesem Fall ausnahmsweise weglassen.

- **`Date(int month, int year)`**
  Erzeugt ein Datums Objekt mit Monat und Jahr. Den Tag müssen Sie intern entsprechend mit einem default vorbelegen.
  Fehlerbehandlung dürfen Sie in diesem Fall ausnahmsweise weglassen.

- **`String toString()`**
  Liefert die Werte des Datums als String zurück.

- **`Getter`** Methoden für Tag, Monat und Jahr

  Achtung: Die Datumsklasse soll nur die oben angegebenen 3 Instanzvariablen enthalten (nicht mehr) , dabei soll `day`, `year` und `month` als `int` gespeichert werden.

b. Schreiben Sie eine Klasse **`DateTest`** mit einer **`main`** Methode – auch häufig Testtreiber genannt – und testen Ihre Klasse für verschiedene Datumsformate. Dabei verwenden Sie Ihre `toString` Methode.

c. Ändern Sie jetzt den Datentyp für die Instanzvariable **`month`** aus b) von `int` auf `String`. Lassen Sie alle Signaturen aller Methoden völlig gleich. Passen Sie die Implementierung der Konstruktoren geeignet an. Verwenden Sie das exakt identische Testprogramm aus b). Falls Sie alles korrekt gemacht haben, erzeugt ihr Testprogramm exakt die gleichen Ausgaben wie unter b).

Erklären Sie an diesem Beispiel die Datenkapselung und den damit verbundenen Vorteil für
- den Anwender (also das Testprogramm) der Klasse
- den Entwickler der Klasse

6.5 (*) Magisches Quadrat *(Klasse und Klassenmethode)*

In der Vorlesung ist ein objektbasierter Entwurf für magische Quadrate vorgestellt worden. Hierzu liegt unter CodeBeispiel\Magische Quadrate ein Klassengerüst. Implementieren und vervollständigen Sie alle Methoden.
Testen Sie ihre Methode für folgende Quadrate:

```
int[][] a1 = { {2,2,2,2}, {2,2,2,2}, {2,2,2,0},{2,2,2,2}};
int[][] a2 = { {2,2,2,2}, {2,2,2,2}, {2,2,2,2},{2,2,2,2}};
int[][] a3 = { {2,7,6}, {9,5,1}, {4,3,8,}};
int[][] a4 = { {2,3,3,4}, {2,2,5,2}, {5,7,2,2},{4,7,9,2}};
int[][] a5 = { {1,3,3,1}, {3,2,2,2}, {5,3,3,2},{4,7,9,4}};
```

Außerdem lassen Sie sich für einige `n` Werte magische Quadrate durch
`erzeugeMagischesQuadrat` erzeugen und testen für diese `isMagic`.

Lösunghinweis: Bei der Methode `isMagic` müssen Sie u.a. prüfen, ob jede Zahl von 1 bis n^2 genau einmal vorkommt (siehe Definition Magisches Quadrat). Für diese Prüfung verwenden Sie ein `boolean` array, in dem Sie festhalten, welche Werte in der Matrix vorkommen. Für jeden Wert in der Matrix sehen Sie in diesem array nach, ob er bereits einmal gefunden wurde.

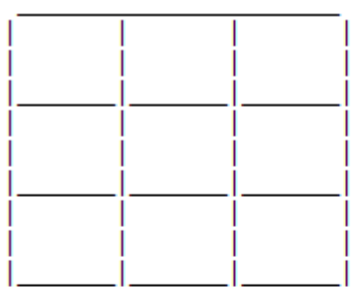
6.6 (**) Zusatzaufgabe: Projektaufgabe oder TicTacToe (Objektorientierter Entwurf)

Implementieren Sie das Spiel `TicTacToe`. Dazu liegen im moodle bereits 3 java Klassen `TicTacToeMain, TicTacToe, TicTacToeConsole`.
Beachten Sie:

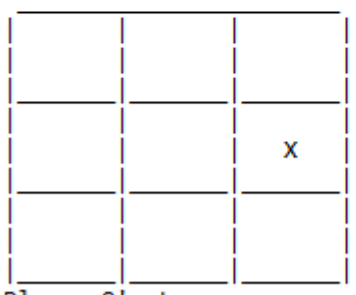
- Die Klasse `TicTacToeMain` übernehmen sie unverändert
- In der Klasse `TicTacToeConsole` müssen sie die `play` Methode sowie den Konstruktor realisieren. Zur Ausgabe auf Console sind bereits vordefinierte Methoden vorhanden.
- In der Klasse `TicTacToe` müssen Sie alle vorgegeben Methoden implementieren.

Verändern Sie die Signaturen der Methoden dabei nicht.

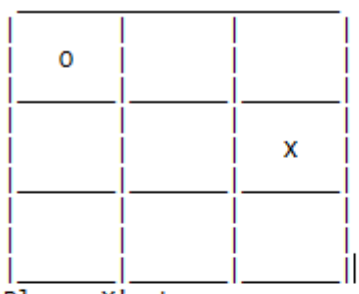
Überlegen Sie sich auch die Vor und Nachteile des Designs.
Die folgende Seite zeigt beispielhaft einen Ablauf auf der Console:



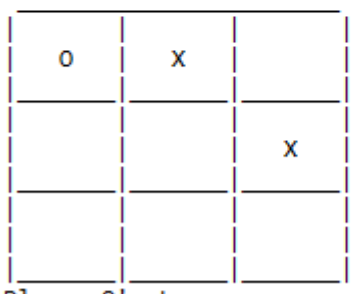
```
Player X's turn.
Player X: Enter row ( 0, 1 or 2 ): 1
Player X: Enter column ( 0, 1 or 2 ): 2
```



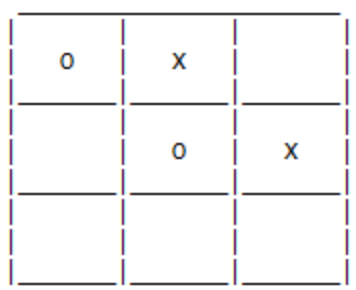
```
Player O's turn.
Player O: Enter row ( 0, 1 or 2 ): 0
Player O: Enter column ( 0, 1 or 2 ): 0
```



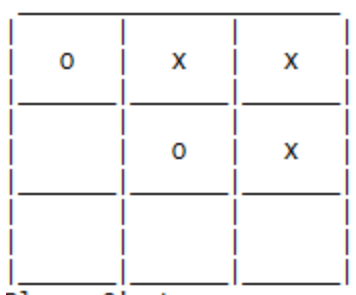
```
Player X's turn.
Player X: Enter row ( 0, 1 or 2 ): 0
Player X: Enter column ( 0, 1 or 2 ): 1
```



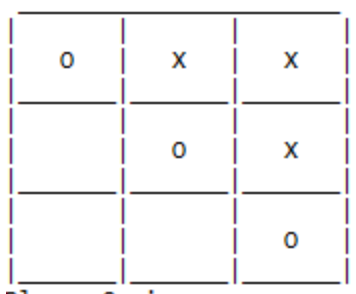
```
Player O's turn.
Player O: Enter row ( 0, 1 or 2 ): 1
Player O: Enter column ( 0, 1 or 2 ): 1
```



```
Player X's turn.
Player X: Enter row ( 0, 1 or 2 ): 0
Player X: Enter column ( 0, 1 or 2 ): 2
```



```
Player O's turn.
Player O: Enter row ( 0, 1 or 2 ): 2
Player O: Enter column ( 0, 1 or 2 ): 2
```



```
Player O wins.
```