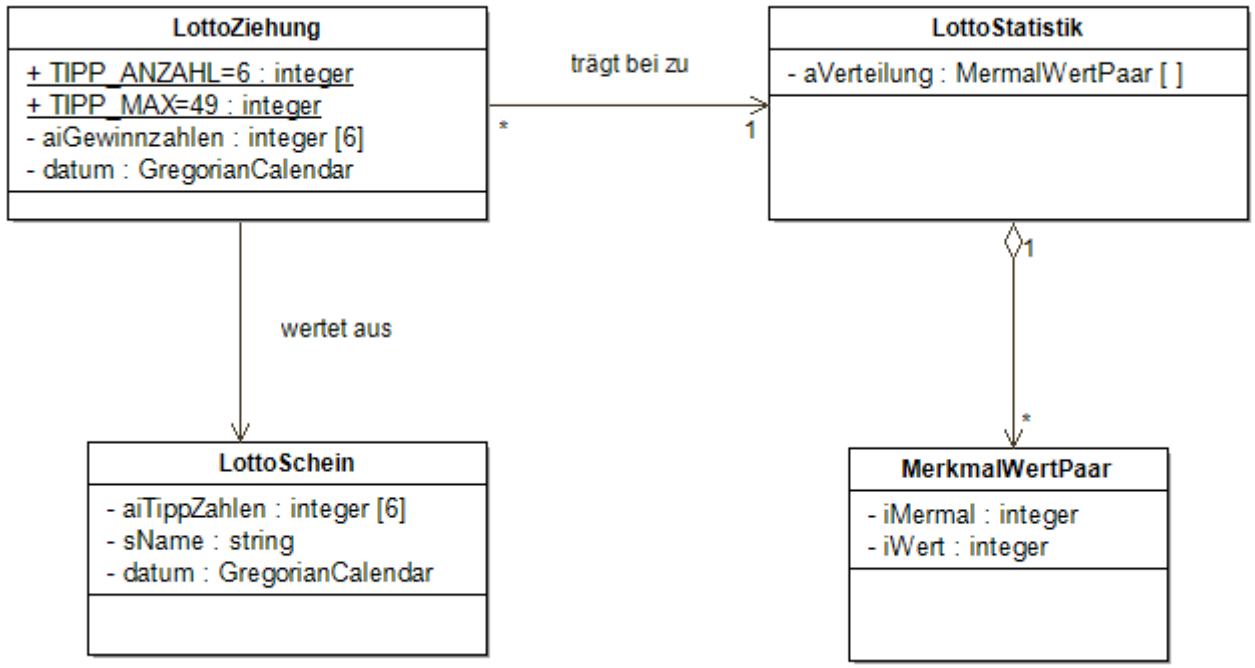
2.6. Simulation Lotto-Tipp „objektorientiert“

Die Lotto Simulation soll jetzt objektorientiert implementiert werden. Das folgende Klassendiagramm zeigt die erforderlichen Fachklassen.



- Die Beschreibung der notwendigen Objektmethoden findet sich in Moodle. (`DatenLottoSimulation` → entpacken)Dort in LottoDok die Datei index.html doppelklicken.
- Die Startklasse LottoStart kann ebenfalls von Moodle (`DatenLottoSimulation`) kopiert und in das Package der Anwendung eingebunden werden.

```
public class LottoSchein
{
    // Anfang Attribute
    private int[] aiTippZahlen;
    private String sName;
    private GregorianCalendar datum;
    // Ende Attribute
    // Anfang Methoden
    /**
     *        Initialisiert sName mit "-leer-";
     *                      datum mit aktuellem Datum;
     *                      erzeugt das int-Array für die Tippzahlen
     * @param iTippAnzahl Anzahl der Zahlen (6 bei 6 aus 49)
     */
    public LottoSchein(int iTippAnzahl)
    {
        this.aiTippZahlen = new int[iTippAnzahl];
        this.sName = "-leer-";
        this.datum = new GregorianCalendar();
    }

     * @return Die Tipp-Zahlen im int-Array (sortiert)
    public int[] getTippZahlen()

     * Tipp-Zahlen können im int-Array übergeben werden
    public void setTippZahlen(int[] aiTippZahlen)

    public String getSName()

    public void setSName(String sName)

    public GregorianCalendar getDatum()

    public void setDatum(GregorianCalendar datum)

    /**
     *
     * Ein Lotto-Tipp wird zufällig ermittelt und gespeichert;
     * Zahlen dürfen nicht doppelt vorkommen;
     * die Zahlen werden sortiert abgespeichert.
     */
    public void generiereTipp()
    {
        int iTipp;
        Random zufall = new Random();
        int iZahl = 0;

        while (iZahl < this.aiTippZahlen.length)
        {
            iTipp = zufall.nextInt(LottoZiehung.TIPP_MAX) + 1;
            if (ArrayTools.sucheSequenziell(this.aiTippZahlen, 0, iZahl - 1, iTipp) == -1)
            {
                this.aiTippZahlen[iZahl] = iTipp;
                iZahl++;
            }
        }
        Arrays.sort(this.aiTippZahlen);
    }
    public String toString()
    {
        StringBuilder sbAlles = new StringBuilder("Tipp von:");

        sbAlles.append(this.sName);
        sbAlles.append("   ");
        sbAlles.append(Arrays.toString(this.aiTippZahlen));

        return sbAlles.toString();
    }
    // Ende Methoden
}
```

```java
public class LottoZiehung
{
    // Anfang Attribute
    public static final  int TIPP_ANZAHL = 6;
    public static final  int TIPP_MAX = 49;
    private GregorianCalendar datum;
    private int[] aiGewinnzahlen;
    // Ende Attribute
    // Anfang Methoden
    /**
     * Erzeugt das int-Array für die Gewinnzahlen;
     * setzt das Datum auf das aktuelle Datum;
     * ermittelt zufällige Gewinnzahlen; Zahlen dürfen nicht doppelt vorkommen;
     * Gewinnzahlen werden sortiert abgespeichert.
     */
    public LottoZiehung()
    {
        int iTipp;
        int iZahl = 0;
        aiGewinnzahlen = new int[TIPP_ANZAHL];
        Random zufall = new Random();

        this.datum = new GregorianCalendar();

        while (iZahl < this.aiGewinnzahlen.length)
        {
            iTipp = zufall.nextInt(TIPP_MAX) + 1;
            if (ArrayTools.sucheSequenziell(this.aiGewinnzahlen, 0, iZahl - 1, iTipp) == -1)
            {
                this.aiGewinnzahlen[iZahl] = iTipp;
                iZahl++;
            }
        }
        Arrays.sort(this.aiGewinnzahlen);
    }

    public GregorianCalendar getDatum()

    public void setDatum(GregorianCalendar datum)

    public int[] getAiGewinnzahlen()

    public void setAiGewinnzahlen(int[] aiGewinnzahlen)

    public int ermittleTreffer(LottoSchein einLottoSchein)
    {
        int iRichtige = 0;

        for (int iTipp : einLottoSchein.getTippZahlen())
        {
            if (Arrays.binarySearch(this.aiGewinnzahlen, iTipp) >= 0)
              iRichtige++;
        }
        return iRichtige;
    }

    /* (non-Javadoc)
     * Eine Lotto-Ziehung wird in folgender Form als String
    public String toString()
    {
        StringBuilder sbAlles = new StringBuilder("Ziehung vom : ");

        sbAlles.append(
                String.format("%02d.%02d.%4d  ",
                              datum.get(GregorianCalendar.DAY_OF_MONTH),
                              datum.get(GregorianCalendar.MONTH)+1,
                              datum.get(GregorianCalendar.YEAR)));


        sbAlles.append(Arrays.toString(this.aiGewinnzahlen));

        return sbAlles.toString();
    }
    // Ende Methoden
}
```

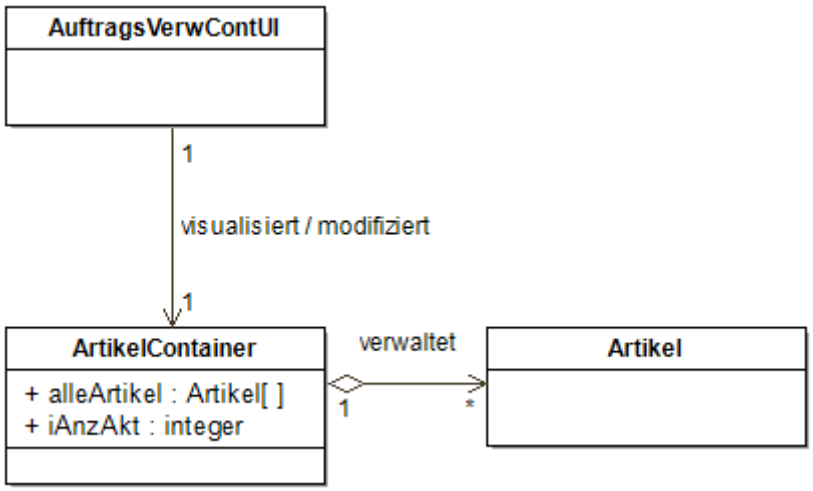
```java
public class MerkmalWertPaar implements Comparable<MerkmalWertPaar>
{
    private int iMerkmal;
    private int iWert;

     * @param iMerkmal
    public MerkmalWertPaar(int iMerkmal, int iWert)
    {
        this.iMerkmal = iMerkmal;
        this.iWert = iWert;
    }
    public int getiMerkmal()
    public void setiMerkmal(int iMerkmal)
    public int getiWert()
    public void setiWert(int iWert)
    public void inkrementWert()
    {
        this.iWert++;
    }

    /* (non-Javadoc)
    /**
     * compareTo Methode definiert die Ordnung der MerkmalWertPaare
     * siehe 20_OOP_Arrays - Sortierung
     */
    @Override
    public int compareTo(MerkmalWertPaar anderesPaar)
    {  // Das erste Minuszeichen sorgt für absteigende Sortierung
       return (-(this.getiWert() - anderesPaar.getiWert()));
    }

}
```

```java
public class LottoStatistik
{
    MerkmalWertPaar[] aVerteilung;

     * Erzeugt ein MerkmalWertPaar Array mit Länge iAnzahlWerte zu Speichern
    public LottoStatistik(int iAnzahlWerte)
    {
        this.aVerteilung = new MerkmalWertPaar[iAnzahlWerte];
        // Initialisiere die Verteilung mit Werten und der Häufigkeit 0
        for (int i = 0; i < this.aVerteilung.length; i++)
        {
            this.aVerteilung[i] = new MerkmalWertPaar(i+1, 0);
        }
    }

    public MerkmalWertPaar[] getaVerteilung()
    {
        return this.aVerteilung;
    }

     * Die im Parameter gegebene LottoZiehung wird der Häufigkeitsverteilung h
    public void addiereZiehung(LottoZiehung eineZiehung)
    {
        int[] aiListe = eineZiehung.getAiGewinnzahlen();
        for (int i : aiListe)
        {
            if (i <= this.aVerteilung.length)
            {
                this.aVerteilung[i-1].inkrementWert();
            }
        }
    }

     * Eine Kopie der gespeicherten Häufigkeitsverteilung wird sortiert (entsp
    public MerkmalWertPaar[] getNachHäufigkeit()
    {
        MerkmalWertPaar[] aSortiert;

        aSortiert = Arrays.copyOf(aVerteilung, aVerteilung.length);
        Arrays.sort(aSortiert);
        return aSortiert;
    }
}
```

3 Containerklassen und 3-Schichten-Architektur

Ziel: Entsprechend der 3-Schicht-Architektur soll eine Containerklasse zum Verwalten von Artikel Objekten erstellt werden.

3.1. Gegeben ist die bereits früher erstelle Fachkonzeptklasse Artikel und eine einfache UI-/Startklasse. Die Beschreibung der Methoden liegt in Form JavaDoc vor. (Siehe `Moodle: DatenArtikelContainer`)



Kopieren Sie sich die Klassen `Artikel` und `AuftragsVerwContUI` in eine neu erstelltes Package. Implementieren Sie die Klasse ArtikelContainer und testen Sie die Methoden. Gehen Sie schrittweise vor indem Sie Teile von `AuftragsVerwContUI` auskommentieren.

```java
public class ArtikelContainer
{
    private Artikel[] alleArtikel;
    private int iAnzAkt;

     * Konstruktor, erzeugt Array für max 20 Artikel (wird dynamisch verlängert)
    public ArtikelContainer()
    {
//      this.alleArtikel = new Artikel[20];
//      iAnzAkt = 0;
        this(20); // Konstruktor-Verkettung
    }

     * Konstruktor, erzeugt Array für iAnzArtikel Artikel
    public ArtikelContainer(int iMaxArtikel)
    {
        this.alleArtikel = new Artikel[iMaxArtikel];
        iAnzAkt = 0;
    }

     * @return liefert aktuelle Anzahl gespeicherter Artikel
    public int getIAnzAkt()
    {
        return this.iAnzAkt;
    }

     * neuerArtikel wird in Array gespeichert und die Anzahl aktualisiert
    public void speichereArtikel(Artikel neuerArtikel)
    {
        if (this.iAnzAkt == this.alleArtikel.length) // wenn Array voll
        {   // Verlängern auf doppelte Länge
            this.alleArtikel = Arrays.copyOf(this.alleArtikel, alleArtikel.length * 2);
        }
        this.alleArtikel[iAnzAkt] = neuerArtikel;
        iAnzAkt++;
    }

     * Sucht den Artikel der die gegebene iArtikelNr besitzt;
    public Artikel sucheArtikelNachNr(int iArtikelNr)
    {
        Artikel gesucht = null;
        int iInd = 0;

        while (iInd < this.iAnzAkt && this.alleArtikel[iInd].getINr() != iArtikelNr)
        {
            iInd++;
        }

        if (iInd < this.iAnzAkt) // ArtikelNr wurde gefunden
        {
            gesucht = this.alleArtikel[iInd];
        }

        return gesucht; // null, wenn nicht gefunden

    }
```

```
     * sucht alle Artikel mit sBezeichnung == sSuchBezeichnung
    public Artikel[] sucheArtikelNachBezeichnung(String sSuchBezeichnung)
    {
        Artikel[] gefundeneArtikel = null;
        int iAnz = 0;
        for (int i = 0; i < this.iAnzAkt; i++)
        {
            if (this.alleArtikel[i].getSBezeichnung().equals(sSuchBezeichnung))
            {
                if (gefundeneArtikel == null) gefundeneArtikel = new Artikel[1];
                if (iAnz == gefundeneArtikel.length) // gilt ab dem 2. gefundenen Artikel
                    gefundeneArtikel = Arrays.copyOf(gefundeneArtikel,
                                                     gefundeneArtikel.length + 1);
                gefundeneArtikel[iAnz] = this.alleArtikel[i];
                iAnz++;
            }
        }
        return gefundeneArtikel; // null, wenn nichts gefunden
    }
}
```

4 Zwei-dimensionale-Arrays

Ziel: Der Umgang mit zwei-dimensionalen Arrays soll am Beispiel eines Memory-Spiels geübt werden.

4.1. Erstellen Sie ein neues Package `memorySpiel`. Laden Sie von Moodle die Datei `DatenMemorySpiel` herunter und entpacken Sie die Dateien in den Ordner des Packages. Nach einem „Refresh“ sollten die Klassen sichtbar sein.

Ergänzt werden muss nur in der Klasse `Memory` der Inhalt des Konstruktors, entsprechend der dort angegebenen Kommentare.
Das Array `aMaske` hat folgenden Inhalt: Das Array `aVorlage` z.B. folgenden:
(jeweils ohne Ränder):

```
 -1-2-3-4-5-6-7-8-
1 #|#|#|#|#|#|#|#|
2 #|#|#|#|#|#|#|#|
3 #|#|#|#|#|#|#|#|
4 #|#|#|#|#|#|#|#|
5 #|#|#|#|#|#|#|#|
6 #|#|#|#|#|#|#|#|
7 #|#|#|#|#|#|#|#|
8 #|#|#|#|#|#|#|#|
-----------------
```

```
 -1-2-3-4-5-6-7-8-
1 2|B|9|8|2|1|)|-|
2 .|=|>|;|C|&|,|@|
3 5|?|/|>|+|'|;|0|
4 6|,|'|4|@|A|8|(|
5 %|3|*|)|$|0|5|(|
6 7|%|6|&|.|<|?|B|
7 C|A|/|9|:|1|=|:|
8 4|7|*|-|<|3|$|+|
-----------------
```

```
public Memory(int iAnzPaare)
{
    Random rd = new Random();
    int iZeile, iSpalte, iZufallZeile, iZufallSpalte;
    int iLaenge;
    char cZeichen;

    // iAnzPaare in Attribut speichern
    this.iAnzPaare = iAnzPaare;
    // Seitenlänge des quadratischen Spielfeldes bestimmen
    iLaenge = (int)Math.sqrt(iAnzPaare * 2);
    // 2-dim. Array für die gesuchten Paare erzeugen: acVorlage (siehe Attribute)
    acVorlage = new char[iLaenge][iLaenge];
    // 2-dim. Array für die bereits aufgedeckten Paare erzeugen: acMaske (siehe Attribute)
    acMaske = new char[iLaenge][iLaenge];

    // acMaske mit '#' füllen -> Startzustand, alle Felder sind "abgedeckt"
    // acVorlage mit zufälligen Paaren auf zufälligen Plätzen belegen
    //           Für die Paare können Zeichen aus dem Unicode Bereich ab dem
    //           '$' Zeichen nacheinander verwendet werden (oder andere nach Wahl)
    // Die Koordinaten für die Platzierung müssen zufällig in dem entsprechenden
    // Wertebereich erzeugt werden, und es muss geprüft werden, ob der zufällig
    // ermittelte Platz nichtschon belegt ist, sonst müssen neue Koordinaten gewählt werden.

    cZeichen = '$'; // Startzeichen '$'
    for (iZeile = 0; iZeile < iLaenge; iZeile++)
    {
        for (iSpalte = 0; iSpalte < iLaenge; iSpalte++)
        {
            this.acMaske[iZeile][iSpalte] = '#';

            if ((iZeile * iLaenge + iSpalte) < this.iAnzPaare)
            {
                for (int i = 1; i <= 2; i++) // Paar plazieren
                {
                    do  // Plazierung wiederholen, falls Feld belegt
                    {
                        iZufallZeile = rd.nextInt(iLaenge);
                        iZufallSpalte = rd.nextInt(iLaenge);
                    } while (this.acVorlage[iZufallZeile][iZufallSpalte] != 0);

                    this.acVorlage[iZufallZeile][iZufallSpalte] = cZeichen;
                }
                cZeichen++; // nächstes Zeichen in Unicode Tabelle
            }
        }
    }
}
```