GK 11 IF 1 / Ziemke	Informatikklausur Nr. 2	20. April 2010
	Name:	
Themen:	Objektorientierte Modellierung, UML, Beziehungen zwischen Klassen und Objekten	
Erl. Mittel:	keine	
Arbeitszeit:	2 Unterrichtsstunden	

1. Aufgabe: ***Beziehungen zwischen Klassen und Objekten***

Für diese Aufgabe gilt das in **Anlage 1** abgebildete UML-Klassendiagramm Version 3.

a) Skizzieren Sie im Heft das Klassendiagramm (Klassen ohne Attribute und Methoden) und ergänzen Sie die fehlenden Darstellungen von Beziehungen.
Notieren Sie zu jeder Beziehung auch:
'Die Klasse K1 erbt von K2.' bzw. 'Objekte von K1 kennen K2-Objekte.' bzw. 'Objekte von K1 haben Objekte von K2.'

b) Begründen Sie, weshalb zwischen Lottofeld6 und Lottozahl keine Hat-Beziehung besteht.

c) Notieren Sie in der Klausur mit den nötigen Ergänzungen […] folgende Aussagen:
(c1) […] und […] bezeichnet man auch als Hat-Beziehungen.
(c2) Spezialisierte Klassen […] alle Attribute und […] ihrer Oberklasse.
(c3) Die […] ist eine existentielle […]-Beziehung; ohne das Objekt kann das Besitzer-Objekt nicht existieren.

d) Bewerten Sie die folgenden Aussagen hinsichtlich ihrer Korrektheit.
Notieren Sie dazu in der Klausur z. B.: zu d1) wahr / zu d2) falsch
Hinweis: Komposit und Aggregat sind in der Beziehung die jeweiligen Besitzer.
(d1) Objekte von Lotto sind Komposit von Tipp-Objekten.
(d2) Lottofeld6 ist die Generalisierung der Tipp-Klasse.
(d3) Ziehung ist eine Spezialisierung der Lottofeld6-Klasse.
(d4) Das ConLotto-Objekt hat ein Lotto-Objekt.
(d5) Objekte von Lottofeld6 haben ein Lottozahl-Objekt.
(d6) Assoziationen nennt man auch Kennt-Beziehungen; sie können gerichtet sein.

2. Aufgabe: ***Modellierung und Implementierung einer Methode***

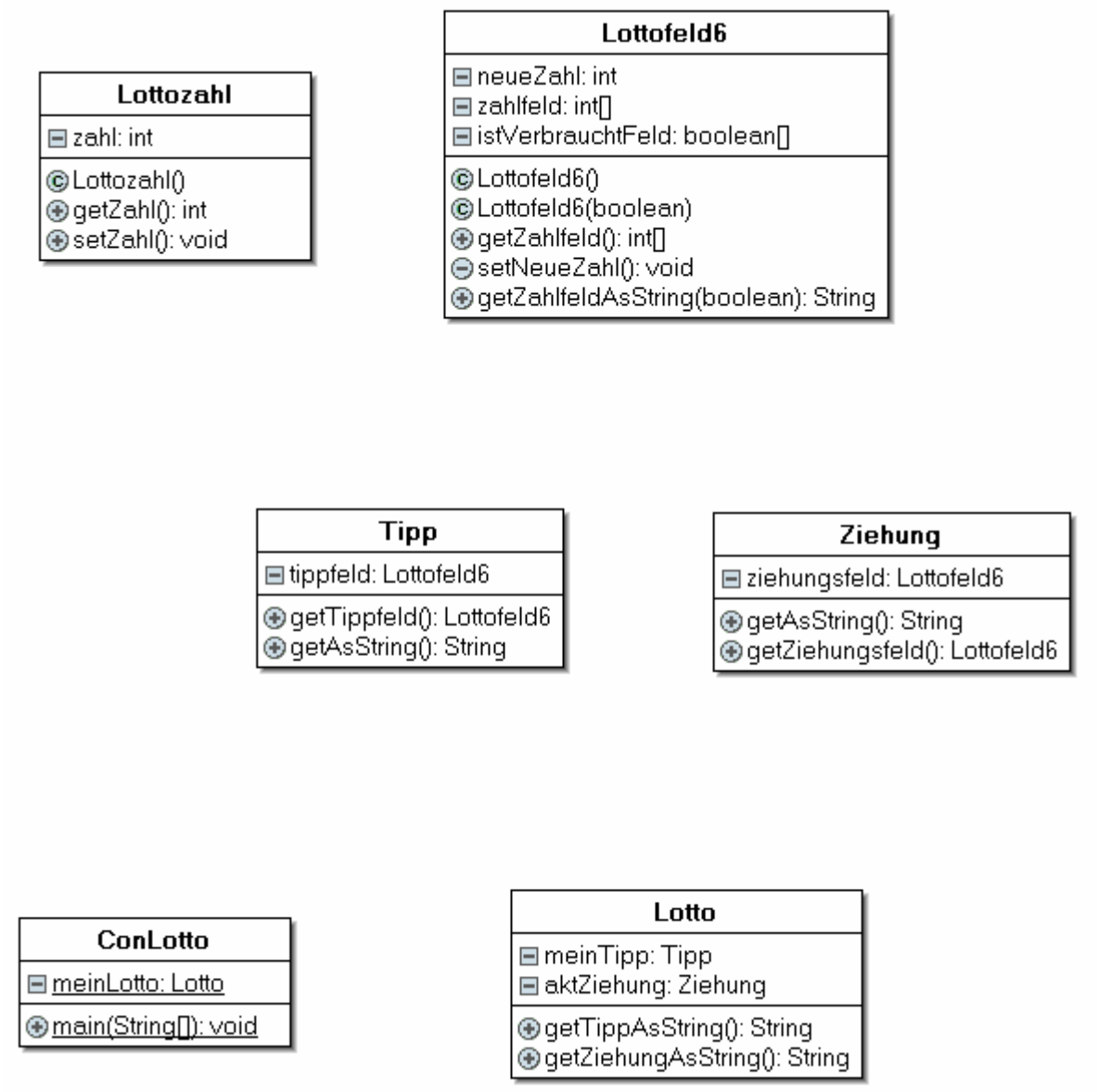
a) In **Anlage 2** ist in Teilblöcken der Grob-Algorithmus einer Methode notiert, die sieben Ziehungszahlen mit den sechs Tippzahlen vergleicht und ein Ergebnis zurückgibt. Die Teilblöcke sind mit Buchstaben (A-E) markiert. Notieren Sie in der Klausur (z. B. als EDCBA) die korrekte Reihenfolge der Teilblöcke. Geben Sie auch Alternativen an, falls es weitere sinnvolle Reihenfolgen der Teilblöcke gibt.

b) Erläutern Sie, welche Bedeutung die Rückgabe des Wertes 8 hat.
Als Tipp sei 7,13,15,22,23,30 und als Ziehung 5,13,20,22,30,32 mit Zusatzzahl 23 bekannt. Notieren Sie den Wert der Rückgabe nach Beenden des Vergleiches.

c) Formulieren Sie nun in Java-Syntax diese Methode *vergleicheTippMitZiehung()*. Sie können – müssen aber nicht – den Methodenrumpf aus **Anlage 3** ergänzen.
Hinweis:
Für die (später mehrfach) neue Ziehung wird übrigens bereits hier ein vereinfachtes Klassenmodell verwendet, das Sie (nur zur Information) in **Anlage 4** finden.
So wird die IntArray-Übergabe z. B. durch *aktZiehung.getZahlfeld()* möglich und Probleme beim Neu-Initialisieren des Ziehungsfeldes entfallen.
Für die hier erforderliche Ergänzung des Methodenrumpfes ist dies aber unerheblich.

Viel Erfolg bei der Bearbeitung!

Anlage 1: UML-Klassendiagramm v3



Anlage 2: Grob-Algorithmus zur Methode vergleicheTippMitZiehung() der Klasse Lotto

A	Hole die Ziehungszahlen und speichere sie als Array tmpZieh ab. Für jedes Element aus tmpZieh ab Position 1 wiederhole Lese den Wert aus und weise ihn der Variablen zahl zu Falls das Element in hilfTipp an Position zahl den Wert true hat, dann erhöhe den Wert von ergebnis um 1.
B	Erzeuge ein Array hilfTipp der Dimension 50. Initialisiere jedes Elemente von hilfTipp mit dem Wert false. Deklariere eine Variable ergebnis und initialisiere sie mit dem Wert 0.
C	Gebe den Wert von ergebnis zurück.
D	Lese den Wert des Elementes in tmpZieh an Position 0 aus und speichere ihn in zahl. Falls das Element in hilfTipp an Position zahl den Wert true hat, dann erhöhe den Wert von ergebnis um 7.
E	Hole die Tippzahlen und speichere Sie als Array tmpTipp ab. Für jedes Element aus tmpTipp wiederhole Lese den Wert aus und weise ihn der Variable zahl zu Weise dem Element von hilfTipp an Position zahl den Wert true zu.

Anlage 3: Rumpf einer Methode vergleicheTippMitZiehung() der Klasse Lotto

```
public int vergleicheTippMitZiehung () {
  boolean[] hilfTipp = new boolean[50];
  int[] tmpTipp = meinTipp.getZahlfeld();
  int[] tmpZieh = aktZiehung.getZahlfeld();
  int i;
  int ergebnis = 0;
  int zahl;
  // Initialisieren von hilfTipp:
  /* HIER ERGÄNZEN */

  // Vergleichen der 6 Ziehungszahlen:
  /* HIER ERGÄNZEN */

  // Vergleich auch der Zusatzzahl:
  /* HIER ERGÄNZEN */

  // Rückgabe:
  return ergebnis;
}
```

Anlage 4: UML-Klassendiagramm v4