Beuth Hochschule für Technik Berlin
Prof. Dr. Simone Strippgen

Klausur

Programmierung I – Konzepte

Di. 20.09.2011

Name: ..... Vorname: .....

Matr.-Nr.: ..... letzter Versuch: ❑ Ja ❑ Nein

Bitte zuerst den Kopf dieses Blattes vollständig und leserlich ausfüllen. Lassen Sie bei der Bearbeitung der Aufgaben die Blätter zusammen geheftet. Bitte schreiben Sie die Lösungen unter den jeweiligen Aufgabentext. Benutzen Sie kein eigenes Papier (außer zum Vorschreiben).

Viel Erfolg!

Aufgabe	**Mögliche Punkte**	**Erstkorrektur**	**Zweitkorrektur**
Aufgabe 1 : Aufwärmen	11		
Aufgabe 2: Sammlungen, Schleifen	16		
Aufgabe 3: HashMap	8		
Aufgabe 4: Klassenkonstanten/ -methoden	20		
Aufgabe 5: Vererbung, Überschreiben von Methoden	24		
Aufgabe 6: ArrayList, Substitution	5		
Aufgabe 7: Interfaces	16		
Punkte Gesamt	**100**		
Note			

Aufgabe 1: Aufwärmen

Die folgenden Zuweisungen sind zum Teil falsch. Umkreisen Sie die 6 Zuweisungen, die der Compiler nicht erlauben würde.

1.) int x = 34;

2.) short s;

3.) int d = 2 * 2.5f;

4.) int y = x;

5.) y = y + 2.5;

6.) double v = x;

7.) boolean boo = x;

8.) float b = 3.4;

9.) v = 2.3f;

10.) int g = 17.3;

11.) s = y;

12.) short n = 12;

Umkreisen Sie in den folgenden logischen Ausdrücken die zwei Ausdrücke die den Wert *true* ergeben.

1.) 3 > 0 && 12 < 10

2.) 3 < 10 && 5 > 9 || 2 == 1 + 1

3.) !(3 > 0) || 5 > 10

4.) 3 >= 5-2 ^ 5 < 10

5.) !(3+4>=7) || !(11 <=10)

6.) 3 >=10 && (5 > 9 || 2 == 1 + 1)

Schreiben Sie eine Anweisung, die die Zeichenkette in der Variablen *wort* so zerlegt, dass das folgende Array dabei herauskommt: { "langes", "langes", "langes", "wort" }. Weisen Sie dieses Rückgabe-Array der Variablen *wortArray* vom Typ Array von *String* zu. Nutzen Sie dazu geeignete Methoden der Klasse String.

```
String wort  =  "Langes--Langes--Langes--Wort";
String[] wortArray;
```

Aufgabe 2: Sammlungen, Schleifen

Schreiben Sie Import-Anweisungen, um im Quellcode die Klasse *ArrayList* und die Klasse *Iterator* benutzen zu können.

Es sei die folgende Klasse gegeben:

```
public class Hund {

   private String name;

   public Hund(String name) {...}

   public String gibName() {
      return name;
   }

   public String toString() {...}
}
```

Ergänzen Sie die Klasse *Hundepension* um die Deklaration eines Attributs *gaeste* vom Typ *ArrayList*, das Objekte vom Typ *Hund* aufnehmen kann. Ergänzen Sie den Konstruktor um die entsprechenden Zuweisungen. Schreiben Sie die Rümpfe der beiden Methoden. Nutzen Sie in der Methode *auschecken(..)* die Klasse *Iterator*.

```
public class Hundepension
{
  ____________________________________________________________________

  private String name;

  public Hundepension(String name)
  {
   ____________________________________________________________________

   ____________________________________________________________________
  }
```

```
/**
 * Fügt das übergebene Hund-Objekt in die Sammlung ein, wenn ein Objekt
 * übergeben wurde und der Name des Hundes mindestens ein Zeichen beinhaltet.
 * @param  neuerGast   das einzufügende Hund-Objekt
 */
 public void einchecken(Hund neuerGast)
 {









 }

/**
 * Entfernt das zuerst gefundene Hund-Objekt mit dem Namen name
 * aus der Sammlung und liefert es zurueck.
 * @param  name   name des zu entfernenden Hund-Objekts
 * @return  ausgeschecktes Hund-Objekt, null sonst
 * Hier soll ein Iterator benutzt werden.
 */
 public Hund auschecken(String name) {









 }
```

Aufgabe 3: HashMap

Deklarieren und Initialisieren Sie eine Variable *hunde* vom Typ *HashMap* in der man Objekte vom Typ *Hund* unter dem Namen des *Hund*-Objekts abspeichern kann.

Schreiben Sie eine Methode *einfuegen(..)*, die ein *Hund*-Objekt übergeben bekommt und das *Hund*-Objekt in die *HashMap hunde* einfügt. Sorgen Sie dafür, dass es keine *NullPointerException* gibt.

Schreiben Sie eine Methode *gibHund(..)*, die als Argument einen Namen bekommt und als Rückgabewert das Hund-Objekt mit dem angegebenen Namen aus der HashMap *hunde* zurück liefert.

Aufgabe 4: Klassenkonstanten/-methoden, Switch-Anweisung

Schreiben Sie eine Klasse *Player*, die zwei Audio-Dateiformate abspielen kann: AAC und MP3. Die Dateiformate sollen in der Klasse als öffentliche Klassenkonstanten definiert sein.

Die Klasse soll eine öffentliche Klassenmethode

```
spiele(int format, String datei)
```

besitzen, der das Format (unter Angabe einer der Klassenkonstanten) und der Pfad der Datei übergeben wird. In der Methode *spiele* soll per switch-Anweisung - je nach Dateiformat - eine spezielle private Klassenmethode aufgerufen werden (z.B. *spieleAAC(..)*), die das Abspielen der Datei bewirkt.

Definieren Sie auch diese Klassenmethoden, deren Rumpf kann aber leer bleiben, d.h. `{...}`. Sollte die Methode *spiele(..)* ein unbekanntes Dateiformat übergeben bekommen, wird die Fehlermeldung „Unbekanntes Format“ auf der Konsole ausgegeben. Die Fehlermeldung soll als private Konstante der Klasse angelegt werden.

Notieren Sie einen externen Methodenaufruf der Methode *spiele(..)* für das AAC-Dokument in *song*.

```
String song = "mySong.aac";
```

Aufgabe 5: Vererbung / Überschreiben von Methoden

Gegeben seien folgende Klassen *Katze* und *Fisch*. Modellieren Sie eine Klassenhierarchie, die die Gemeinsamkeiten der beiden Tiere so abbildet, dass sich leicht weitere Tierarten einfügen lassen. Versuchen Sie dazu, möglichst viele allgemeine Eigenschaften und Verhaltensweisen bereits in der gemeinsamen Superklasse anzubieten.

Schreiben Sie die drei Klassen auf und definieren Sie sowohl die Konstruktoren (deren Rümpfe hier ausgelassen wurden) als auch die entsprechenden Methoden. Die Methode *toString()* sollte so ausformuliert werden, dass sie eine Beschreibung des Objekts und dessen Eigenschaften als Zeichenkette zurück gibt. Sie müssen keine getter- und setter-Methoden für die Attributwerte hinzufügen. Überlegen Sie, ob es abstrakte Methoden gibt.

Klassen auf der nächsten Seite.

```
public class Katze
{
    private String typ;
    private boolean wild;  // Wild- oder Hauskatze
    private String fresschen;
    private boolean langhaarig;
    private int wieAlt;

    public Katze (String bezeichner, boolean wild, int wieAlt,
                  String nahrung, boolean langh)
    { ... }

    public void nahrungAufnehmen() {
        System.out.println("Ich fresse " + fresschen);
    }

    public void geraeuschMachen()
    {
        if(wild) System.out.println("Roaaar, roaaar!");
            else System.out.println("Miau, miau!");
    }

    public String toString() {
        // Rückgabe von String mit Beschreibung des Objekts
    }
}


public class Fisch
{
    private String art;
    private String wasserArt; //Salz- oder Süßwasser
    private String futter;
    private float wasserTemperatur;
    private int alter;

    public Fisch(String futter, String art, float wTemp, int dasAlter,
                 String wArt)
    {...}

    public void fressen() {
        System.out.println("Ich fresse " + futter);
    }

    public void blubbern()
    {
        System.out.println("Blubb, blubb!");
    }

    public String toString() {
        // Rückgabe von String mit Beschreibung des Objekts
    }
}
```

Hier kommt die Superklasse hin:

Hier kommt die Klasse *Katze* hin:

Hier kommt die Klasse *Fisch* hin:

Aufgabe 6: ArrayList, Substitution

Deklarieren und Initialisieren Sie eine Variable *tiere*, mit einer *ArrayList, die* beliebige Tiere (aus Aufgabe 5) aufnehmen kann.

Schreiben Sie eine Methode *tiereAusgeben()*, die in einer For-Each-Schleife für jedes Tier in der Variablen *tiere* dessen Eigenschaften auf der Konsole ausgibt und bei allen Katzen zusätzlich das Geräusch der Katze ausgibt.

Aufgabe 7: Interfaces

Unten sehen Sie die Klasse *Drucker*. Diese besitzt eine einzige statische Methode *drucke(..)*. Die Methode kann alle Objekte drucken, die das Interface *Druckbar* implementieren.

```
public class Drucker
{

  public static void drucke(Druckbar datei) {
    int typ = datei.gibTyp();
    String pfad = datei.gibDateipfad();

    if(pfad.equals("")) {
        System.out.println("PFAD_ERROR ");
        return;
    }
    if(typ == Druckbar.PDF) druckePDF(pfad);
     else if(typ == Druckbar.DOC) druckeDOC(pfad);
        else System.out.println("TYP_ERROR ");
  }
  ...
}
```

Schreiben Sie das Interface *Druckbar*, das garantiert, dass die Objekte aller Klassen, die das Interface implementieren, von *Drucker* gedruckt werden können.

Schreiben Sie eine Klasse *Dokument* die das Interface *Druckbar* implementiert.