Aufgabe 1: Aufwärmen

a) Schreiben Sie eine sondierende Methode ausluesen(), die den Wert des Attributs fuellstand (vom Typ float) zurück liefert und das Attribut fuellstand auf 0 setzt.

(5 Punkte)

b) Schreiben Sie eine verändernde Methode aendern, die ein Zeichen übergeben bekommt. Bei '+' soll sie auf das Attribut menge vom Typ int 10 aufaddieren, bei '-' soll sie 10 abziehen, aber nur wenn der Wert von menge dabei nicht unter 0 fällt. Nutzen Sie hierfür eine switch-Anweisung.

(5 Punkte)

c) Dem Konstruktor der Klasse Auto kann bei der Initialisierung eines Objekts der Bezeichner des Fabrikats mitgegeben werden. Auf welchen Wert verweist die Variable auto2 nach den drei Zuweisungen.

(2 Punkte)

```
Auto auto1 = Auto("BMW");
Auto auto2 = Auto("Ferrari");
auto2 = auto1;
auto1 = null;
```

1

b) Schreiben Sie eine verändernde Methode *aendern*, die ein Zeichen übergeben bekommt. Bei '+' soll sie auf das Attribut *menge* vom Typ *int* 10 aufaddieren, bei '-' soll sie 10 abziehen, aber nur wenn der Wert von *menge* dabei nicht unter 0 fällt. Nutzen Sie hierfür eine switch-Anweisung. (5 Punkte)

Aufgabe 2: Sammlungen, Schleifen, Klassenkonstanten, -methoden

a) Ergänzen Sie die Klasse *BesucherZahlen* um ein privates Attribut *woechentlicheWerte*, das die Besucherzahlen für jede Woche eines Jahres in einem *Array* aufnehmen kann. Die Anzahl der Kalenderwochen im Jahr (53) soll als private Klassenkonstante angelegt werden. Ergänzen Sie den Konstruktor um entsprechende Zuweisungen. (5 Punkte)

```
public class BesucherZahlen {
   private String jahr;
```

Aufgabe 2: Sammlungen, Schleifen, Klassenkonstanten, -methoden

a) Ergänzen Sie die Klasse *BesucherZahlen* um ein privates Attribut *woechentlicheWerte* das die Besucherzahlen für jede Woche eines Jahres in einem *Array* aufnehmen kann Die Anzahl der Kalenderwochen im Jahr (53) soll als private Klassenkonstante angeleg werden. Ergänzen Sie den Konstruktor um entsprechende Zuweisungen. (5 Punkte

```
public class BesucherZahlen {
   private String jahr;

   public BesucherZahlen(String jahr)
   {
```

b) Schreiben Sie für die Klasse *BesucherZahlen* eine private Klassenmethode *berechneGesamt*, die ein *Array* mit ganzzahligen Werten übergeben bekommt. Das *Array* enthält für eine beliebige Menge von Tagen die Anzahl der Besucher des Tages. Die Methode soll aus den Werten im *Array* die Summe der darin befindlichen Werte berechnen und diesen Wert als Rückgabewert liefern. (7 Punkte)

Aufgabe 3: HashMap

Es sei die folgende Klasse gegeben:

```
public class Kunde {
   private String name;
   private int nummer;
   ...

   public String gibName() {
      return name;
   }

   public int gibKundenNummer() {
      return nummer;
   }
   ...
}
```

a) Deklarieren und Initialisieren Sie ein privates Attribut *kunden* vom Typ *HashMap* in der man Objekte vom Typ *Kunde* unter der Kundennummer des *Kunde*-Objekts abspeichern kann.

(3 Punkte)

b) Schreiben Sie eine Methode *einfuegen*, die ein *Kunde*-Objekt übergeben bekommt und das *Kunde*-Objekt in die *HashMap kunden* einfügt. Sorgen Sie dafür, dass es keine *NullPointerException* gibt und kein Kunde eingefügt wird, dessen Name ein leerer String ist (kein Zeichen oder nur Leerzeichen).

(6 Punkte)

Aufgabe 4: Klassenattribute/-methoden

Versehen Sie die Klasse *Kunde* mit einem Mechanismus, der es erlaubt, die nächste verfügbare Kundennummer – beginnend mit dem Wert 1 – zu berechnen und diese bei der Erzeugung einer *Kunde*-Instanz dem Attribut *nummer* zuzuweisen. (6 Punkte)

```
public class Kunde {
   private String name;
   private int nummer;
```

5

```
public class Monster {

    private String art;
    private Waffe waffe;
    private int xPos;
    private int yPos;
    private boolean getroffen;


    public Monster(String art, Waffe waffe, in
     ...
    }

    public void bewegen(int x, int y) {
        xPos += x;
        yPos += y;
        neuZeichnen();
    }

    private void neuZeichnen(){
        // Code zum zeichnen
    }

    public void getroffen() {
        getroffen = true;
    }
}
```

Aufgabe 6: Interfaces

Unten sehen Sie die Klasse *Loader*. Diese besitzt die statische Methode *download(..)* und eine private statische Methode zum Laden der Datei (hier nicht dargestellt). Die Methode kann alle Objekte laden, die das Interface *Downloadable* implementieren.

```
public class Loader {

  public static void download(Downloadable datei) {
    if(datei != null) {
      int groesse = datei.gibGroesse();
      String url = datei.gibUrl();
      if(url.equals(""))
        throw new IllegalArgumentException(Downloadable.ERROR_URL);
      if(groesse == 0)
        throw new IllegalArgumentException(Downloadable.ERROR_SIZE);
      load(url);
    }
    else
      throw new IllegalArgumentException("Wert null uebergeben");
  }
  ...
}
```

Schreiben Sie das Interface *Downloadable*, das garantiert, dass die Objekte aller Klassen, die das Interface implementieren, von *Loader* geladen werden können. (6 Punkte)

Aufgabe 6: Interfaces

Unten sehen Sie die Klasse *Loader*. Diese besitzt die statische Methode *download(..)* u
eine private statische Methode zum Laden der Datei (hier nicht dargestellt). D
Methode kann alle Objekte laden, die das Interface *Downloadable* implementieren.

```
public class Loader {

  public static void download(Downloadable datei) {
    if(datei != null) {
      int groesse = datei.gibGroesse();
      String url = datei.gibUrl();
      if(url.equals(""))
        throw new IllegalArgumentException(Downloadable.ERROR_URL);
      if(groesse == 0)
        throw new IllegalArgumentException(Downloadable.ERROR_SIZE);
      load(url);
    }
    else
      throw new IllegalArgumentException("Wert null uebergeben"
    }
  ...
}
```

Schreiben Sie das Interface *Downloadable*, das garantiert, dass die Objekte aller Klas
die das Interface implementieren, von *Loader* geladen werden können. (6 Pun

Aufgabe 7: ArrayList, Substitution

Deklarieren und Initialisieren Sie ein privates Attribut *spielfiguren*, mit einer *ArrayList*, die sowohl Objekte der Klasse *Spieler* als auch *Monster* (aus Aufgabe 5) aufnehmen kann.

(1 Punkt)

Schreiben Sie eine Methode *bewegeSpielfiguren(int x, int y)*, in der mit einem Iterator über alle Objekte in *spielfiguren* iteriert wird. Jede Figur soll um die angegebenen Werte bewegt werden. Wenn es sich bei dem Objekt um ein Monster handelt, so soll vor dem Bewegen geprüft werden, ob das Monster getroffen wurde. Hierfür steht in der Klasse *Monster* die Methode `boolean istGetroffen()` zur Verfügung Ist dies der Fall, so soll das Objekt nicht bewegt, sondern aus der *ArrayList* gelöscht werden. (9 Punkte)