**Information:** Die Beispiele sind bis zur Übung am 7.11. vorzubereiten.

**Beispiel 5.1:**
Gegeben ist die Klasse *Registrierkassa* mit dem Konstruktor

```
Registrierkassa(String typ)
```

und der Methode

```
double getTagesumsatz(),
```

die den Tagesumsatz zurückgibt.

Schreiben Sie eine Unterklasse *RegistrierkassaNeu* der Klasse *Registrierkassa* mit Konstruktor

```
RegistrierkassaNeu(String typ, double steuersatz),
```

der ein entsprechendes Objekt der Klasse *RegistrierkassaNeu* mit vorgegebenem Steuersatz erstellt. Schreiben Sie weiters eine Methode

```
double getFaelligeUmsatzsteuer(),
```

die die fällige Umsatzsteuer (d.i. die im Steuersatz festgelegten % des Tagesumsatzes) zurückgibt.

**Beispiel 5.2:**
Gegeben sei eine Klasse *Person* mit Konstrukor

```
Person(String name),
```

der eine Person mit entsprechendem Namen erzeugt. Die Klasse *Person* verfügt außerdem über eine Methode

```
public String getName(),
```

die den Namen der Person zurückgibt.

Schreiben Sie eine Klasse *Akademiker*, die von *Person* erbt, und über einen Konstruktor

```
Akademiker(String name, String titel)
```

verfügt, der einen Akademiker mit entsprechendem Namen und Titel erzeugt. Schreiben Sie in die Klasse *Akademiker* außerdem eine Methode

```
public void promoviere(),
```

die dem Akademiker einen zusätzlichen Dr.-Titel hinzufügt. Überschreiben Sie die Methode *getName()* so, dass vor dem Namen sämtliche akademischen Titel angeführt werden.

**Beispiel 5.3:**
Gegeben sei eine Klasse *Kamera* mit parameterlosem Konstruktor und einer Methode

```
public ArrayList<Bild> getBilder(),
```

die alle von der Kamera aufgenommenen Bilder in einer ArrayList zurückgibt.

Weiters gebe es eine Klasse *Objekterkennung* mit einer statischen Methode

```
public static boolean zeigenDasselbeObjekt(Bild b1, Bild b2),
```

die genau dann *true* zurückgibt, wenn die Bilder *b1* und *b2* dasselbe Objekt zeigen.

Schreiben Sie eine Klasse *KameraMitObjekterkennung*, die von *Kamera* erbt und über einen parameterlosen Konstruktor verfügt, der eine Kamera für die Erkennung von Bildern erstellt.

Schreiben Sie in die Klasse *KameraMitObjekterkennung* eine Methode

```
public ArrayList<Bild> getBilderWie(Bild dasBild),
```

die alle aufgenommenen Bilder, die dasselbe Objekt wie *dasBild* zeigen, in einer ArrayList zurückgibt.

**Beispiel 5.4:**
Zur Regelung der Einfahrt in ein Parkhaus sind die Klassen *Parkkarte* und *Schrankenanlage* gegeben. Die Klasse *Schrankenanlage* verfügt über einen Konstruktor

```
public Schrankenanlage (String typ),
```

der eine Schrankenanlage vom angegebenen Typ erstellt. Die Klasse *Schrankenanlage* verfügt außerdem über die Methoden

```
public Parkkarte fuerEinfahrtOeffnen(),
public boolean fuerAusfahrtOeffnen(Parkkarte eineKarte)
```

die den Schranken für die Einfahrt öffnen und eine Parkkarte zurückgeben bzw. den Schranken für die Ausfahrt öffnen, falls auf *eineKarte* eine korrekte Zahlung für den Aufenthalt gebucht wurde. In diesem Fall gibt die Methode *true* zurück, andernfalls *false*.

Schreiben Sie eine Klasse *SchrankenanlageMitKapa*, die von *Schrankenanlage* erbt. Der Konstruktor

```
public SchrankenanlageMitKapa(String typ, int kapa)
```

soll eine Schrankenanlage vom angegebenen Typ für ein Parkhaus mit maximaler Auslastung von *kapa* Fahrzeugen erstellen. Überschreiben Sie geeignete Methoden, sodass nach Aufruf von *fuerEinfahrtOeffnen* der Schranken nur dann geöffnet und eine Parkkarte zurückgegeben wird, wenn die maximale Auslastung noch nicht erreicht wurde. Andernfalls soll *null* zurückgegeben werden.
Schreiben Sie außerdem eine Methode

```
public int gibFreiePlaetze(),
```

die die Anzahl der freien Plätze im Parkhaus zurückgibt.

**Beispiel 5.5:**
Gegeben seien die Klassen *Person* und *Projekt*.

Die Klasse *Person* verfügt über einen Konstruktor

```
public Person(String name, int personalNummer),
```

sowie eine Methode

```
public void macheVerantwortlichFuer(Projekt p),
```

die die Person für das Projekt *p* verantwortlich macht.

Schreiben Sie eine Klasse *Verantwortlicher*, die von *Person* erbt und über eine zusätzliche Methode

```
public ArrayList<Projekt> getProjekte()
```

verfügt, die alle Projekte, für die der Mitarbeiter verantwortlich ist, in einer ArrayList zurückgibt.

Überlegen Sie sich auch einen sinnvollen Konstruktor für die Klasse *Verantwortlicher.*

**Beispiel 5.6:**
Gegeben sei eine Klasse *Datei* mit Konstruktor

```
public Datei(String einText),
```

der eine Datei erstellt, die den Text *einText* enthält. Weiters gebe es in der Klasse *Datei* die Methoden

```
public String liesDatei(),
public boolean schreibeDatei(String text),
```

mit denen die Datei ausgelesen bzw. neu geschrieben werden kann. Die Methode *schreibeDatei* gibt dabei *true* zurück, wenn der Schreibvorgang erfolgreich war, sonst *false*.

Schreiben Sie eine Klasse *GeschuetzteDatei*, die von *Datei* erbt. Der Konstruktor

```
public GeschuetzteDatei(String derText)
```

soll eine den Text *derText* enthaltende Datei erstellen, die sowohl für Lese- als auch Schreibzugriff gesperrt ist.

Implementieren Sie außerdem Methoden, mit denen der Lese- und Schreibzugriff neu gesetzt werden kann. Dabei soll eine Datei ohne Lesezugriff automatisch auch schreibgeschützt sein. Überschreiben Sie die Methoden *liesDatei* und *schreibeDatei* so, dass die Datei nur dann gelesen bzw. neu geschrieben werden darf, wenn der Lese- und Schreibzugriff entsprechend gesetzt ist.

**Beispiel 5.7:**
Schreiben Sie eine Klasse *HeizungssteuerungPlus* mit einem Konstruktor Ihrer Wahl, die von *Heizungssteuerung* aus Beispiel 1.6 erbt, und über zusätzliche Methoden

```
public void aufdrehen()
public void abdrehen()
```

verfügen soll, die die Heizungssteuerung auf den Maximal- bzw. Minimalwert setzen.