IT 1 – WS 2019/2020 – Übungsblatt 8

Informationen: Die Beispiele sind bis 28.11., 8 Uhr abzugeben bzw. zur Übung vorzubereiten.

Beispiel 8.1:
Gegeben seien die Klassen *Auftrag* und *Person*. In der Klasse *Person* sind die Methoden *equals* und *hashCode* aus *Object* überschrieben.

Schreiben Sie eine Klasse *Auftragsverwaltung* mit parameterlosem Konstruktor, der eine Auftragsverwaltung ohne Aufträge anlegt.

Schreiben Sie in die Klasse *Auftragsverwaltung* außerdem noch folgende Methoden:

```
public boolean addAuftrag(Auftrag einA, Person eineP)
```

soll den Auftrag *einA* der Person *eineP* zuweisen, falls dieser Auftrag noch nicht einer anderen Person zugewiesen ist. In diesem Fall soll die Methode *true* zurückgeben, andernfalls *false*.

```
public void loescheAuftrag(Auftrag einAuftrag)
```

soll den Auftrag *einAuftrag* aus der Auftragsverwaltung löschen.

```
public Person getBeauftragten(Auftrag einAuftrag)
```

soll jene Person zurückgeben, die dem Auftrag *einAuftrag* zugewiesen ist.

Beispiel 8.2:
Gegeben sei eine Klasse *Person*.

Schreiben Sie in einer Klasse Ihrer Wahl eine statische Methode

```
public static Map<Integer,Person> getMapFromList(List<Person> liste),
```

die die in der Liste enthaltenen Personen in einer Map speichert und zurückgibt, wobei der Index der entsprechende Key sein soll.

Beispiel 8.3:
Gegeben seien die Klassen A*uto* und *Person*, in denen die Methoden *equals* und *hashCode* aus *Object* überschrieben sind.

Schreiben Sie in einer Klasse Ihrer Wahl eine Methode

```
public boolean gibtsPersonenMitMehrAlsEinemAuto(Map<Auto,Person> m),
```

die genau dann *true* zurückgeben soll, wenn in der Map *m* mehreren Autos dieselbe Person zugeordnet ist.

Beispiel 8.4: ***[Lieferanalyse.java]***
Gegeben seien die Klassen *Firma* und *Lieferant*. In der Klasse *Firma* gibt es eine Methode

```
HashMap<Lieferant,Double> getLieferanten(),
```

die in der HashMap für jeden Lieferanten das jeweilige Liefervolumen angibt. Wird die Firma von einem Lieferanten nicht beliefert, ist kein Eintrag in der HashMap vorhanden. Schreiben Sie eine Klasse *Lieferanalyse* mit einer Methode

```
double getUmsatzanteil(Firma firma, Lieferant lieferant),
```

die den prozentuellen Anteil von *lieferant* am gesamten Volumen zurückgibt, das an *firma* geliefert wird.

Beispiel 8.5:
Gegeben sei eine Klasse *Kino* mit Konstruktor

```
public Kino(int n, int m, Map<Integer,Double> diePreise),
```

der ein Kino mit *n* Reihen, *m* Sitzen pro Reihe und Preisen wie in *diePreise* angegeben erstellt. Dabei ist in *diePreise* für jede Reihe ein Preis für einen beliebigen Sitz in dieser Reihe hinterlegt. Die Klasse *Kino* verfügt außerdem über eine Methode

```
public Map<Integer,Double> getPreise(),
```

die die Preise wie oben angegeben in einer Map zurückgibt.

Schreiben Sie eine Klasse *KinoMitErmaessigung* mit einem Konstruktor Ihrer Wahl, die von *Kino* erbt und über eine Methode

```
public Map<Integer,Double> getErmaessigtePreise(double rabatt)
```

verfügt, die um *rabatt* Prozent ermäßigte Preise zurückgibt.

Beispiel 8.6:
Gegeben seien die Klasse *Fahrzeug* und das Interface *PrivateNutzung*:

```
class Fahrzeug
{
      /* Konstruktor, dem die Fahrzeugnummer uebergeben wird.*/
      Fahrzeug(int nummer);

      /* Erhoeht den Kilometerstand um die gefahrenen km. */
      public void fahre(double anzahlKilometer);

      /* Gibt den Kilometerstand zurueck. */
      public double getKilometerstand();
}

interface PrivateNutzung
{
      /* Vermerkt die Laenge einer privaten Fahrt. */
      public void fahrePrivat(double anzahlKilometer);

      /* Liefert die Summe der privat gefahrenen Kilometer. */
      public double getSummePrivatfahrten();
}
```

Weiters sei in Fahrzeug die Methode *toString()* aus Object so überschrieben, dass die Nummer des Fahrzeugs (als String) zurückgegeben wird.

Schreiben Sie eine Klasse *Dienstwagen*, die Unterklasse von *Fahrzeug* ist und auch das Interface *PrivateNutzung* implementiert. Diese Klasse soll über einen Konstruktor

```
Dienstwagen(int nummer, String dienstnehmer)
```

verfügen, dem die Nummer des Fahrzeugs und der Name des Dienstnehmers übergeben wird. Überschreiben Sie in *Dienstwagen* auch die Methode *toString()* so, dass die Nummer des Dienstwagens, der Dienstnehmer, und der Kilometerstand zurückgeliefert werden. Achten Sie darauf, dass die Methode *getKilometerstand()* auch für Dienstwägen korrekt funktioniert.

Beispiel 8.7:
Gegeben sei eine Klasse *Kugel* mit einer Methode

```
String getColor(),
```

die die Farbe der Kugel als String zurückgibt.

Schreiben Sie in einer Klasse Ihrer Wahl eine statische Methode

```
public static List<String> getDifferentColors(List<Kugel> dieKugeln),
```

die die verschiedenen Farben der in *dieKugeln* vorkommenden Kugeln in einer Liste zurückgibt. Dabei soll in der zurückgegebenen Liste jede Farbe höchstens einmal vorkommen.