Informationen: Die Beispiele sind bis zur Übung am 19.12. vorzubereiten.

Beispiel 10.1:
Gegeben seien die Klasse *Uhrzeit*, die das Interface *Comparable* implementiert (dem entsprechend sind natürlich auch *equals* und *hashCode* überschrieben), sowie die Klasse *Zugverbindung* mit der Methode

```
public ArrayList<String> gibStrecke(),
```

die die Haltebahnhöfe der Zugverbindung in einer *ArrayList* zurückgibt.

Schreiben Sie eine Klasse *BahnhofsInfo* mit Konstruktor

```
public BahnhofsInfo (String ort,
            TreeMap<Uhrzeit,Zugverbindung> dieVerbindungen),
```

wobei *ort* der Ort ist, in dem der Bahnhof steht, und in *dieVerbindungen* alle Zugverbindungen des Bahnhofs an einem Tag unter der entsprechenden Abfahrtszeit abgelegt sind. (Sie können davon ausgehen, dass die Strecken dieser Verbindungen nur die Haltebahnhöfe ab *ort* enthalten.)

Schreiben Sie in die Klasse *BahnhofsInfo* außerdem eine Methode

```
public Zugverbindung gibNaechsteVerbindung(Uhrzeit aktZeit,
                                           String zielOrt),
```

die die nächste Zugverbindung nach *zielOrt* zurückgibt, wenn *aktZeit* die aktuelle Uhrzeit ist. Falls es am selben Tag keine entsprechende Verbindung gibt, so soll die Methode *null* zurückgeben.

Beispiel 10.2:
Gegeben sei eine Klasse *Benutzerkonto* mit Konstruktor

```
public Benutzerkonto(String benutzername, String passwort),
```

der ein Benutzerkonto mit entsprechendem Benutzernamen und Passwort anlegt. Die Klasse *Benutzerkonto* verfügt außerdem über die Methode

```
public boolean anmelden(String username, String passwort),
```

mit der man sich unter Angabe von Benutzername und Passwort am Benutzerkonto anmelden kann. Bei erfolgreicher Anmeldung gibt die Methode *true* zurück, ansonsten *false*. Weiters verfügt die Klasse Benutzerkonto über eine Methode

```
public void sperren(),
```

die das Benutzerkonto sperrt, sodass eine Anmeldung nicht mehr möglich ist.

Schreiben Sie eine Unterklasse *SicheresBenutzerkonto* von Benutzerkonto mit Konstruktor

```
public SicheresBenutzerkonto(String uname, String pwd).
```

Überschreiben Sie weiters die Methode *anmelden* so, dass der Account nach drei aufeinanderfolgenden erfolglosen Anmeldeversuchen gesperrt wird.

Beispiel 10.3:
Schreiben Sie die Klasse *Kernkraftwerk* aus Beispiel 2.6 so um, dass jene Reaktoren, die den Test nicht bestehen, aus dem Kraftwerk entfernt (d.h. aus der ArrayList gelöscht) werden. Was muss dabei beachtet werden?

Beispiel 10.4:
Gegeben sei die Klasse *Person* mit einer Methode

```
public boolean sindGeschwister(Person andrePerson),
```

die genau dann *true* zurückgibt, wenn die Person und *andrePerson* Geschwister sind. Außerdem sind in *Person* die Methoden *equals* und *hashCode* aus *Object* überschrieben.

Schreiben Sie (in einer Klasse Ihrer Wahl) eine statische Methode

```
public static Set<Person> getEinzelkinder(Set<Person> diePersonen),
```

die jene Personen aus *diePersonen* in einem *Set* zurückgibt, die in *diePersonen* keine Geschwister haben.

Beispiel 10.5:
Schreiben Sie eine Klasse *Pruefungsverwaltung* mit einer statischen Methode

```
public static Map<String,Double> getPunkteGesamt(
     Map<String,Double> zwitest, Map<String,Double> abtest).
```

Diese Methode nimmt als Parameter die Ergebnisse des Zwischentests *zwitest* sowie des Abschlusstests *abtest* (jeweils als Map, die für jede Matrikelnummer die erreichte Punkteanzahl gespeichert hat) und gibt die sich daraus ergebende Gesamtpunkteanzahl (ebenfalls als Map) zurück. (Beachten Sie, dass es Studierende geben kann, die nur zu einem der beiden Tests antreten.)

Beispiel 10.6:
Gegeben seien die Klassen *Schirennfahrer* und *Schirennen*.

In der Klasse *Schirennfahrer* sind die Methoden *equals* und *hashCode* aus *Object* überschrieben.

Die Klasse *Schirennen* verfügt über die Methode

```
public List<Schirennfahrer> getResult(),
```

die die Schirennfahrer gemäß ihrer Platzierung (beginnend mit dem erstplatzierten) in einer List zurückgibt.

Schreiben Sie eine Klasse Ergebnisauswertung mit einer statischen Methode

```
public static Map<Schirennfahrer, Integer> getBestResult(
                                    List<Schirennen> dieRennen),
```

die für jeden Schirennfahrer, der zumindest ein Rennen bestritten hat, die beste (=niedrigste) Platzierung in *dieRennen* sucht, in einer Map speichert und diese zurückgibt.

Beispiel 10.7:
Gegeben seien die Klassen *Website* und *Browser*.

Die Klasse *Browser* verfügt über einen Konstruktor

```
public Browser(String[] einstellungen, String benutzer)
```

sowie über eine Methode

```
public Website visit(String url),
```

die die Website mit der Adresse *url* besucht und zurückgibt.

Schreiben Sie eine Klasse *BrowserMitHistory*, die von *Browser* erbt, mit einem Konstruktor Ihrer Wahl.

Die Klasse *BrowserMitHistory* soll weiters über eine Methode

```
public Set<String> getVisitedUrls(int minNrVisits)
```

verfügen, die die Adressen aller Seiten in einem Set zurückgibt, die mindestens *minNrVisits*-mal besucht wurden.

Beispiel 10.8:
Gegeben sei eine Klasse *Handy* mit parameterlosem Konstruktor und einer Methode

```
void sendeNachricht(String empfaengerNr, String txt),
```

die den entsprechenden Text an die übergebene Nummer des Empfängers sendet.

Schreiben Sie eine Klasse *GehacktesHandy* mit Konstruktor

```
GehacktesHandy(String hackerNr).
```

Überschreiben Sie in *GehacktesHandy* die Methode *sendeNachricht* so, dass alle bisher gesendeten Nachrichten über die Methode

```
void sendeAllesAnHacker()
```

an die im Konstruktor übergebene Nummer des Hackers geschickt werden können. Dabei soll nur eine einzige Nachricht an den Hacker versandt werden, deren Text sowohl die Texte als auch die Empfängernummern aller bisher versendeten Nachrichten enthält.