**Aufgabe 11.1 (Ü) Überschreiben und Überladen von Methoden**

Betrachten Sie folgenden Klassendeklarationen:

```
public class A {
    public void f() {
        System.out.println("bin Methode f in Klasse A");
    }
    public void g() {
        System.out.println("bin Methode g in Klasse A");
    }
}
public class B extends A {
    public void f() {
        System.out.println("bin Methode f in Klasse B");
    }
}
public class C {
    private A a;
    public C(A a) {
         this.a = a;
    }
    public void g(){
         System.out.println("bin Methode g in Klasse C");
          a.f();
    }
}
public class D {
    public void f(A a) {
        a.f();
        System.out.println("bin Methode f in Klasse D mit Typ A");
    }
    public void f(B b) {
        b.f();
        System.out.println("bin Methode f in Klasse D mit Typ B");
    }
}
public static void main (String[] s){
    A a0 = new A();
    a0.f();
    a0.g();
    A a1 = new B();
    a1.f();
    a1.g();

    C c0 = new C(a0);
    c0.g();
```

```
    C c1 = new C(a1);
    c1.g();

    D d = new D();
    d.f(a0);
    d.f(a1);
    d.f(new B());
}
```

Welche Ausgaben erwarten Sie für die jeweiligen Methodenaufrufe? Begründen Sie Ihre Antwort.

## Aufgabe 11.2 (Ü) Termine

Termine haben haben einen Zeitpunkt, eine Beschreibung und finden an einem Ort statt. Man unterscheidet Termine, die

- einmalig,
- wiederholt alle $n$ Tage (endlos) sowie
- wiederholt alle $n$ Tage innerhalb einer festen Zeitspanne

stattfinden. Ziel dieser Aufgabe ist es, diese Hierarchie und die entsprechenden Gemeinsamkeiten durch Vererbung möglichst gut umzusetzen.

Zur Vereinfachung gehen wir davon aus, dass alle Zeitangaben (also auch Beginn und Ende von Zeitspannen) in Tagen nach Beginn Ihres Studiums (1. Oktober) erfolgen. Der `int`-Wert 6 repräsentiert also den 7. Oktober.

Programmieren Sie eine Klasse `Termin` sowie die davon abgeleitete Klassenhierarchie zur Darstellung einmaliger, sich endlos wiederholender sowie sich nur innerhalb einer vorgegebenen Zeitspanne wiederholender Termine.

Ausserdem soll die Klasse `Termin` eine Methode `int diff(int tag)` zur Verfügung stellen, die bestimmt, in wievielen Tagen relativ zum Zeitpunkt `tag` der entsprechende Termin zum nächsten Mal stattfindet. Im Falle vergangener, einmaliger Termine wird entsprechend eine negative Zahl zurückgegeben. Für sich wiederholende Termine betrachten Sie die folgenden Beispiele:

- Ein sich endlos wiederholender Termin, der seit dem 25. Tag jede Woche wiederholt stattfindet, liefert für `t.diff(27)` die Zahl 5 zurück.
- Ein Termin, der sich in der Zeit vom 2. bis zum 32. Tag alle 3 Tage wiederholt $(2, 5, \ldots, 26, 29, 32)$, liefert für `t.diff(27)` die Zahl 2 zurück.
- Für denselben Termin, der also vom 2. bis zum 32. Tag alle 3 Tage stattfindet, liefert `t.diff(33)` entsprechend die Zahl -1 zurück.

Schreiben Sie eine Klasse `TerminTest`, in deren `main`-Methode Sie Ihre Klassen testen.

## Aufgabe 11.3 (H) Terminkalender

Führen Sie Aufgabe 11.2 fort in dem Sie die Klasse `Terminkalender` implementieren, mit deren Hilfe

- Termine eingetragen (`void fuegeTerminHinzu(Termin t)`),
- Termine eines Tages abgefragt (`Termin[] termineAm(int tag)`) und
- der relativ zu einem Tag `tag` zukünftig als nächstes stattfindende Termin bestimmt (`Termin naechsterTermin(int tag)`)

werden können. **Hinweis:** Zur Verwaltung der Liste der Termine können Sie die angepas-

ste Listenimplementierung auf der Übungshomepage verwenden. Beachten Sie auch das Sie Objekte der Unterklassen in Listen/Arrays des Typen einer Oberklasse Speichern können. Dieses Konzept nennt man dynamisches Binden und wird in der nächsten Vorlesung behandelt. In unserem Fall können Sie Termine jeglichen Typs in einer Liste mit einträgen des Klassentyps `Termin` speichern.

Schreiben Sie eine Klasse `TerminkalenderTest`, in deren `main`-Methode Sie Ihre Klassen und die erwarteten Rückgabewerte testen.