TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Lehrstuhl für Sprachen und Beschreibungsstrukturen
Praktikum Grundlagen der Programmierung
Prof. Dr. Helmut Seidl

WS 2007/2008
Aufgabenblatt 9
14. Dezember 2007

Abgabe: keine

Praktikum Grundlagen der Programmierung

Aufgabe 44 (Ü) Generizität

Auf einem der letzten Übungsblätter haben Sie eine doppelt verkettete Liste kennenglernt.

a) Realisieren Sie diese doppelt verkettete Liste mit einem Typparameter.
b) `Iterierer` ist ein (generisches) Interface, um sequentiell auf eine Sammlung von Objekten zuzugreifen. Implementieren Sie einen `Iterierer`, der die Möglichkeit bietet über die Objekte einer Sammlung mit Hilfe der Methoden `next()` und `hasNext()` zu iterieren.
c) Implementieren Sie eine Klasse `SortierPruefer`, die über Listen beliebigen Typs iteriert und überprüft, ob diese in aufsteigender Reihenfolge sortiert sind. Verwenden Sie zu diesem Zwecke das (generische) Interface `Vergleichbar`, dessen Typparameter beschreibt, zu welchem Typ ein `Vergleichbar`-Objekt vergleichbar ist.

Hinweis: Die Klassen `VerketteteListe` und `Eintrag` können Sie von der Übungsseite herunterladen.

Aufgabe 45 (Ü) Vererbung und Interfaces

Gegeben seien folgende Java-Klassen:

```
class A <T>{
    public void f() { System.out.println("f␣in␣A"); }
    public void g() { System.out.println("g␣in␣A"); f(); }
    public T    i() { return null; }
}
class B extends A {
    public void f() { System.out.println("f␣in␣B"); }
    public void h() { System.out.println("h␣in␣B"); f(); g(); }
}
class C extends A {
    public void h() { System.out.println("h␣in␣C"); f(); g(); }
}
interface I<T>{
    public void m();
    public T l();
}
abstract class D {
    public abstract void n(boolean b);
    public void o() { System.out.println("o␣in␣D"); n(false); }
}
class E extends D {
```

```
    public void n(boolean b) {
        System.out.println("n in E");
        if(b) o();
    }
    public void m() { System.out.println("m in E"); }
}
class F<T> extends E implements I<T> {
    public void n(boolean b) { System.out.println("n in F"); }
    public T l() { return null; }
}
class G<T> extends A<T> {
    public void f() { System.out.println("f in G<T>");}
}
```

a) Stellen Sie die Klassen-Hierarchie mithilfe eines UML-Diagramms dar.

b) Welche der folgenden Zuweisungen sind erlaubt und welche nicht?

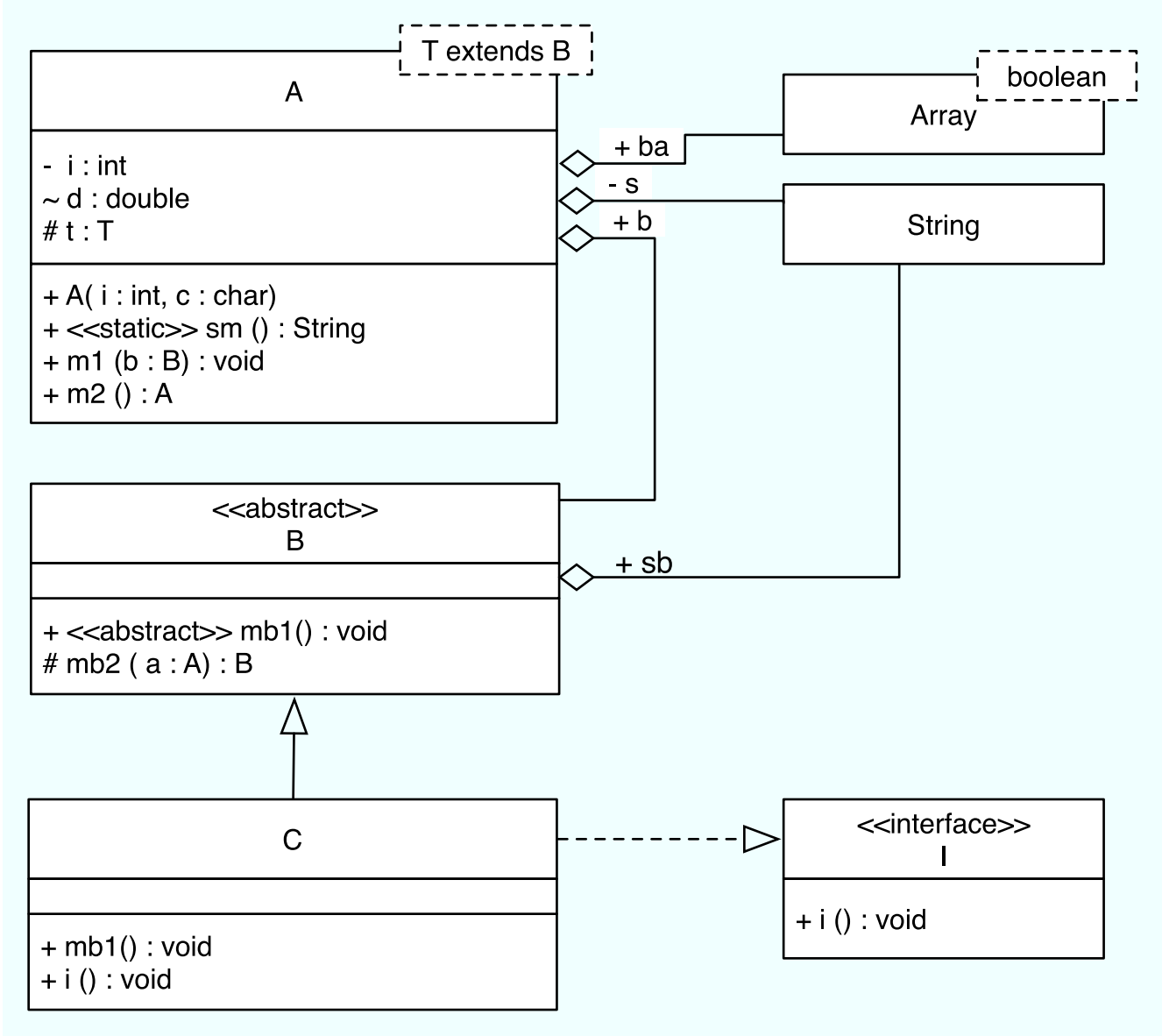
(i) `A a = new A();`
(ii) `B b = new B(); A t = b;`
(iii) `B b = new A();`
(iv) `D d = new D();`
(v) `D d = new F();`
(vi) `F f = new F(); I i = f;`
(vii) `A a = new A<B>();`
(viii) `A<B> a = new A();`
(ix) `A a = new G();`
(x) `A<D> a = new G<E>();`
(xi) `A<? extends E> a = new G<F<A<B>>>();`
(xii) `E e = a.i();`
(xiii) `F f = a.i();`

c) Welche Ausgaben produzieren die folgenden Anweisungen?

(i) `A a = new A(); a.g();`
(ii) `B b = new B(); b.h();`
(iii) `C c = new C(); c.g();`
(iv) `B b = new B(); A a = b; a.g();`
(v) `D d = new E(); d.n(true);`
(vi) `F f = new F(); f.o();`

UML Referenz

```
public class A<T extends B>{
    private int i;
    double d;
    protected T t;
    public B b;
    private String s;
    public boolean[] ba;

    public A(int i, char c) {
        ...
    }

    public void m1(B b) {
        ...
    }

    public A m2() {
        ...
    }

    public static String sm() {
        ...
    }
}

public abstract class B {
    public String sb;
    public abstract void mb1();
    protected B mb2(A a) {
        ...
    }
}

public interface I {
    public void i();
}

public class C extends B implements I {
    public void mb1() {
        ...
    }
    public void i() {
        ...
    }
}
```


T extends B
A
- i : int
~ d : double
# t : T
+ A(i : int, c : char)
+ <<static>> sm () : String
+ m1 (b : B) : void
+ m2 () : A
boolean
Array
+ ba
- s
+ b
String
<<abstract>>
B
+ sb
+ <<abstract>> mb1() : void
# mb2 (a : A) : B
C
+ mb1() : void
+ i () : void
<<interface>>
I
+ i () : void