Tagesprogramm

Interface

Klassenableitung

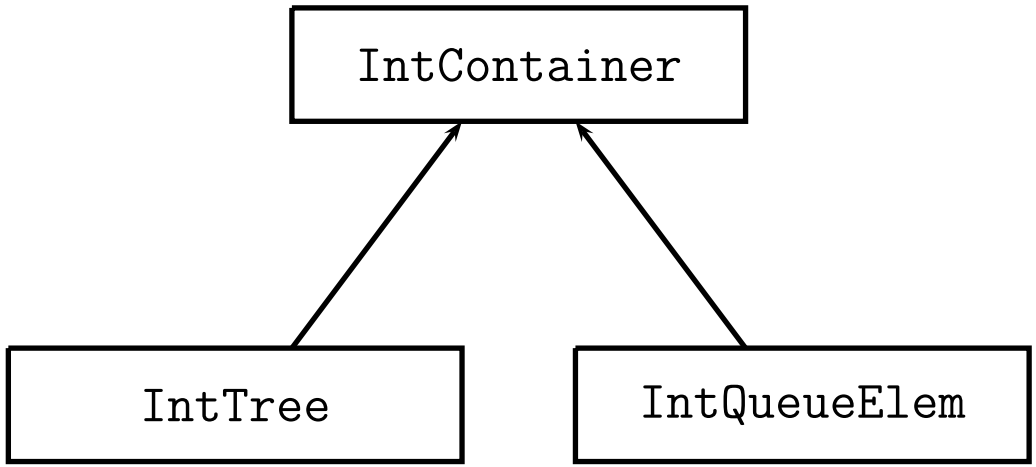
Definition eines Interfaces

```
// a container holding values of type int
public interface IntContainer {

    // add elem to the container
    void add(int elem);

    // is elem in the container?
    boolean element(int elem);

    // is the container empty?
    boolean empty();
}
```

Klassen implementieren Interfaces

```
// a tree holding values of type int
public class IntTree implements IntContainer {
    public void add(int elem) {...}
    public boolean element(int elem) {...}
    public boolean empty() {...}
}

// a queue with search holding values of type int
public class IntQueueElem implements IntContainer {
    public void add(int elem) {...}
    public boolean element(int elem) {...}
    public boolean empty() {...}
}
```

Typhierarchie



Verwendung von Interfaces

```
public class IntContainerTest {
    public static void test(IntContainer container) {
        ... container.add(...) ...
        ... container.empty() ...
        ... container.element(...) ...
    }
}
```

```
IntContainer cont = new IntTree();
IntContainerTest.test(cont);
cont = new IntQueueElem();
IntContainerTest.test(cont);
```

Aufgabe: Warum Interfaces?

Erfahrene Programmierer(innen) verwenden häufig Interfaces und nicht direkt die Klassen, welche die Interfaces implementieren.

Welche dieser Aussagen können als Begründung dafür dienen?

A Interfaces erhöhen den Abstraktionsgrad.

B Dieselbe Variable kann Objekte unterschiedlicher Klassen enthalten.

C Eine Methode kann mit Objekten unterschiedlicher Typen umgehen.

D Interfaces erleichtern den Austausch von Implementierungen.

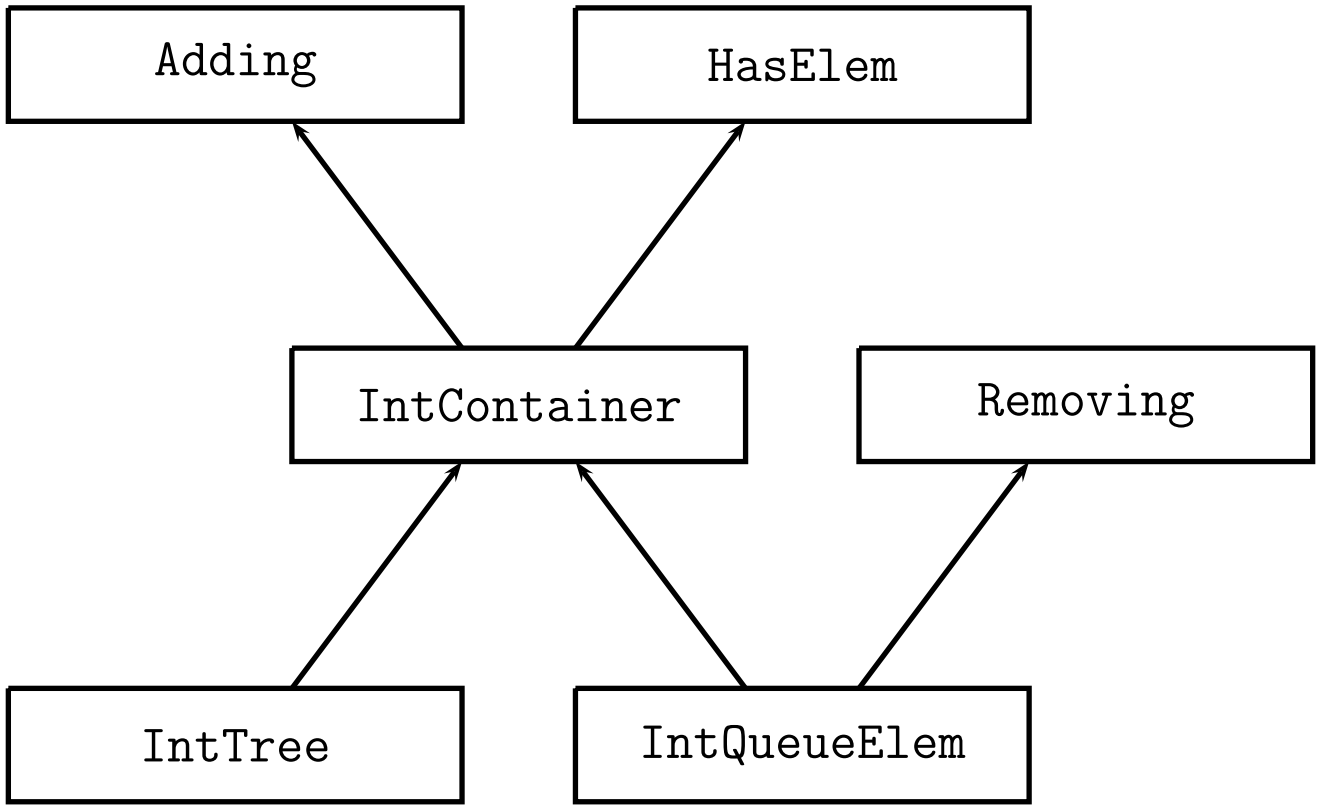
Mehrere Interfaces

```
public interface Removing {
    int remove();
}

public interface Adding {
    void add(int elem);
}

public interface HasElem {
    boolean element(int elem);
}


public class IntQueueElem
    implements IntContainer, Removing, Adding, HasElem {...}
```

Erweiterung von Interfaces

```
public interface IntContainer extends Adding, HasElem {
    boolean empty();
    boolean element(int elem);
}

public class IntQueueElem
             implements IntContainer, Removing {...}
```

Erweiterte Typhierarchie



Aufgabe: Verwendbarkeit

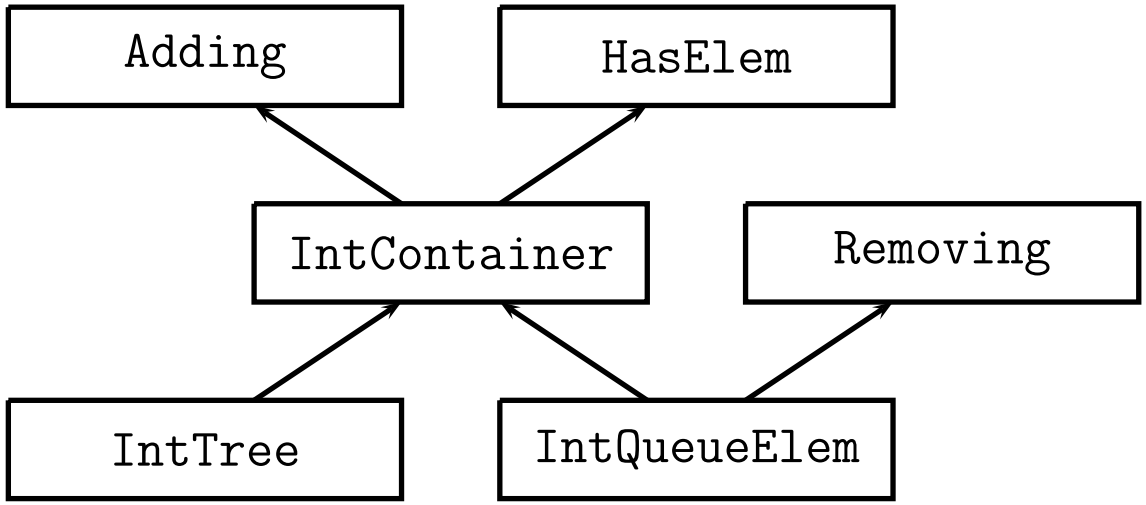
Welche der folgenden Aussagen sind korrekt?

A Objekte von `Removing` sind auch Objekte von `IntContainer`

B Objekte von `IntContainer` sind auch Objekte von `Adding`

C statt `IntQueueElem` kann man `IntTree` verwenden

D statt `HasElem` kann man `IntTree` verwenden



Unterklassen

Klasse von genau einer anderen Klasse **abgeleitet**
(ohne `extends`-Klausel von `Object` abgeleitet)

```
public class IntTree extends Object
                     implements IntContainer { ...}
```

`IntTree` ist **Unterklasse** von `Object`,
`Object` ist **Oberklasse** von `IntTree`

`IntTree` ist **Untertyp** von `Object` und `IntContainer`,
`Object` und `IntContainer` sind **Obertypen** von `IntTree`

Vererbung

Unterklasse **erbt** nicht-statische Methoden und Variablen von Oberklasse

ererbte Methoden können **überschrieben** (neu implementiert) werden

historisch gesehen war Vererbung sehr wichtig,
hat heute stark an Bedeutung verloren,
oft nur mehr als Alternative zu Interfaces gesehen

Aufgabe: Untertypen versus Unterklassen

Welche der folgenden Aussagen sind korrekt?

A ist U Untertyp von T, dann ist U auch Unterklasse von T

B ist U Unterklasse von T, dann ist U auch Untertyp von T

C `extends` erzeugt Unterklassen, `implements` Untertypen

D `extends` verwendet man für Klassen, `implements` für Interfaces

Object

`java.lang.Object` ist die oberste Oberklasse aller Klassen

Methoden von `Object` sind in jeder Klasse vorhanden:

```
public String toString() {...}
public boolean equals(Object other) {...}
public int hashCode() {...}
...
```

Aufgabe: Untertypen und Unterklassen

Suchen Sie in Gruppen zu 2 bis 3 Personen Antworten auf folgende Fragen:

Aus welchen Gründen verwendet man Untertypen und Unterklassen?

Kann man auf Unterklassen verzichten, wenn man Untertypen hat?

Kann man auf Untertypen verzichten, wenn man Unterklassen hat?