PGdP

Linked List

Streams

Lambdas

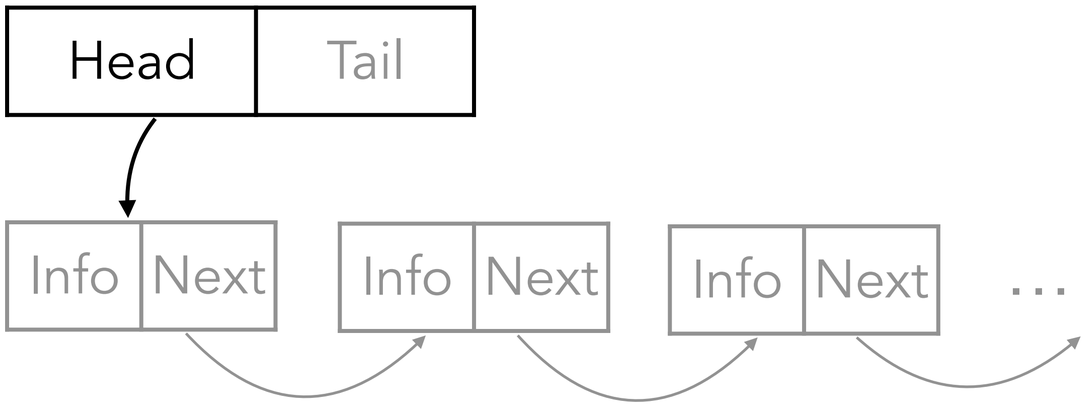
Polymorphie

Folien: go.tum.de/904005

Linked List

Achtung: Diese Implementierung ist sehr beschränkt und behandelt keine Edge-Cases!

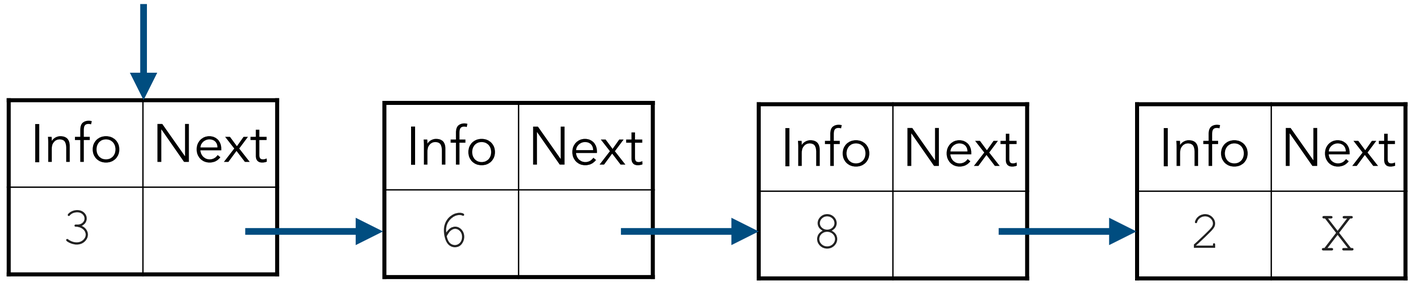
Normalerweise würde man ein zusätzliches Objekt 'über' den Listenelementen anlegen.



Linked List

Einfach verkettete Liste, Überblick

```
List liste;
```

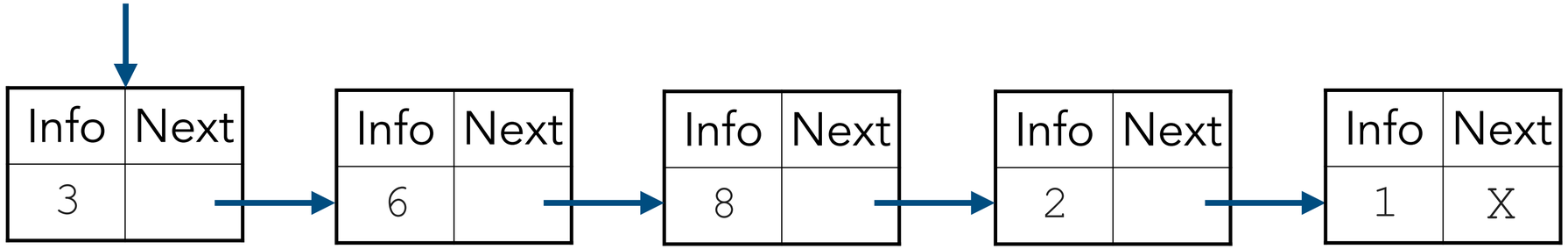



```
class List {
  int info;
  List next;
  public List(int info, List next) {
    this.info = info; this.next = next;
  }
}
```

Linked List

Einfach verkettete Liste, Einfügen

```
List liste;
```

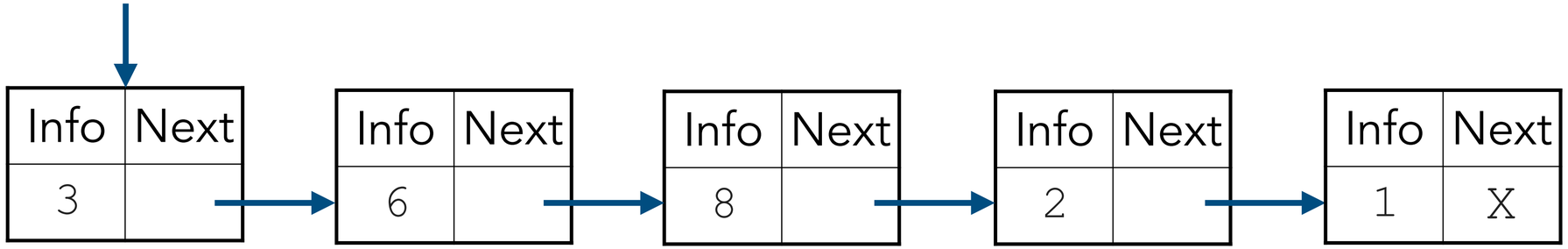



```
void insert(int info) {
  List current = this;
  while(current.next != null)
    current = current.next;
  current.next = new List(info, null);
}
```

Linked List

Einfach verkettete Liste, Indexzugriff

```
List liste;
```

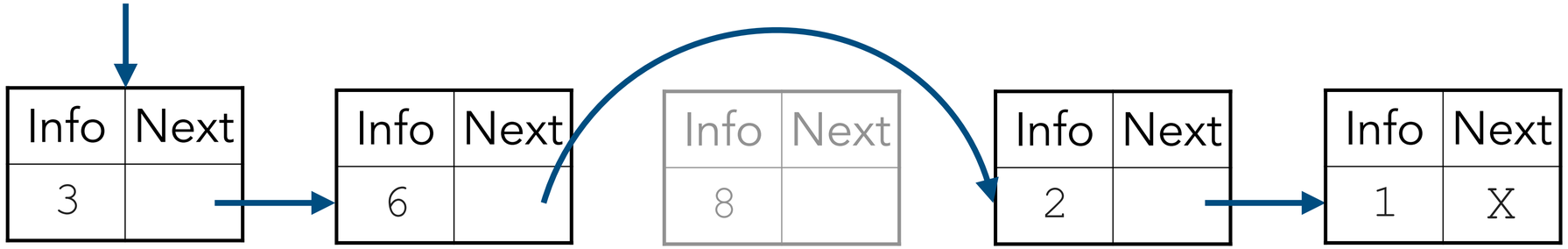



```
List get(int at) {
  List current = this; int count = 0;
  if(at < size())
    while(count < at) {
      current = current.next;
      count ++;
    }
  return current; //returnt this für ungültig
}
```

Linked List

Einfach verkettete Liste, Löschen

```
List liste;
```

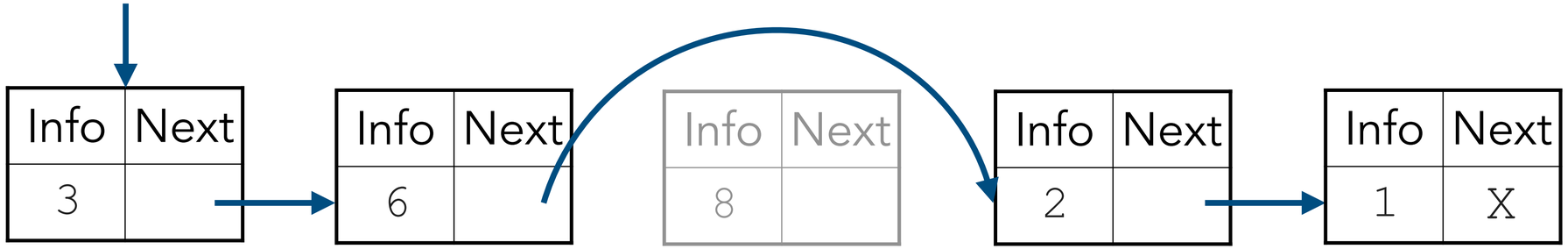



```
List delete(int at) {
  if(at == 0) return this.next;
  if(at + 1 < size()) {
    List preRemove = get(at - 1);
    preRemove.next = preRemove.next.next;
  } else { //letztes Element
    (get(at - 1)).next = null;
  }
  return this;
```

Linked List

Einfach verkettete Liste, Größe

```
List liste;
```




```
int size() {
  List current = this; int count = 1;
  while(current.next != null) {
    current = current.next;
    count ++;
  }
  return count;
}
```

Streams

Vergleiche: Fließband, auf dem verschiedene Operationen ausgeführt werden.

Quelle (erzeugt Stream):

```
- Stream.of( */ values */ );
- Arrays.stream( */ array */ );
- Collection.stream; //für alle Collections
- IntStream.range( /* from */ , /* to */ );
```


Basierend auf: Java 8 Streams Tutorial - Joe James

Streams

Vergleiche: Fließband, auf dem verschiedene Operationen ausgeführt werden.

Intermediäre Operationen (verändern Stream):

- `map(Function<From, To>)`
- `filter(Predicate<Type>)`
- `distinct()`
- `sorted(Comparator<Type>)`
- `...`

Basierend auf: Java 8 Streams Tutorial - Joe James

Streams

Vergleiche: Fließband, auf dem verschiedene Operationen ausgeführt werden.

```
map(Function<From, To>)
abstract class Function<From, To> {
  abstract To apply(From input);
}

filter(Predicate<Type>)
abstract class Predicate<Type> {
 abstract boolean test(Type input);
}

sorted(Comparator<Type>);
abstract class Comparator<Type> {
  abstract int compareTo(Type a, Type b);
}
```

Streams

Vergleiche: Fließband, auf dem verschiedene Operationen ausgeführt werden.

Terminale Operation (beenden Stream):

- `forEach(Consumer<Type>)`
- `reduce(Type, BinaryOperator<Type>)`

Identität (Startwert)

Basierend auf: Java 8 Streams Tutorial - Joe James

Streams

Vergleiche: Fließband, auf dem verschiedene Operationen ausgeführt werden.

```
forEach(Consumer<Type>)
abstract class Consumer<Type> {
  abstract void accept(Type input);
}

reduce(Type, BinaryOperator<Type>)
abstract class BinaryOperator<Type> {
  abstract Type apply(Type a, Type b);
}
```

Die angegebenen Klassen sind vereinfacht. Teilweise handelt es sich auch um Interfaces.

Lambdas

Lambdas, Beispiel map(Function<From, To>), quadrieren

```
map(Function<From, To>)
abstract class Function<From, To> {
  abstract To apply(From input);
}
```

Function hat nur eine Methode, deren Körper definiert werden muss. Das geht mit Lambda-Ausdrücken, einer anonymen Klasse oder einer separaten Klasse

Lambdas

Lambdas, Beispiel map(Function<From, To>), quadrieren

> in eigener Klasse

```
class QuadratFunktion extends
 Function<Integer, Integer> {
  Integer apply(Integer input) {
    return (Integer) (input * input);
  }
}
```

Verwendung

```
map(new QuadratFunktion())
```

Lambdas

Lambdas, Beispiel map(Function<From, To>), quadrieren

> in anonymer Klasse

```
Function<Integer, Integer> quadratFunktion
 = new Function<Integer, Integer>() {
    Integer apply(Integer input) {
    return (Integer) (input * input);
  }
};
```

Verwendung

```
map(quadratFunktion)
```

Lambdas

Lambdas, Beispiel map(Function<From, To>), quadrieren

> mit Lambda

```
Function<Integer, Integer> quadratFunktion
 = (input) -> (input * input)
```

Verwendung

```
map(quadratFunktion)
```

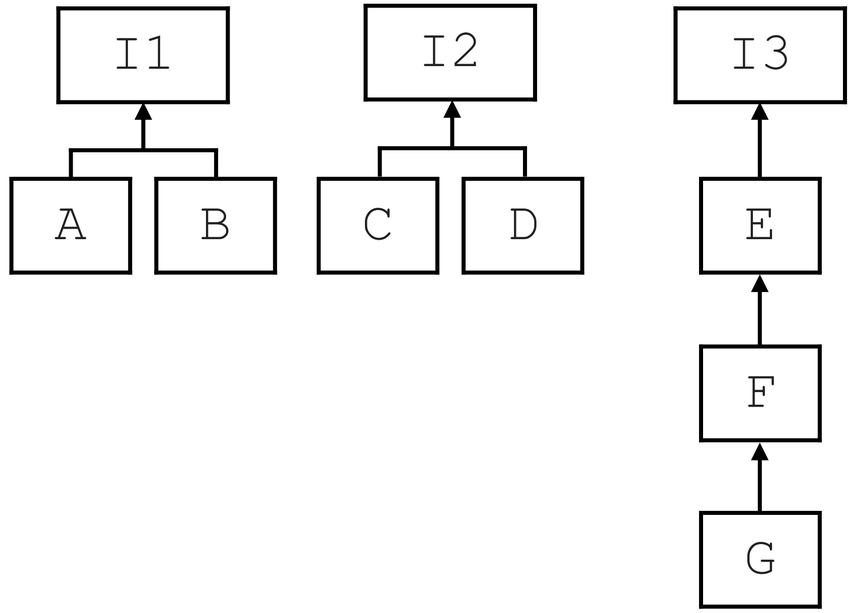
Man kann den Lambda-Ausdruck auch direkt übergeben:

```
map(i -> i*i)
```

Polymorphie

1: Klassenhierarchie Visualisieren

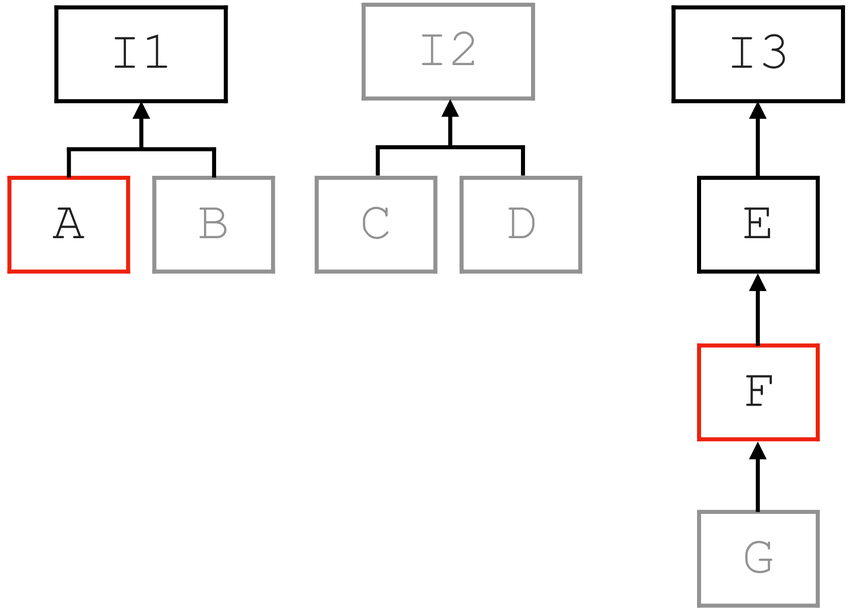
```
interface I1
interface I2
interface I3
class A implements I1
class B implements I1
class C implements I2
class D implements I2
class E implements I3
class F extends E
class G extends F
```



Polymorphie

1: Klassenhierarchie Visualisieren

```
interface I1
interface I2
interface I3
class A implements I1
class B implements I1
class C implements I2
class D implements I2
class E implements I3
class F extends E
class G extends F
```




```
f(A first, F second)
> f((A|I1) first, (I3|E|F) second)
```

Polymorphie

2: Zur Compilezeit ausgewählte Methode bestimmen (stat.)
> speziellste Signatur mit den wenigsten impliziten Casts

3: Eventueller Dispatch, falls es für den dynamischen Typen eine Methode mit der gewählten Signatur gibt, die in der Hierarchie näher am dynamischen Typ liegt

Übersicht