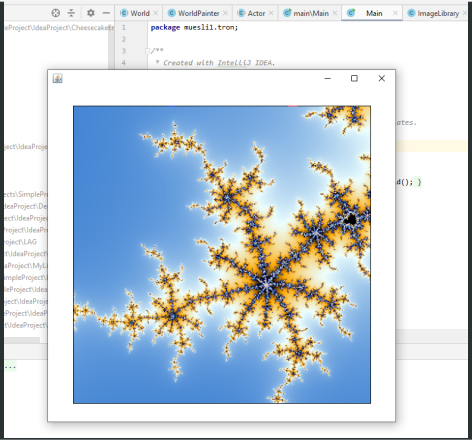
FOP Recap #10

Java Generics



Guten Tag!

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Generics Basiskonzept
- Instanziierung
- Nutzen von Typparameter
- Bennenung von Typparameter
- Mehrere Typparameter

Diamond operator

Generische Methoden

Statische generische Methoden

Eingeschränkte Typparameter

Generics!

„**Generics —** You think you understand it, and then you don't. . . “

Generics Basiskonzept

TECHNISCHE UNIVERSITÄT DARMSTADT

ohne Generics

```
1  public class StringHolder {
2      public String value;
3  }
4  public class IntegerHolder {
5      public Integer value;
6  }
```

mit Generics

```
1  public class Holder<T> {
2      public T value;
3  }
```

Generics Basiskonzept

TECHNISCHE UNIVERSITÄT DARMSTADT

ohne Generics

```
1  public class StringHolder {
2      public String value;
3  }
4  public class IntegerHolder {
5      public Integer value;
6  }
```

mit Generics

```
1  public class Holder<T> {
2      public T value;
3  }
```

Vorteile von Generics

- Lösung mit Generics kürzer

Generics Basiskonzept

TECHNISCHE UNIVERSITÄT DARMSTADT

ohne Generics

```
1  public class StringHolder {
2      public String value;
3  }
4  public class IntegerHolder {
5      public Integer value;
6  }
```

mit Generics

```
1  public class Holder<T> {
2      public T value;
3  }
```

Vorteile von Generics

- Lösung mit Generics kürzer
- Vermeidet Redundanz ⇒ Fehler vermeiden, Wartbarkeit verbessern

Generics Basiskonzept

Instanziierung

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class Holder<T> {
2      public T value;
3  }
```

Generics Basiskonzept

Instanziierung

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class Holder<T> {
    public T value;
}
```

```java
Holder<Integer> integerHolder = new Holder<Integer>();
Holder<Boolean> booleanHolder = new Holder<Boolean>();
```

Generics Basiskonzept

Nutzen von Typparameter

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class Holder<T> {
    public T value;

    public T getValue() {
        return value;
    }
    public void killItWithFire() {
        value = null;
    }
}
```

Generics Basiskonzept

Bennenung von Typparameter

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ein Großbuchstabe
- T für Type
- K für Key
- V für Value
- E für Element
- ....

Generics Basiskonzept

Mehrere Typparameter

TECHNISCHE UNIVERSITÄT DARMSTADT

ohne Generics

```
1  public class
   ↪  StringAndDoubleHolder {
2      public String valueA;
3      public Double valueB;
4  }
5  public class
   ↪  IntegerAndShortHolder {
6      public Integer valueA;
7      public Short valueB;
8  }
```

mit Generics

```
1  public class Pair<A, B> {
2      public A valueA;
3      public B valueB;
4  }
```

Das steht heute auf dem Plan

Generics Basiskonzept

Diamond operator

Generische Methoden

Statische generische Methoden

Eingeschränkte Typparameter

Erben von generischen Klassen

Diamond operator

TECHNISCHE UNIVERSITÄT DARMSTADT

Diamond Operator

```
1  ArrayList<String> myList1 = new ArrayList<String>();
2  ArrayList<String> myList2 = new ArrayList<>(); // String wird impliziert
3  List<String> myList3 = new Arraylist<>();
```

Das steht heute auf dem Plan

Generics Basiskonzept

Diamond operator

Generische Methoden

Statische generische Methoden

Eingeschränkte Typparameter

Erben von generischen Klassen

Generische Methoden

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public List<Double> nonGenericFilter(List<Double> list,
   ↪  Predicate<Double> predicate) {
2      // ....
3  }
4  public List<Boolean> nonGenericFilter(List<Boolean> list,
   ↪  Predicate<Boolean> predicate) {
5      // ....
6  }
```

Generische Methoden

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public List<Double> nonGenericFilter(List<Double> list,
    Predicate<Double> predicate) {
    // ....
}
public List<Boolean> nonGenericFilter(List<Boolean> list,
    Predicate<Boolean> predicate) {
    // ....
}
```

```java
public <T> List<T> filter(List<T> list, Predicate<T> predicate) {
    // ....
}
```

Das steht heute auf dem Plan

Generics Basiskonzept

Diamond operator

Generische Methoden

Statische generische Methoden

Eingeschränkte Typparameter

Erben von generischen Klassen

Statische generische Methoden

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public static <T> List<T> filter(List<T> list, Predicate<T> predicate) {
2      // ....
3  }
```

Das steht heute auf dem Plan

Generics Basiskonzept

Diamond operator

Generische Methoden

Statische generische Methoden

Eingeschränkte Typparameter
- `extends` Klassen
- `extends` Interfaces
- `extends` Klassen und Interfaces

Eingeschränkte Typparameter

`extends` Klassen

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class Holder<T> {
2      public T value;
3      public void test() {
4          int i = // ...?
5      }
6  }
```

Eingeschränkte Typparameter

`extends` Klassen

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class Holder<T extends Number> {
2      public T value;
3      public void test() {
4          int i = value.intValue();
5      }
6  }
```

Eingeschränkte Typparameter

`extends` Klassen

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class Holder<T extends Number> {
    public T value;
    public void test() {
        int i = value.intValue();
    }
}
```

```java
Holder<Integer> holder1 = new Holder<Integer>();
Holder<Boolean> holder2 = new Holder<Boolean>(); // !!
```

Eingeschränkte Typparameter

`extends` Interfaces

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class Holder<T extends Inf1 & Inf2 & Inf3> {
2      public T value;
3      // ...
4  }
```

Eingeschränkte Typparameter

`extends` Klassen und Interfaces

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class Holder<T extends Number & Inf1 & Inf2 & Inf3> {
2      public T value;
3      // ...
4  }
```

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Generics Basiskonzept

Diamond operator

Generische Methoden

Statische generische Methoden

Eingeschränkte Typparameter

Erben von generischen Klassen

Erben von generischen Klassen

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class MyGenericClass<T> {
2      // ...
3  }
```

Erben von generischen Klassen

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class MyGenericClass<T> {
    // ...
}
```

```java
public class Sub<Q> extends MyGenericClass<Q> {
    // ...
}
```

Erben von generischen Klassen

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class MyGenericClass<T> {
    // ...
}
```

```java
public class Sub<Q> extends MyGenericClass<Q> {
    // ...
}
```

```java
public class IntegerSub extends MyGenericClass<Integer> {
    // ...
}
```

Erben von generischen Klassen

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class MyGenericClass<T extends Integer> {
2      // ...
3  }
```

Erben von generischen Klassen

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class MyGenericClass<T extends Integer> {
    // ...
}
```

```java
public class IntegerSub extends MyGenericClass<Integer> {
    // ...
}
```

Erben von generischen Klassen

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class MyGenericClass<T extends Integer> {
    // ...
}
```

```java
public class IntegerSub extends MyGenericClass<Integer> {
    // ...
}
```

```java
public class BooleanSub extends MyGenericClass<Boolean> { // !!
// ...
}
```

Erben von generischen Klassen

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class MyGenericClass<T extends Integer> {
2      // ...
3  }
```

Erben von generischen Klassen

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class MyGenericClass<T extends Integer> {
2      // ...
3  }
```

```
1  public class IntegerSub extends MyGenericClass<Integer> {
2      // ...
3  }
```

Erben von generischen Klassen

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class MyGenericClass<T extends Integer> {
    // ...
}
```

```java
public class IntegerSub extends MyGenericClass<Integer> {
    // ...
}
```

```java
public class BooleanSub extends MyGenericClass<Boolean> { // !!
// ...
}
```

Das steht heute auf dem Plan

Generics Basiskonzept

Diamond operator

Generische Methoden

Statische generische Methoden

Eingeschränkte Typparameter

Erben von generischen Klassen

Erstellen von generischen Arrays

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class MyGenericClass<T> {
2      public T[] createMyArray() {
3          T[] testArray = (T[]) new Object[25];
4          return testArray;
5      }
6  }
```

Erstellen von generischen Arrays

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class MyGenericClass<T> {
    public T[] createMyArray() {
        T[] testArray = (T[]) new Object[25];
        return testArray;
    }
}
```

```java
public <Q> Q[] createMyArray() {
    Q[] testArray = (Q[]) new Object[25];
    return testArray;
}
```

Das steht heute auf dem Plan

Generics Basiskonzept

Diamond operator

Generische Methoden

Statische generische Methoden

Eingeschränkte Typparameter

Erben von generischen Klassen

Generic Variance

? Wildcards

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public <T> void printList(List<T> list) {
2      for (T item : list) {
3          System.out.println(item);
4      }
5  }
```

Generic Variance

? Wildcards

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public <T> void printList(List<T> list) {
    for (T item : list) {
        System.out.println(item);
    }
}
```

```java
public void printList(List<?> list) {
    for (Object item : list) {
        System.out.println(item);
    }
}
```

Generic Variance

? Wildcards

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public <T extends Number> void printList(List<T> list) {
    for (T item : list) {
        System.out.println(item.intValue() + 1);
    }
}
```

Generic Variance

? Wildcards

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public <T extends Number> void printList(List<T> list) {
    for (T item : list) {
        System.out.println(item.intValue() + 1);
    }
}
```

```java
public void printList(List<? extends Number> list) {
    for (Number item : list) {
        System.out.println(item.intValue() + 1);
    }
}
```

Generic Variance

Wann werden wildcards verwendet?


```
1  public void nonGenMethod(Holder<? extends Number> a, Holder<?> b) {
2      // ...
3  }
```

Generic Variance

Wann werden wildcards verwendet?

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public void nonGenMethod(Holder<? extends Number> a, Holder<?> b) {
    // ...
}
```

```java
public <T extends Number> void genMethodA(Holder<T> a, Holder<T> b) {
    // ...
}
```

Generic Variance

Wann werden wildcards verwendet?

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public void nonGenMethod(Holder<? extends Number> a, Holder<?> b) {
    // ...
}
```

```java
public <T extends Number> void genMethodA(Holder<T> a, Holder<T> b) {
    // ...
}
```

```java
public <T extends Number, Q> void genMethodB(Holder<T> a, Holder<Q> b) {
    // ...
}
```

Generic Variance

? Wildcards mit super

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public void nonGenMethod(Predicate<? super Number> a) {
2      a.intValue(); // !! Nicht erlaubt
3  }
```

Generic Variance

? Wildcards mit super

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public void nonGenMethod(Predicate<? super Number> a) {
2      a.intValue(); // !! Nicht erlaubt
3  }
```

```
1  Predicate<? super Number> a = (Object x) -> true;
2  Predicate<? super Number> b = (Serializable x) -> true;
3  Predicate<? super Number> c = (Number x) -> true;
```

Producers **E**xtend **C**onsumers **S**uper
(PECS)

Generic Variance

Wann `super`, wann `extends`?

TECHNISCHE UNIVERSITÄT DARMSTADT

Generic Variance

- `? extends T` - Covariance (Lesender Zugriff)
- `? super T` - Contravariance (Schreibender Zugriff)
- `T` - Invariance (Beides)

Generic Variance

Covariance - Parameters

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class A {}
2  public class B extends A {}
```

Generic Variance

Covariance - Parameters

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class A {}
2  public class B extends A {}
```

```
1  public void foo(A a) { ... }
```

Generic Variance

Covariance - Parameters

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class A {}
2  public class B extends A {}
```

```
1  public void foo(A a) { ... }
```

```
1  foo(new A());
2  foo(new B()); // parameter covariance
```

Generic Variance

Covariance - Return type

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class A {}
2  public class B extends A {}
```

Generic Variance

Covariance - Return type

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class A {}
public class B extends A {}
```

```java
public class Baz {
    public A foo() {...}
}
public class SpecialBaz extends Baz {
    @Override
    public B foo() {...} // return type covariance
}
```

Generic Variance

Covariance - Arrays

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class A {}
2  public class B extends A {}
```

Generic Variance

Covariance - Arrays

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class A {}
public class B extends A {}
```

```java
A[] myArray = new B[2]; // array covariance
```

Generic Variance

Covariance - Arrays

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class A {}
public class B extends A {}
```

```java
A[] myArray = new B[2]; // array covariance
```

```java
A[] myArray = new A[2];
myArray[0] = new B(); // array covariance
```

Generic Variance

Covariance - Arrays


```
1  public class A {}
2  public class B extends A {}
3  public class C extends A {}
```

Generic Variance

Covariance - Arrays

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class A {}
2  public class B extends A {}
3  public class C extends A {}
```

```
1  A[] myArray = new B[2];
2  myArray[0] = new B();
```

Generic Variance

Covariance - Arrays

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class A {}
public class B extends A {}
public class C extends A {}
```

```java
A[] myArray = new B[2];
myArray[0] = new B();
```

```java
A[] myArray = new B[2];
myArray[0] = new C(); // !! array covariance
```

Generic Variance

Invariance

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class A {}
2  public class B extends A {}
```

Generic Variance

Invariance

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class A {}
2  public class B extends A {}
```

```
1  List<B> myList = new ArrayList<B>();
2  List<A> otherList = myList; // !!
3
4  List<? extends A> list = myList;
```

Generic Variance

Invariance

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class A {}
public class B extends A {}
```

```java
List<B> myList = new ArrayList<B>();
List<A> otherList = myList; // !!

List<? extends A> list = myList;
list.add(new A()); // !! compile error
list.add(new B()); // !! compile error
```

Generic Variance

Invariance

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public void readFromList(List<? extends Number> list) {
    // z.B: List<Double>, List<Integer>, List<Number>
    Number a = list.get(0);
}
public void readFromList(List<? super Number> list) {
    // z.B: List<Number>, List<Object>
    Number a = list.get(0); // !!
}
```

Generic Variance

Wann super, wann extends?


```
1  public void writeToList(List<? extends Number> list) {
2      // z.B: List<Double>, List<Integer>, List<Number>
3      list.add(Double.valueOf(2.0)); // !!
4  }
5  public void readFromList(List<? super Number> list) {
6      // z.B: List<Number>, List<Object>
7      list.add(Double.valueOf(2.0));
8  }
```

Generic Variance

Wann super, wann extends?

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public void both(List<? extends Number> list) {
    // z.B: List<Double>, List<Integer>, List<Number>
    Number a = list.get(0);
    list.add(Double.valueOf(2.0)); // !!
}
public void readFromList(List<? super Number> list) {
    // z.B: List<Number>, List<Object>
    Number a = list.get(0); // !!
    list.add(Double.valueOf(2.0));
}
```

Generic Variance

Wann super, wann extends?

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public void both(List<Double> list) {
2      // nur List<Double>
3      Number a = list.get(0);
4      list.add(Double.valueOf(2.0));
5  }
```

Generic Variance

Wann `super`, wann `extends`? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

- Supplier

Generic Variance

Wann `super`, wann `extends`? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

- Supplier
- Consumer

Generic Variance

Wann `super`, wann `extends`? Beispiele

- Supplier
- Consumer
- Function

Generic Variance

Wann super, wann extends? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public interface Supplier<T> {
2      T get();
3  }
```

Generic Variance

Wann `super`, wann `extends`? Beispiele

`? super T` (Contravarianz)

Generic Variance

Wann super, wann extends? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

? super T (Contravarianz)

```
1  public void foo(Supplier<? super T> supplier) {
2      T a = supplier.get(); // !!
3  }
```

Generic Variance

Wann `super`, wann `extends`? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

T (Invarianz)

Generic Variance

Wann super, wann extends? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

T (Invarianz)

```
1  public void foo(List<T> list, Supplier<T> supplier) {
2      list.add(supplier.get());
3  }
4  public void bar() {
5      List<Number> list = new ArrayList<>();
6      Supplier<Integer> supplier = () -> 1;
7      foo(list, supplier); // !!
8  }
```

Generic Variance

Wann `super`, wann `extends`? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

`? extends T` (Covarianz)

Generic Variance

Wann super, wann extends? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

? extends T (Covarianz)

```
1  public void foo(List<T> list, Supplier<? extends T> supplier) {
2      list.add(supplier.get());
3  }
4  public void bar() {
5      List<Number> list = new ArrayList<>();
6      Supplier<Integer> supplier = () -> 1;
7      foo(list, supplier); // geht jetzt
8  }
```

Generic Variance

Wann `super`, wann `extends`? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

`? extends T` (Covarianz)

Generic Variance

Wann super, wann extends? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

? extends T (Covarianz)

```
1  public void foo(List<T> list, Supplier<? extends T> supplier) {
2      list.add(supplier.get());
3  }
4  public void bar() {
5      List<Number> list = new ArrayList<>();
6      Supplier<Integer> supplier = () -> 1;
7      foo(list, supplier); // geht jetzt
8  }
```

Generic Variance

Wann super, wann extends? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public interface Consumer<T> {
2      void accept(T t);
3  }
```

Generic Variance

Wann `super`, wann `extends`? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

`? extends T` (Covarianz)

Generic Variance

Wann super, wann extends? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

? extends T (Covarianz)

```
1  public void foo(Consumer<? extends Number> consumer) {
2      consumer.accept(new Integer()); // !!
3  }
```

Generic Variance

Wann `super`, wann `extends`? Beispiele

T (Invarianz)

Generic Variance

Wann super, wann extends? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

T (Invarianz)

```
1  public void foo(List<T> list, Consumer<T> consumer) {
2      list.add(supplier.get());
3  }
4  public void bar() {
5      List<Integer> list = new ArrayList<>();
6      Consumer<Number> consumer = n -> System.out.println(n.intValue());
7      foo(list, consumer); // !!
8  }
```

Generic Variance

Wann `super`, wann `extends`? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

`? super T` (Contravarianz)

Generic Variance

Wann super, wann extends? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

? super T (Contravarianz)

```
1  public void foo(List<T> list, Consumer<T> consumer) {
2      list.add(supplier.get());
3  }
4  public void bar() {
5      List<Integer> list = new ArrayList<>();
6      Consumer<Number> consumer = n -> System.out.println(n.intValue());
7      foo(list, consumer); // geht jetzt
8  }
```

Generic Variance

Wann super, wann extends? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public interface Function<T, R> {
2      R apply(T t);
3  }
```

Generic Variance

Wann `super`, wann `extends`? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

- `Supplier`

Generic Variance

Wann super, wann extends? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

- `Supplier` → `Supplier<? extends T>`

Generic Variance

Wann super, wann extends? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

- `Supplier` → `Supplier<? extends T>`
- `Consumer`

Generic Variance

Wann super, wann extends? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

- `Supplier` → `Supplier<? extends T>`
- `Consumer` → `Consumer<? super T>`

Generic Variance

Wann super, wann extends? Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

- `Supplier` → `Supplier<? extends T>`
- `Consumer` → `Consumer<? super T>`
- `Function`

Generic Variance

Wann super, wann extends? Beispiele

- `Supplier` → `Supplier<? extends T`
- `Consumer` → `Consumer<? super T`
- `Function` → `Function<? super T, ? extends R`

Das steht heute auf dem Plan

Generics Basiskonzept

Diamond operator

Generische Methoden

Statische generische Methoden

Eingeschränkte Typparameter

Erben von generischen Klassen

Type erasure

TECHNISCHE UNIVERSITÄT DARMSTADT

- Typparameter sind zur Laufzeit nicht bekannt!
- Im Prinzip nur Schutz vor Fehlern beim Kompilieren

Type erasure

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class Test<T extends Number> {
2      T val;
3      public void test(T p) {
4          }
5  }
```

Type erasure

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class Test<T extends Number> {
2      T val;
3      public void test(T p) {
4          }
5  }
```

```
1  public class Test<Number> {
2      Number val;
3      public void test(Number p) {
4          }
5  }
```

Type erasure

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class Test<T> {
2      T val;
3      public void test(T p) {
4      }
5  }
```

Type erasure

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public class Test<T> {
2      T val;
3      public void test(T p) {
4      }
5  }
```

```
1  public class Test<Object> {
2      Object val;
3      public void test(Object p) {
4      }
5  }
```

Generics!

> „**Generics —** You think you understand it, and then you don't. . .
> . . . until you eventually do.“