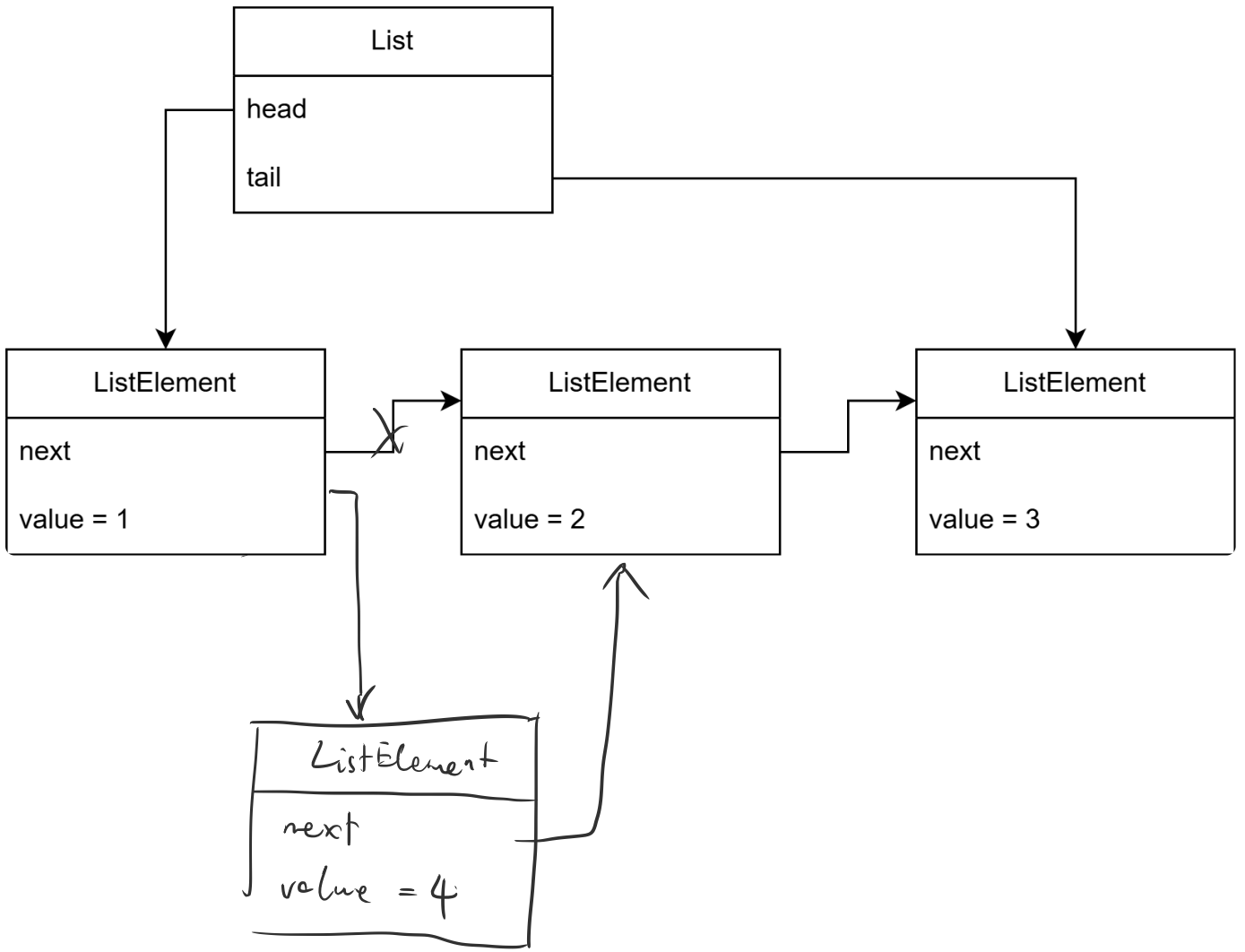
Woche 06

Listen und Vererbung Part I

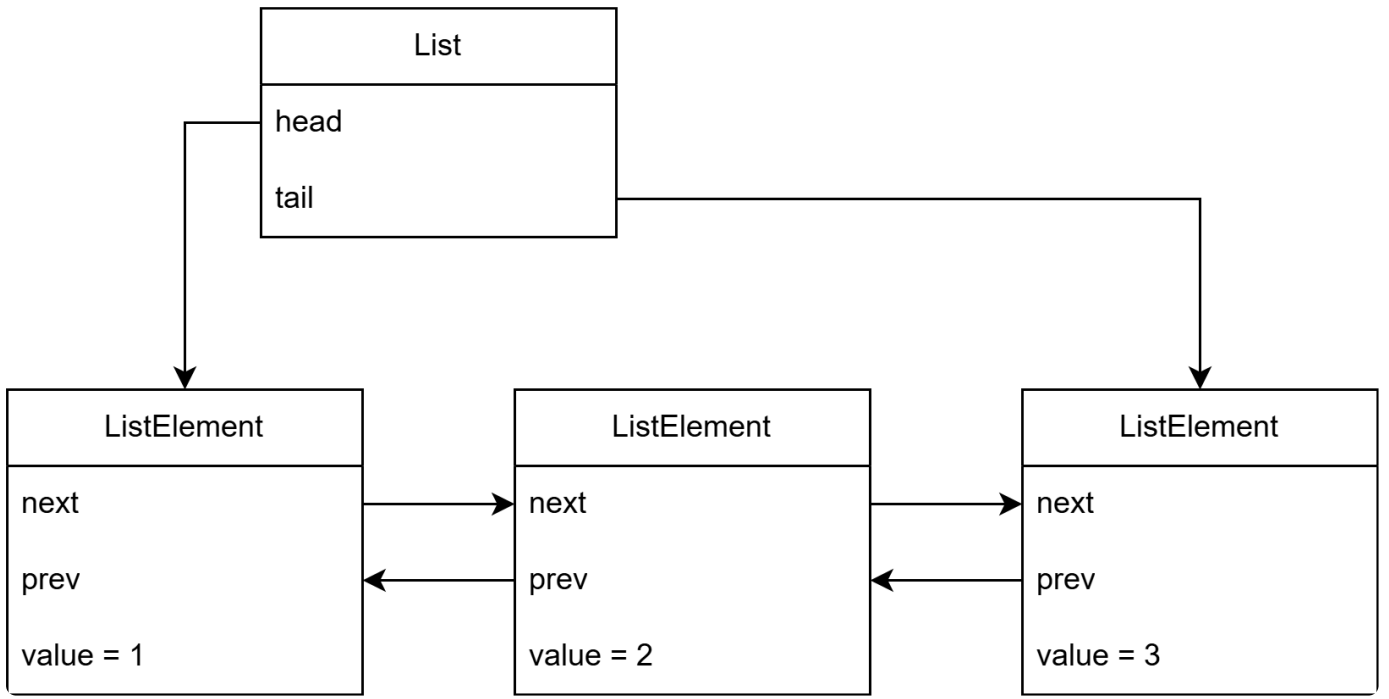
Einfach verkettete Liste



Wichtige Methoden:

- add(e) / add(index, e)
- remove(e) / remove(index)
- size()
- get(index)
- contains(e)

Doppelt verkettete Liste

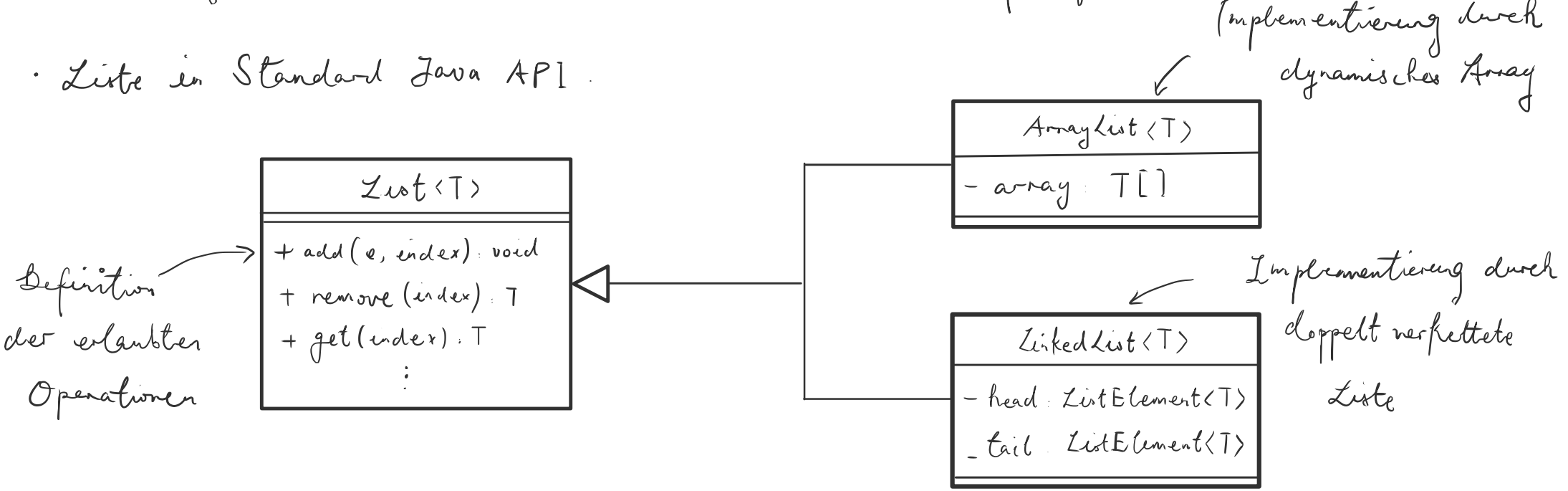


Wichtige Methoden:

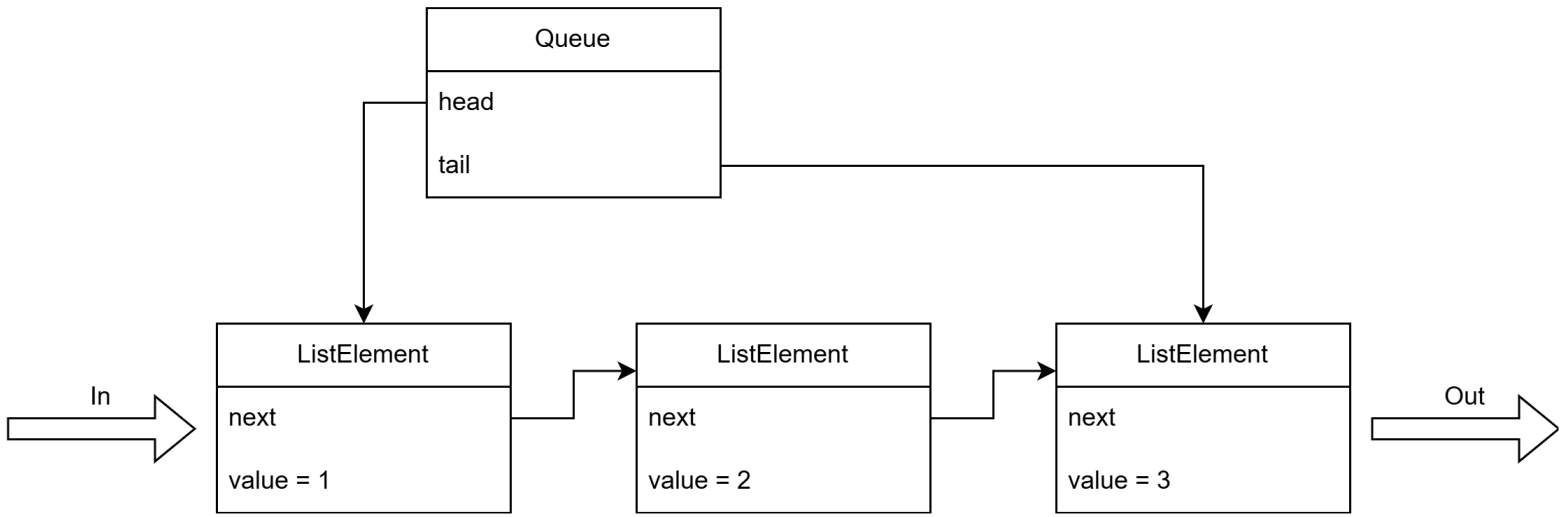
- add(e) / add(index, e)
- remove(e) / remove(index)
- size()
- get(index)
- contains(e)

Liste als abstrakter Datentyp

- Abstrakter Datentyp: definiert über zulässige Operationen, die interne Implementierung ist nicht relevant
- Liste: eine lineare Datenstruktur, die die Operationen add(e[,index]), remove(e|index), get(index), size(), contains(e), ... zur Verfügung stellt
- Liste in Standard Java API



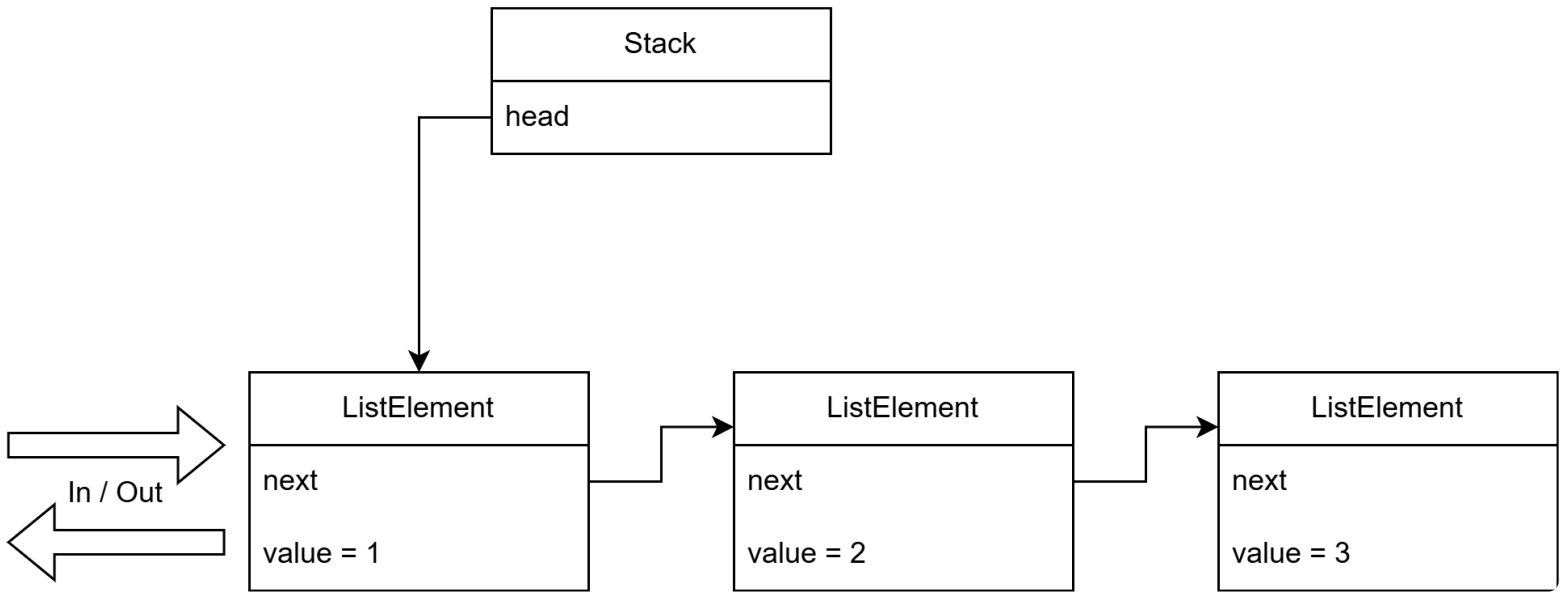
Queue



Wichtige Methoden:

- add(e)
- peek() / poll()
- size()

Stack



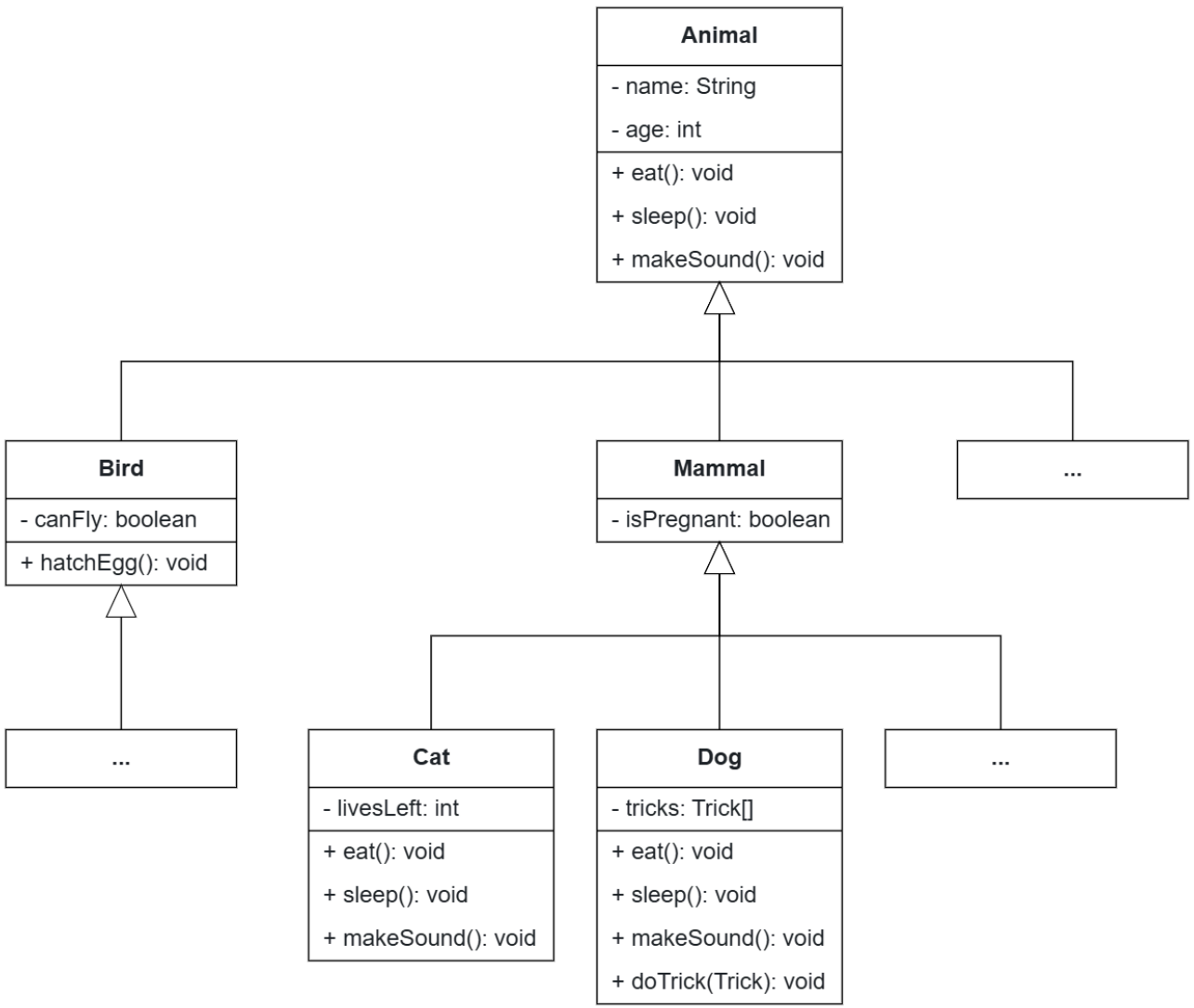
Wichtige Methoden:

- push(e)
- peek() / pop()
- size()

W06P02 - Pinguin Parade

Bearbeite nun die Aufgabe W06P02 - Pinguin Parade

Vererbung Part I

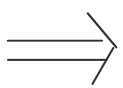


Vererbung

- Code Reuse

```
public class Dog {
    public void eat(){
        ...
    }
}
```

```
public class Cat {
    public void eat(){
        ...
    }
}
```



```
public class Animal {
    public void eat(){
        ...
    }
}
```

```
public class Dog extends Animal{
    ...
}
```

```
public class Cat extends Animal{
    ...
}
```

Vererbung

- Overriding / Überschreiben

```
public class Animal {

    public void eat(){
        System.out.println("Eating food");
    }
}
```

```
public class Dog extends Animal{
    @Override
    public void eat(){
        System.out.println("Eating bones");
    }
}
```

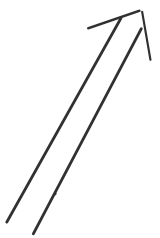
```
public class Cat extends Animal{
    @Override
    public void eat(){
        System.out.println("Eating fish");
    }
}
```

Vererbung

- Oberklasse als „Verallgemeinerung" — man muss sich nicht auf eine konkrete Unterklasse festlegen

```
public static void feed(Dog dog){
    dog.eat();
}
```

```
public static void feed(Cat cat){
    cat.eat();
}
```



```
public static void feed(Animal animal){
    animal.eat();
}
```

Bsp.

```
public static void main(String[] args){
    feed(new Dog());
}
```

```
Eating Bones
```

Vererbung Part I: Overriding

```java
public class A {

    private String s;

    public A(String s) {
        this.s = s;
    }

    public void doSomething() {
        System.out.println("A: " + s);
    }

    public String getS() {
        return s;
    }

    public static void main(String[] args) {
        A a = new A("objA");
        B b = new B("objB");

        a.doSomething();
        b.doSomething();
    }
}
```

```java
public class B extends A {

    public B(String s) {
        super(s);
    }

    @Override
    public void doSomething() {
        System.out.println("B: " + getS());
    }
}
```

Vererbung Part I: Overriding

```java
public class A {

    private String s;

    public A(String s) {
        this.s = s;
    }

    public void doSomething() {
        System.out.println("A: " + s);
    }

    public String getS() {
        return s;
    }

    public static void main(String[] args) {
        A a = new A("objA");
        B b = new B("objB");

        a.doSomething();
        b.doSomething();
    }
}
```

```java
public class B extends A {

    public B(String s) {
        super(s);
    }

    @Override
    public void doSomething() {
        System.out.println("B: " + getS());
    }
}
```

Output

```
A: objA
B: objB
```

Vererbung Part I: Shadowing

```java
public class A {

    public String s;

    public A(String s) {
        this.s = s;
    }

    public void doSomething() {
        System.out.println("A: " + s);
    }

    public String getS() {
        return s;
    }

    public static void main(String[] args) {
        A a = new A("objA");
        B b = new B("objB");

        a.doSomething();
        b.doSomething();
    }
}
```

```java
public class B extends A {

    public String s;

    public B(String s) {
        super(s);
        this.s = " hehe";
    }

    @Override
    public void doSomething() {
        System.out.println("B: " + getS() + s);
    }
}
```

Vererbung Part I: Shadowing

```java
public class A {

    public String s;

    public A(String s) {
        this.s = s;
    }

    public void doSomething() {
        System.out.println("A: " + s);
    }

    public String getS() {
        return s;
    }

    public static void main(String[] args) {
        A a = new A("objA");
        B b = new B("objB");

        a.doSomething();
        b.doSomething();
    }
}
```

```java
public class B extends A {

    public String s;

    public B(String s) {
        super(s);
        this.s = " hehe";
    }

    @Override
    public void doSomething() {
        System.out.println("B: " + getS() + s);
    }
}
```

Output

```
A: objA
B: objB hehe
```

Vererbung Part I: instanceof

```
1  public class A {}
2
3  public class B extends A {
4      public void methodB() {}
5
6      public static void main(String[] args) {
7          A a = new B();
8
9          a.methodB(); // BUG!!
10     }
11 }
```

Vererbung Part I: instanceof

```java
public class A {}

public class B extends A {
    public void methodB() {}

    public static void main(String[] args) {
        A a = new B();

        ((B) a).methodB(); // May be buggy!!
    }
}
```

Vererbung Part I: instanceof

```java
public class A {}

public class B extends A {
    public void methodB() {}
    public static void main(String[] args) {
        A a = new B();

        // Option 1
        if (a instanceof B) {
            B b = (B) a;
            b.methodB();
        }

        // Option 2
        if (a instanceof B b)
            b.methodB();
    }
}
```

W06P03 - Studenten-Zoo

Bearbeite nun die Aufgabe W06P03 - Studenten-Zoo