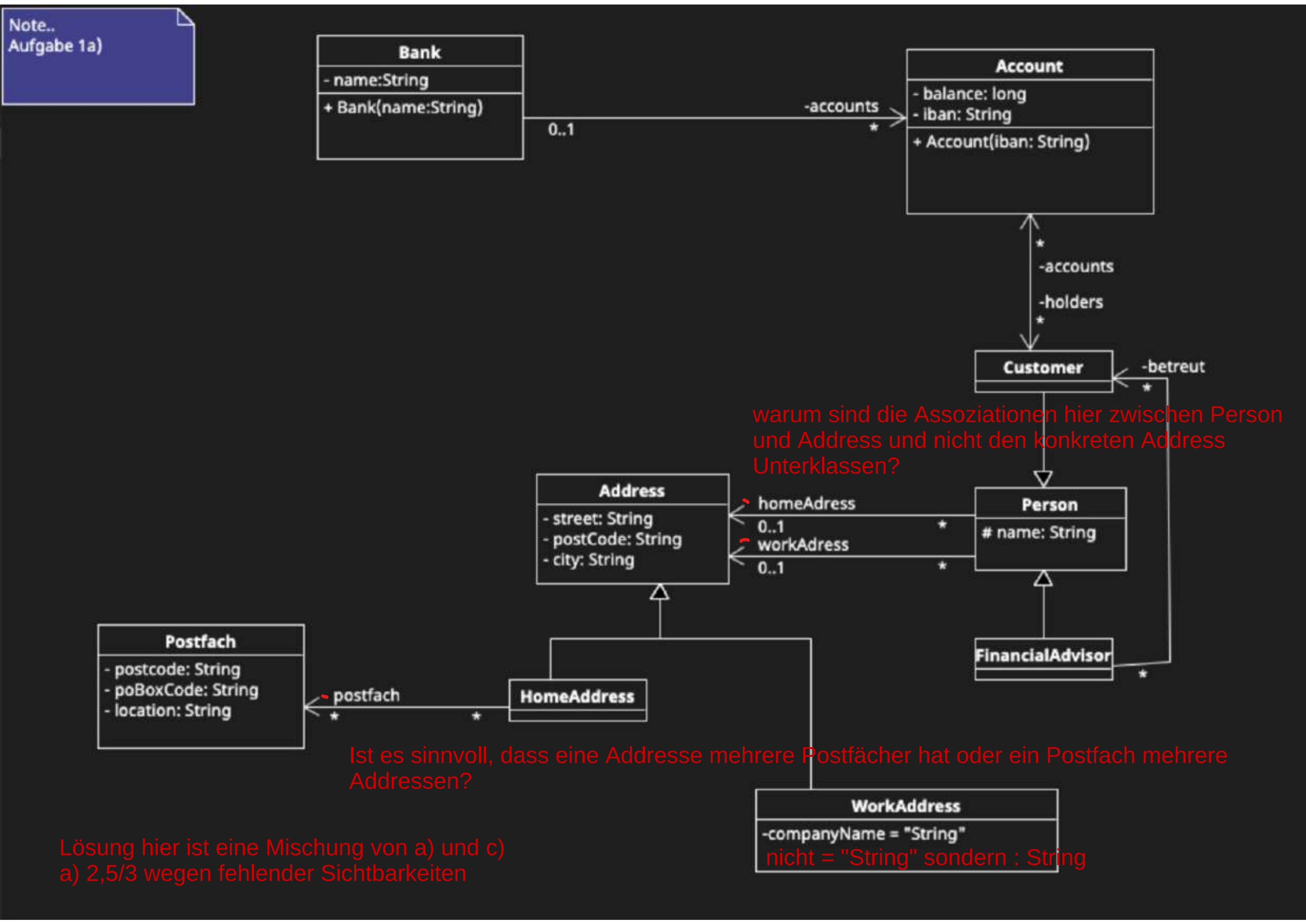
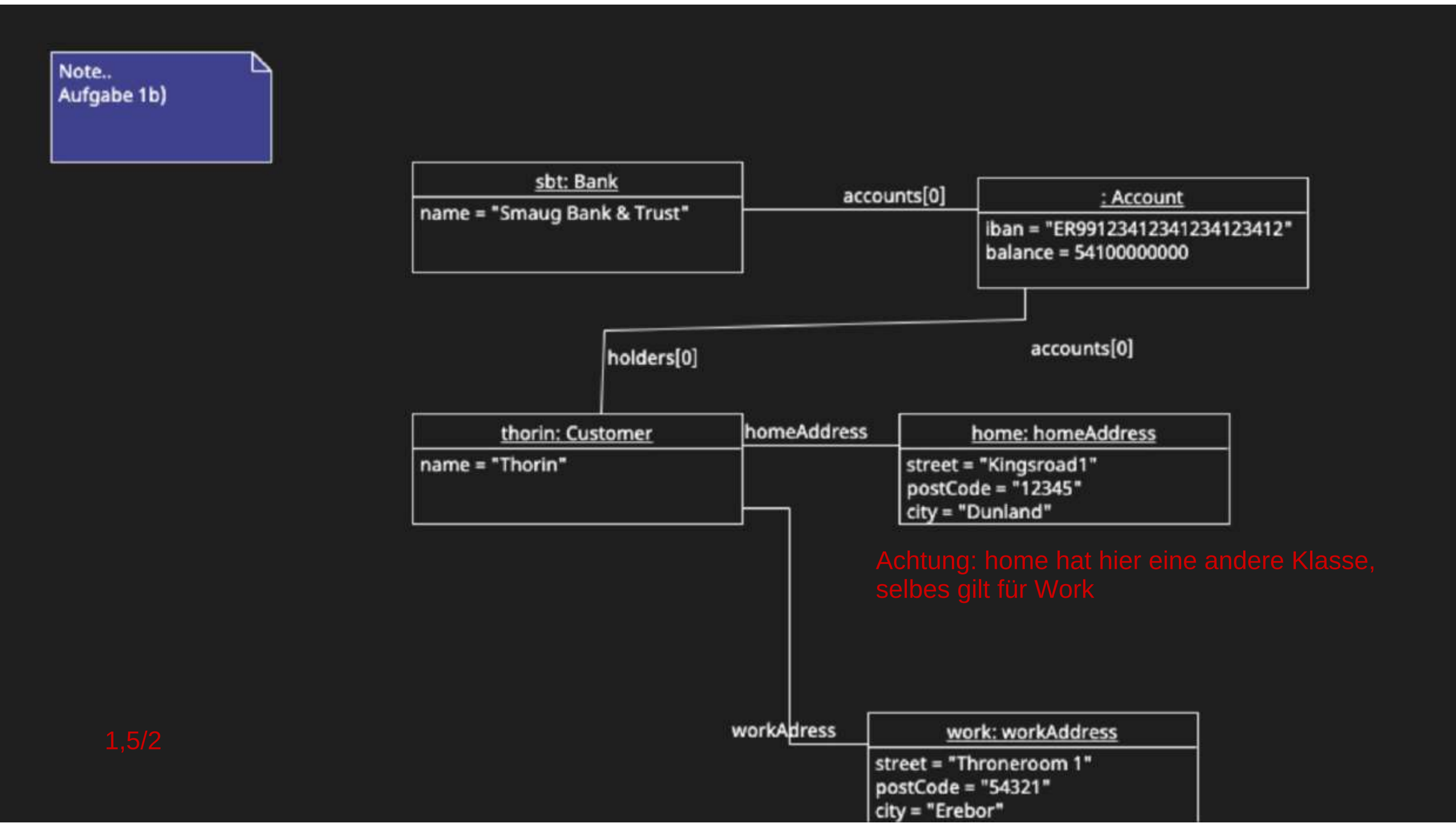


warum sind die Assoziationen hier zwischen Person und Address und nicht den konkreten Address Unterklassen?

Ist es sinnvoll, dass eine Addresse mehrere Postfächer hat oder ein Postfach mehrere Addressen?

nicht = "String" sondern : String

Lösung hier ist eine Mischung von a) und c)
a) 2,5/3 wegen fehlender Sichtbarkeiten



Achtung: home hat hier eine andere Klasse,
selbes gilt für Work

1,5/2

```java
public class Bank {

    private String name;
    private Account[] accounts;

    public Bank(String name) { this.name = name; }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public Account[] getAccounts() { return accounts; }

    public void setAccounts(Account[] accounts) { this.accounts = accounts; }

}

```

```java
public class Person {
    private String name;
    private Address homeAddress;
    private Address workAddress;

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }
    public Address getHomeAddress() { return homeAddress; }

    public void setHomeAddress(Address homeAddress) { this.homeAddress = homeAddress; }

    public Address getWorkAddress() { return workAddress; }

    public void setWorkAddress(Address workAddress) { this.workAddress = workAddress; }
}


```

```
public class Account {

    private Customer[] holders;
    private long balance;
    private String iban;

    public Account(String iban) { this.iban = iban; }

    public Customer[] getHolders() { return holders; }

    public void setHolders(Customer[] holders) { this.holders = holders; }

    public long getBalance() { return balance; }

    public void setBalance(long balance) { this.balance = balance; }

    public String getIban() { return iban; }

}

```

```java
public class Address {

    private String street;
    private String postCode;
    private String city;

    public String getStreet() { return street; }

    public void setStreet(String street) { this.street = street; }

    public String getPostCode() { return postCode; }

    public void setPostCode(String postCode) { this.postCode = postCode; }

    public String getCity() { return city; }

    public void setCity(String city) { this.city = city; }

}
```

```java
public class Banking {

    public static void main(String[] args) {
        Bank sbt = new Bank("Smaug Bank & Trust");
        sbt.setAccounts(new Account[1]);
        sbt.getAccounts()[0] = new Account("ER99123412341234123412");
        sbt.getAccounts()[0].setBalance(54100000000L);
        Customer thorin = new Customer();
        thorin.setAccounts(new Account[1]);
        thorin.getAccounts()[0] = sbt.getAccounts()[0];
        thorin.setName("Thorin");
        Address home = new Address();
        home.setStreet("Kingsroad 1");
        home.setPostCode("12345");
        home.setCity("Dunland");
        thorin.setHomeAddress(home);
        Address work = new Address();
        work.setStreet("Throneroom 1");
        work.setPostCode("54321");
        work.setCity("Erebor");
        thorin.setWorkAddress(work);
        sbt.getAccounts()[0].setHolders(new Customer[] { thorin });
    }

}
```

```java
public class Customer extends Person {


    private Account[] accounts;




    public Account[] getAccounts() { return accounts; }

    public void setAccounts(Account[] accounts) { this.accounts = accounts; }



}

```

Namenskonvention: Postfach

```java
public class postfach {
    private String postcode;
    private String poBoxCode;

    private String location;
}

```

```
public class HomeAddress extends Address{
    private postfach[] postfach;
}

```

```java
public class WorkAddress extends Address{
    private String companyName;
}

```

```java
public class FinancialAdviser {
    private Customer[] betreut;
}

```

c) 4/5

```java
// Press ⇧ twice to open the Search Everywhere dialog and type `show whitespaces`,
// then press Enter. You can now see whitespace characters in your code.
public class Main {
    public static void main(String[] args) {
        // Press ⌥⏎ with your caret at the highlighted text to see how
        // IntelliJ IDEA suggests fixing it.
        System.out.printf("Hello and welcome!");
        VersatileLinkedList h = new VersatileLinkedList();
        VersatileLinkedList b = new VersatileLinkedList();
        VersatileLinkedList c = new VersatileLinkedList();
        h.add(6);
        h.add(7);
        b.add(8);
        b.add(9);
        b.add(7);
        b.add("a");
        h.add(1,2,b);
        h.add("b");

        c = h.reverse();

        c.equals(c);
        System.out.print(c.get(3));



        // Press ^R or click the green arrow button in the gutter to run the code.

    }
}
```

was bedeuten diese Kommentare?

vielleicht noch ein paar Fälle für equals testen
2/2

```java
public class LinkedStringList {

    private LinkedStringListElement start;

    public int size() {
        int result = 0;
        LinkedStringListElement tmp = start;
        while (tmp != null) {
            tmp = tmp.getNext();
            result++;
        }
        return result;
    }

    public String get(int index) {
        if (start == null) { // list is empty
            return null;
        }
        LinkedStringListElement current = start;
        int pos = 0; // counter for finding the right position
        while (pos < index) {
            if (current.getNext() == null) {
                // list does not have enough elements
                return null;
            }
            current = current.getNext();
            pos++;
        }
        return current.getValue();
    }

    public void add(String value) {
        LinkedStringListElement elem = new LinkedStringListElement();
        elem.setValue(value);
        if (start == null) { // list is empty
            start = elem;
        } else {
            LinkedStringListElement tmp = start;
            while (tmp.getNext() != null) { // find last element
                tmp = tmp.getNext();
            }
            tmp.setNext(elem);
        }
    }

    public String remove(int index) {
        if (start == null) { // list is empty
            return null;
        }
        if (index == 0) { // remove from the beginning of non-empty list
            String result = start.getValue();
            start = start.getNext();
            return result;
        }
        // remove from anywhere in a non-empty list
        LinkedStringListElement current = start;
        int pos = 0; // counter for finding the right position
        while (pos < index - 1) {
            if (current.getNext() == null) {
                // list does not have enough elements
                return null;
            }
            current = current.getNext();
            pos++;
        }
        if (current.getNext() == null) { // not enough elements
            return null;
        }
        String result = current.getNext().getValue();
        current.setNext(current.getNext().getNext());
        return result;
    }
```

```java

}
class LinkedStringListElement {

    private LinkedStringListElement next;
    private String value;

    public LinkedStringListElement getNext() {
        return next;
    }
    public void setNext(LinkedStringListElement next) {
        this.next = next;
    }
    public String getValue() {
        return value;
    }
    public void setValue(String value) {
        this.value = value;
    }

}
```

```
public class VersatileLinkedList extends LinkedStringList{
    public void add(int wert){
        String stringwert = Integer.toString(wert);
        this.add(stringwert);   a) 1/1
    }

    public void add(boolean wert){
        String stringwert;
        if(wert == true){
            stringwert = "yes";
        }

        else {
            stringwert = "no";
        }

        this.add(stringwert);
    }   b) 1/1

    public void add(LinkedStringList wert){
        for(int i = 0; i < wert.size(); i++){
            this.add(wert.get(i));
        }
    }  c) 1/1

    public void add(int start, int end, LinkedStringList list){
        for(int i = start; i <= end; i++){
            this.add(list.get(i));
        }
    }    Hier noch prüfen ob die Werte für start und end sinnvoll sind d) 1,5/2

    public VersatileLinkedList reverse(){
        VersatileLinkedList reverselist = new VersatileLinkedList();
        for(int i = this.size() - 1; i >= 0; i--) {
            reverselist.add(this.get(i));
        }

        return reverselist;   e) 2/2
```

```java
    }

    public boolean equals(VersatileLinkedList wert) {
        boolean result = false;
        for (int i = 0; i < this.size(); i++) {
            if (wert.get(i).equals(this.get(i))) {
                result = true;
            }
        }
        return result;
    }







}

```

was passiert wenn wert kürzer oder länger ist als this? 0,5/1