Composition and Aggregation

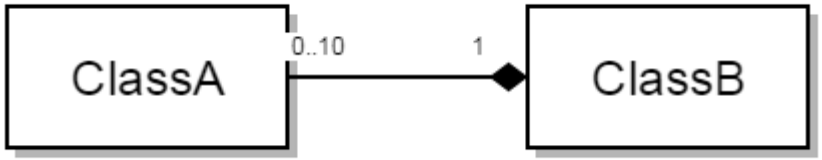
Review Slides

Remember

- In composition, the composing instances (objects) are owned by the other class, their existence depends on it
- In aggregation, the aggregated instances (objects) have an independent existence

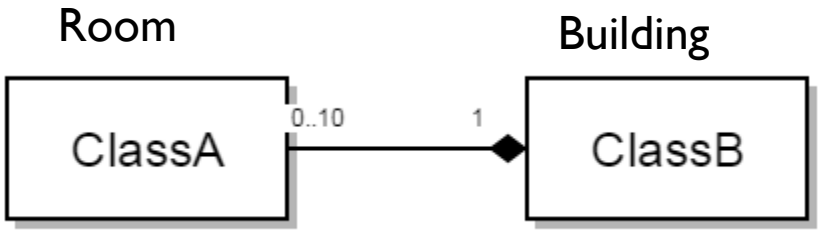
- Both represent a HAS-A relationship

Composition Example 1



ClassB is composed of at most 10 ClassA objects

Composition Example 1



Building is composed of at most 10 Room objects.

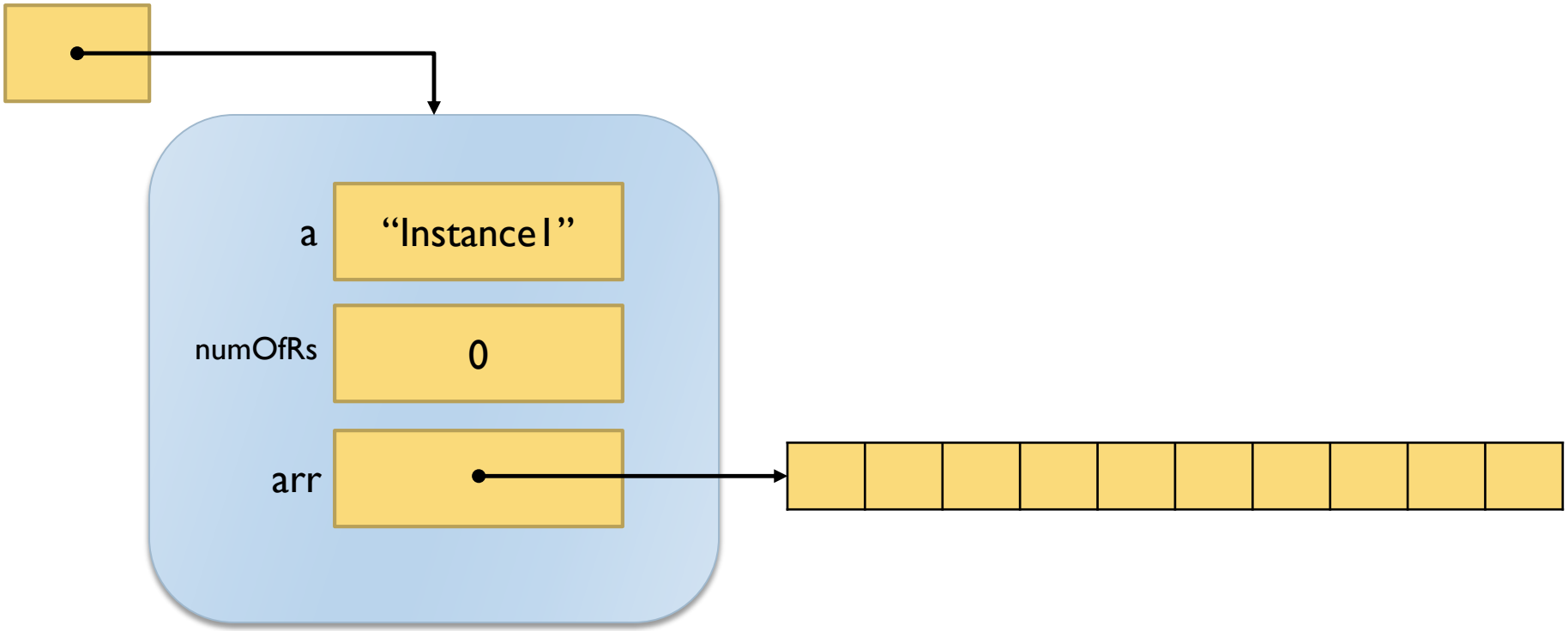
```
public class Building {
   String name;
   int numOfRooms;
   Room [] arr;


   public Building(String n)
   {
      //initialise all attributes
      this.name=n;
      numOfRooms=0;
      this.arr= new Room[10];
   }


   public Building(Building b)
   {
       copy(b);
   }
```

```
public void copy(Building b)
   {
      //copy primitive values directly
      name= b.name;
      numOfRooms= b.numOfRooms;

     //create a new array with same length as b's
      arr=new Room [b.arr.length];

     //assign its elements to copies of b's elements
      for (int i=0; i< numOfRooms; i++)
         arr[i]=new Room (b.arr[i]);
   }


public void add(Room r)
   {
     if (numOfRooms< arr.length)
        //add to the first empty element a copy of the
parameter
         arr[numOfRooms++]= new Room (r);
   }



}
```

Cont Comp. Ex. 1

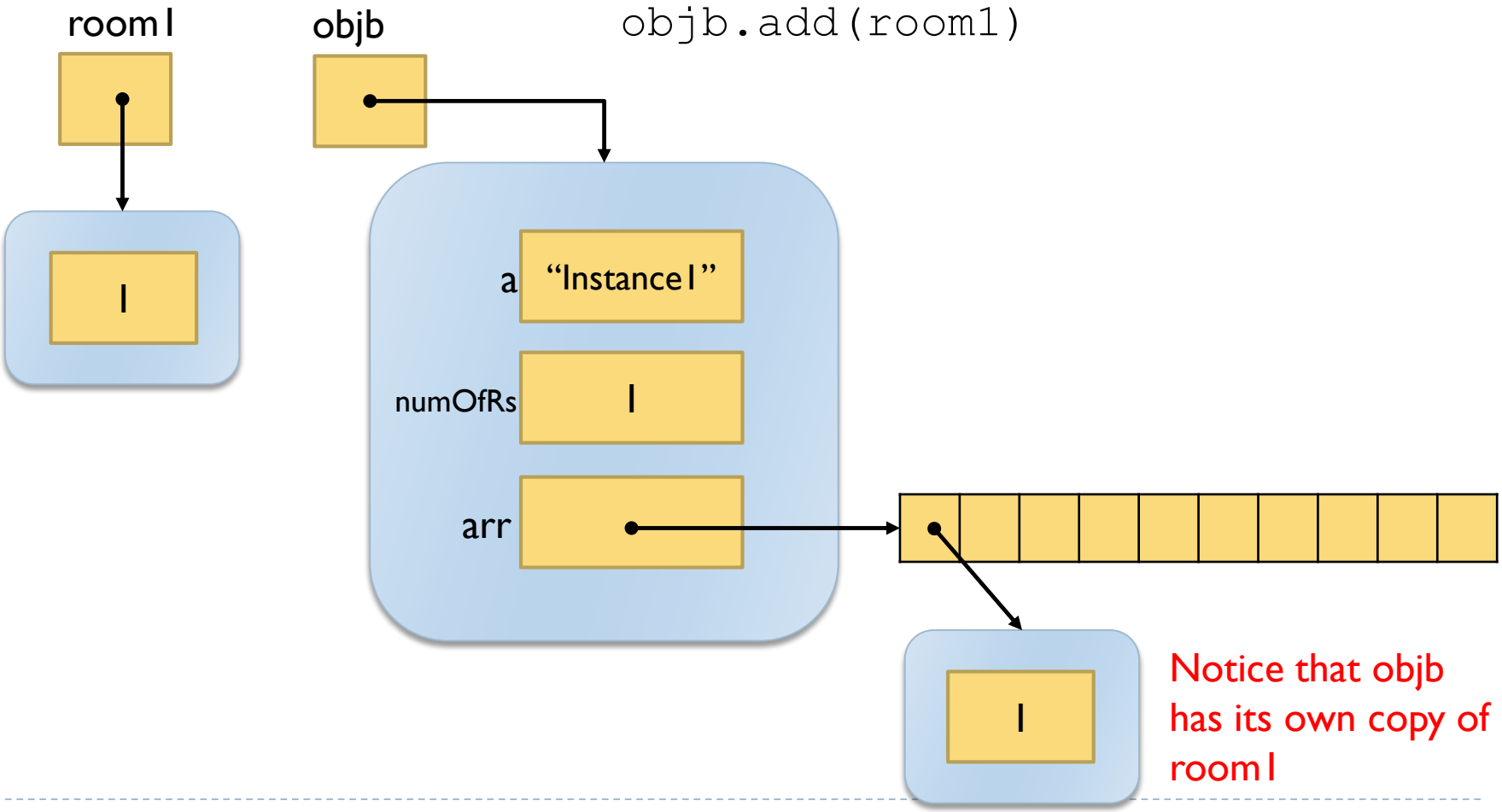
- Let main contain

```
Building objb= new Building("Instance1");
```



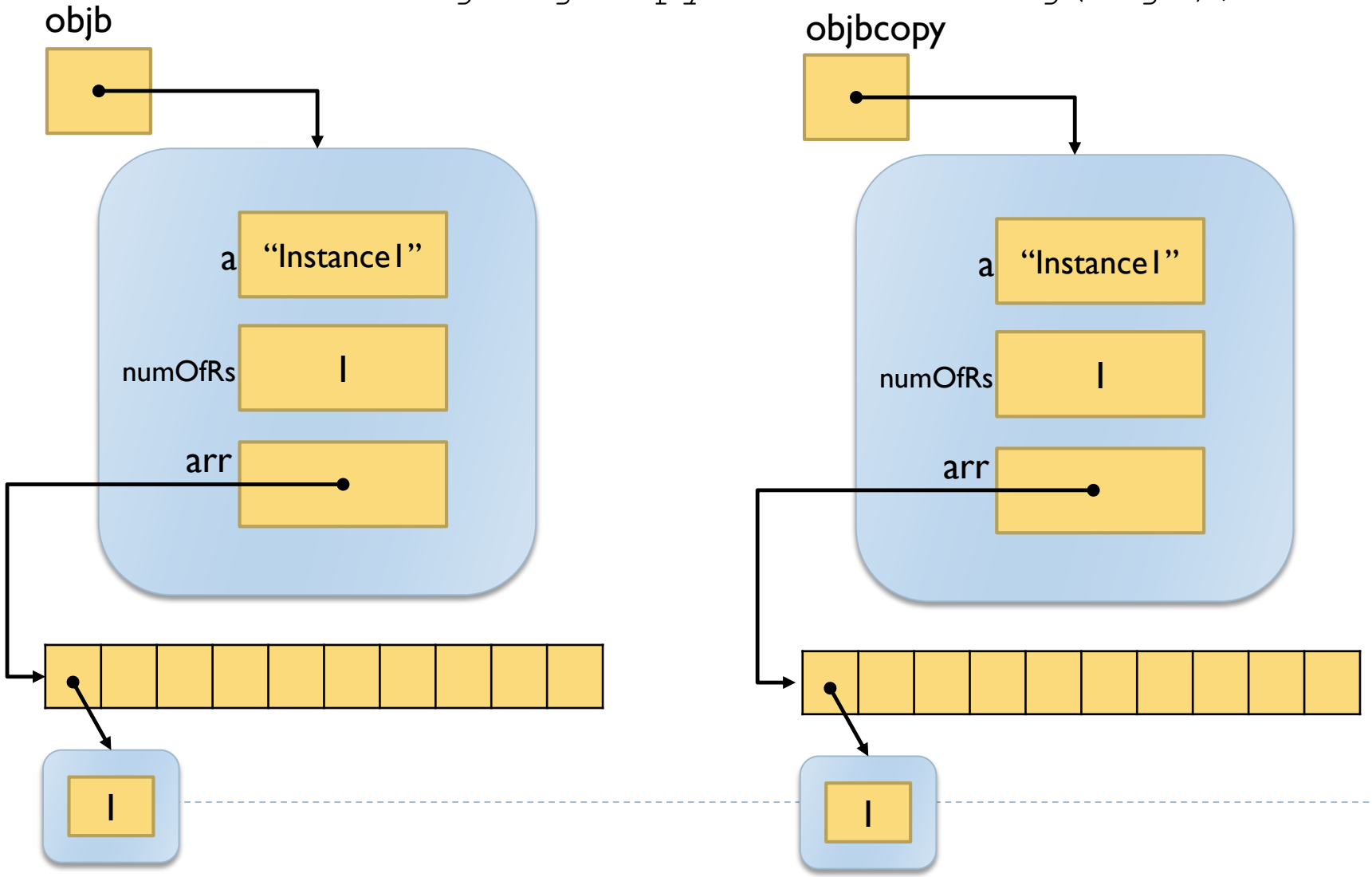
Cont Comp. Ex.1

```
Room room1= new Room(1);
objb.add(room1)
```



Cont Comp. Ex.1

```
Building objbcopy= new Building(objb);
```

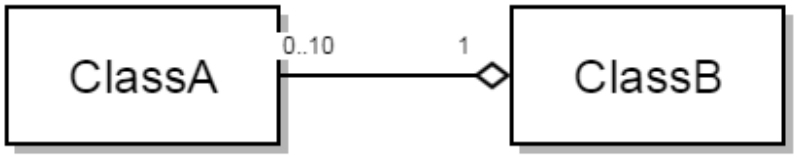


What's the difference between a copy constructor and method copy?

- Both receive an object of the same class and set the attributes to the same values or copies of the received object
- But:
  - A constructor is used to create a new object
  - You need an existing object to call method copy on
    - Example:

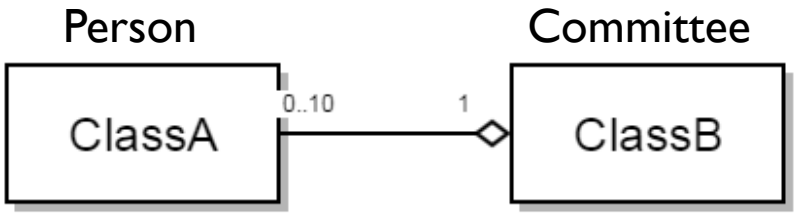
```
//objb was created in previous slides
ClassB objbcopy2= new ClassB("to be overwritten");
objcopy2.copy(objb);
```

Aggregation Example 1



ClassB is an aggregation of at most 10 ClassA objects

Aggregation Example 1



Committee is an aggregation of at most 10 Person objects

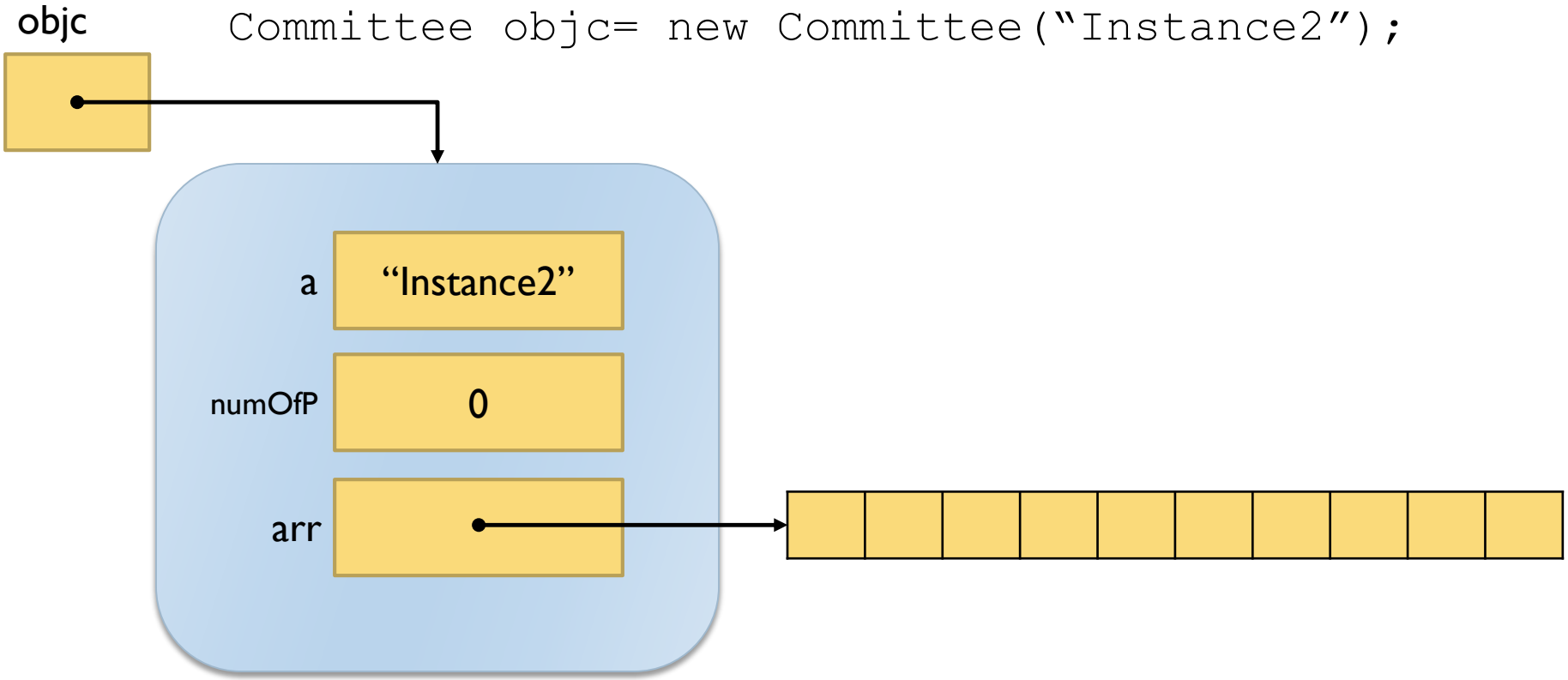
```
public class Committee {


    String name;
    int numOfP;
    Person[] arr;


    public Committee(String n)
    {
       //initialise all attributes
       this.name=n;
       numOfP=0;
       this.arr= new Person[10];
    }



    public Committee (Committee c)
    {
       copy(c);
    }
```

```
public void copy(Committee c)
    {
       //copy primitive values directly
       name= c.name;
       numOfP= c.numOfP;
       //create a new array with same length as c's
       //We do not want two objects to point at the
same array
       arr=new Person[c.arr.length];

       //assign its elements to references to b's
elements
       for (int i=0; i< numOfP; i++)
       {
          arr[i]=c.arr[i];
       }
    }


public void add(Person p)
    {
       if (numOfP < arr.length)
       {
          //add to the first empty element a reference
to the parameter
          arr[numOfP++]= p;
       }
    }


}//end class
```

Cont Agg. Ex. 1

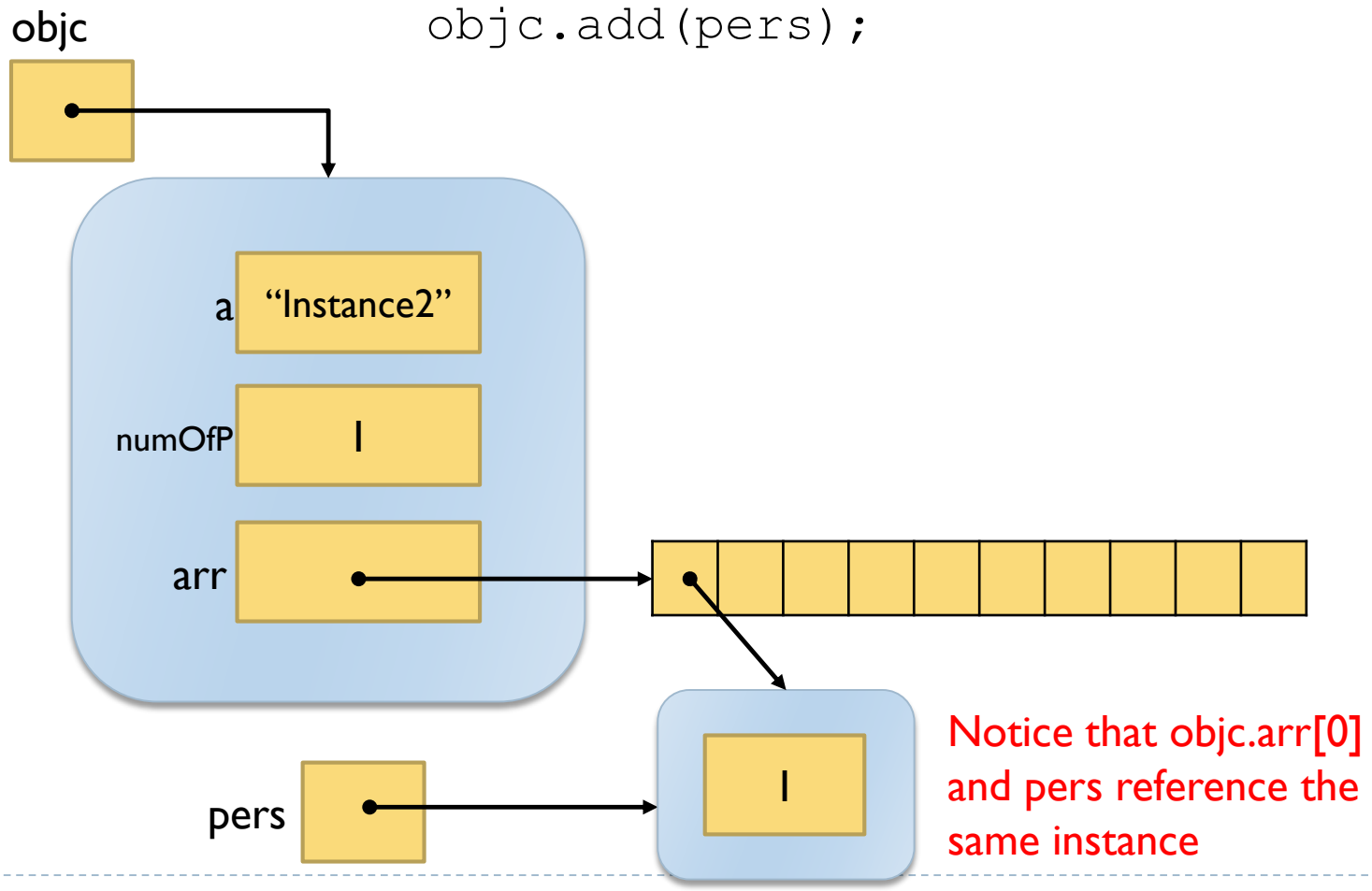
- Let main contain

```
Committee objc= new Committee("Instance2");
```



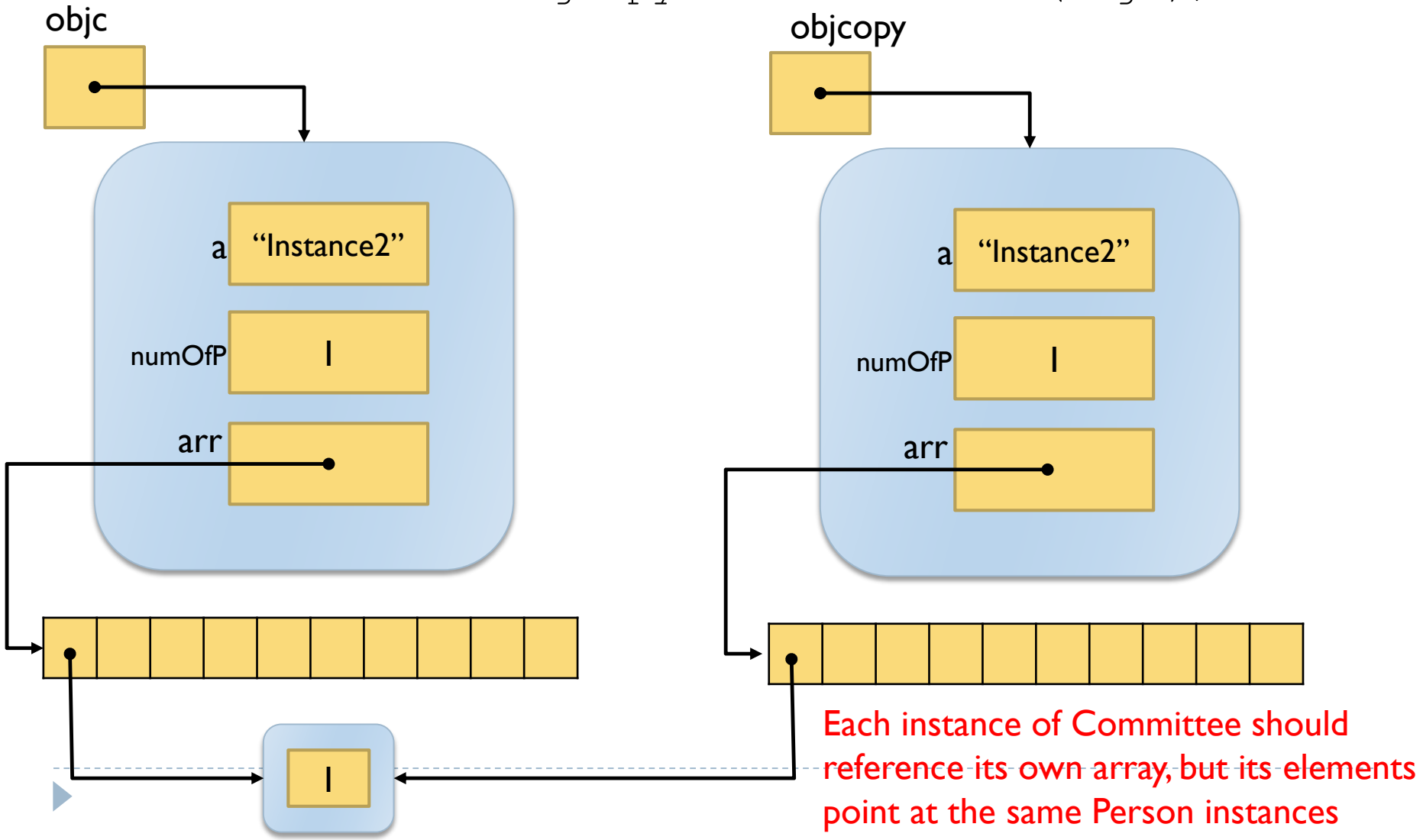
Cont Agg. Ex. 1

```
Person pers= new Person(1);
objc.add(pers);
```



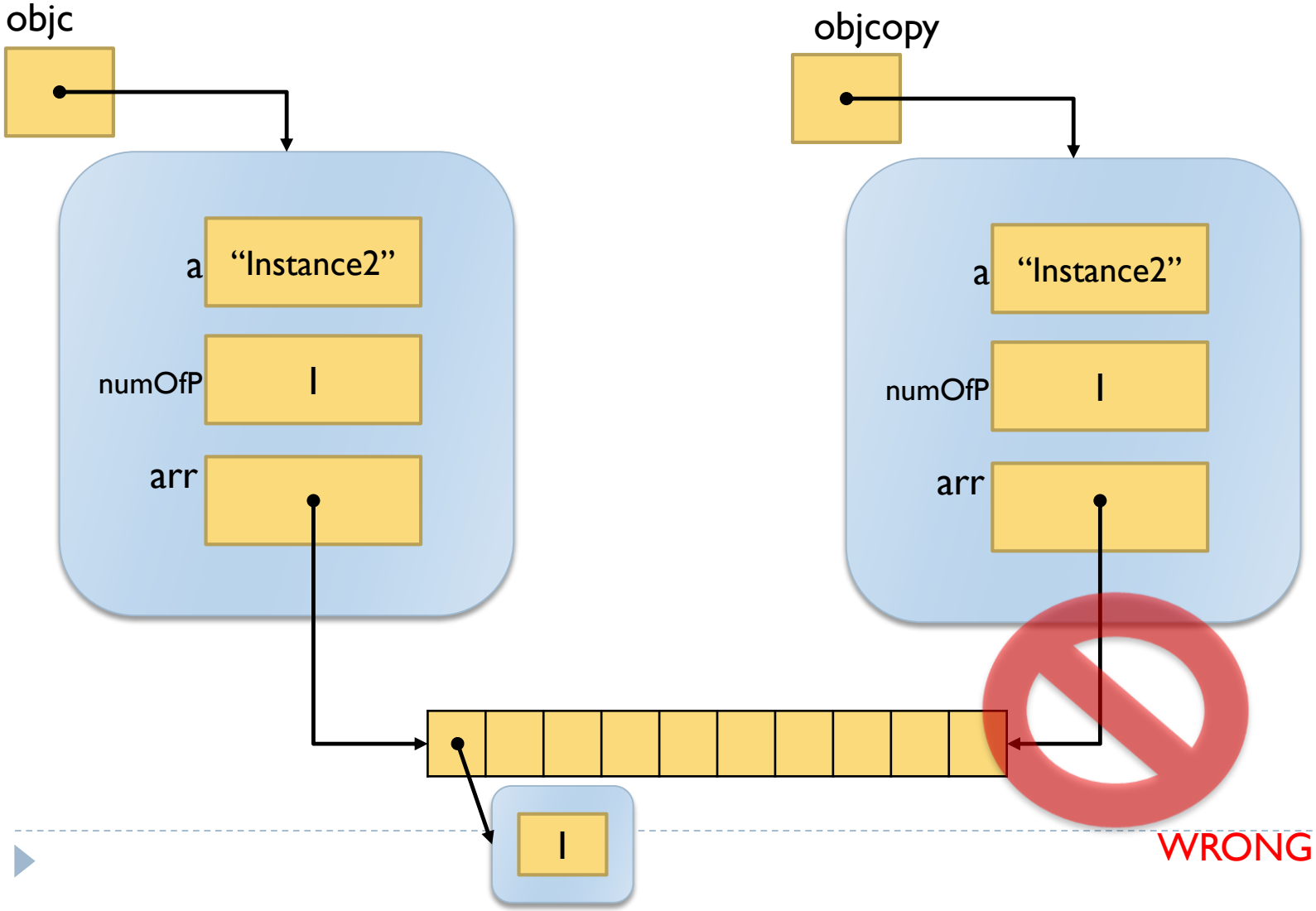
Cont Agg. Ex. 1

```
Committee objcopy= new Committee(objc);
```



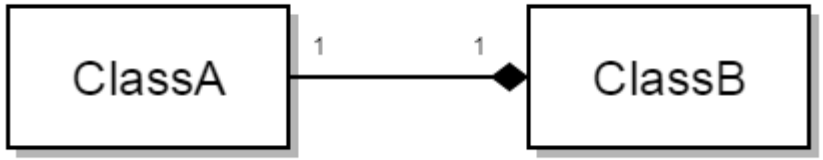
Cont Agg. Ex. 1

```
Committee objcopy= new Committee(objc);
```



Composition Example 2

- What if the association was one-to-one?

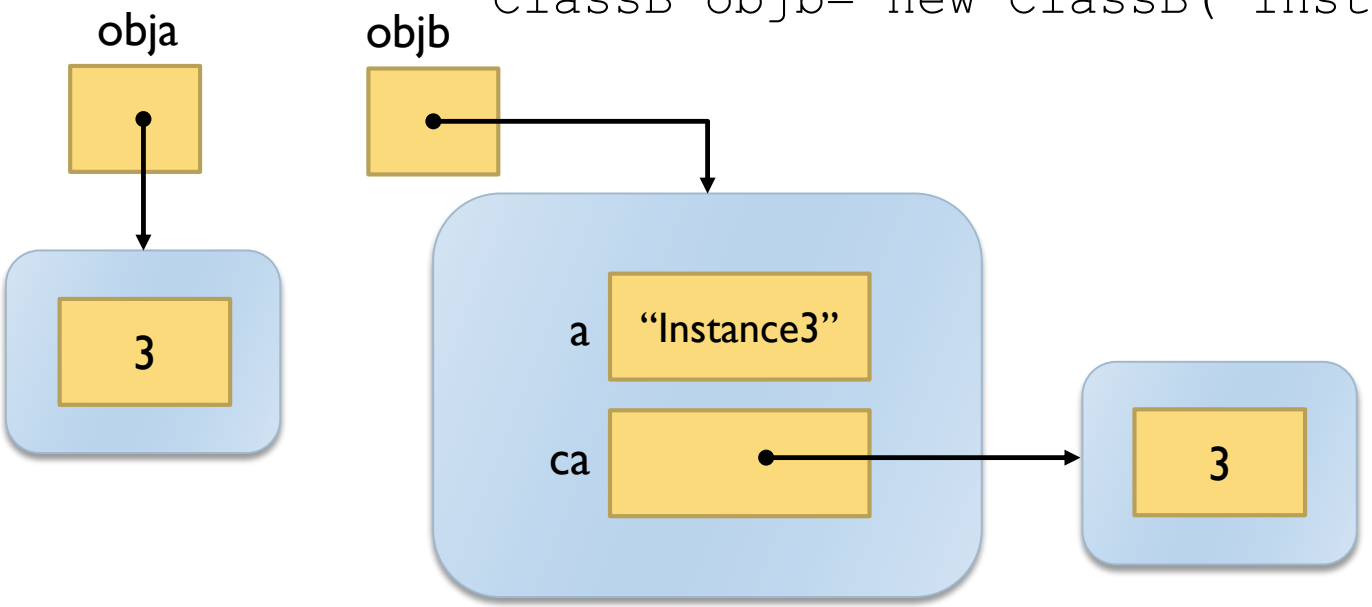


ClassB is composed of at most one ClassA objects

```
public class ClassB{

   String a;
   ClassA ca;

   public ClassB(String a, ClassA ca)
   {
      //initialise all attributes
      this.a=a;

      //a copy of the parameter must be assigned
      this.ca= new ClassA(ca);//OR setCA(b.ca)
   }


   public ClassB(ClassB b)
   {
      copy(b);
      //OR
      //this(b.a, b.ca);
   }

   public void copy(ClassB b)
      {
         //copy primitive values directly
         a= b.a;

         //reference values depend on association
         //need a copy for composition
         ca= new ClassA(b.ca); //OR setCA(b.ca)

      }

   public void setCA(ClassA ca)
      {
         this.ca= new ClassA(ca);
      }


}
```

Cont Comp. Ex.2

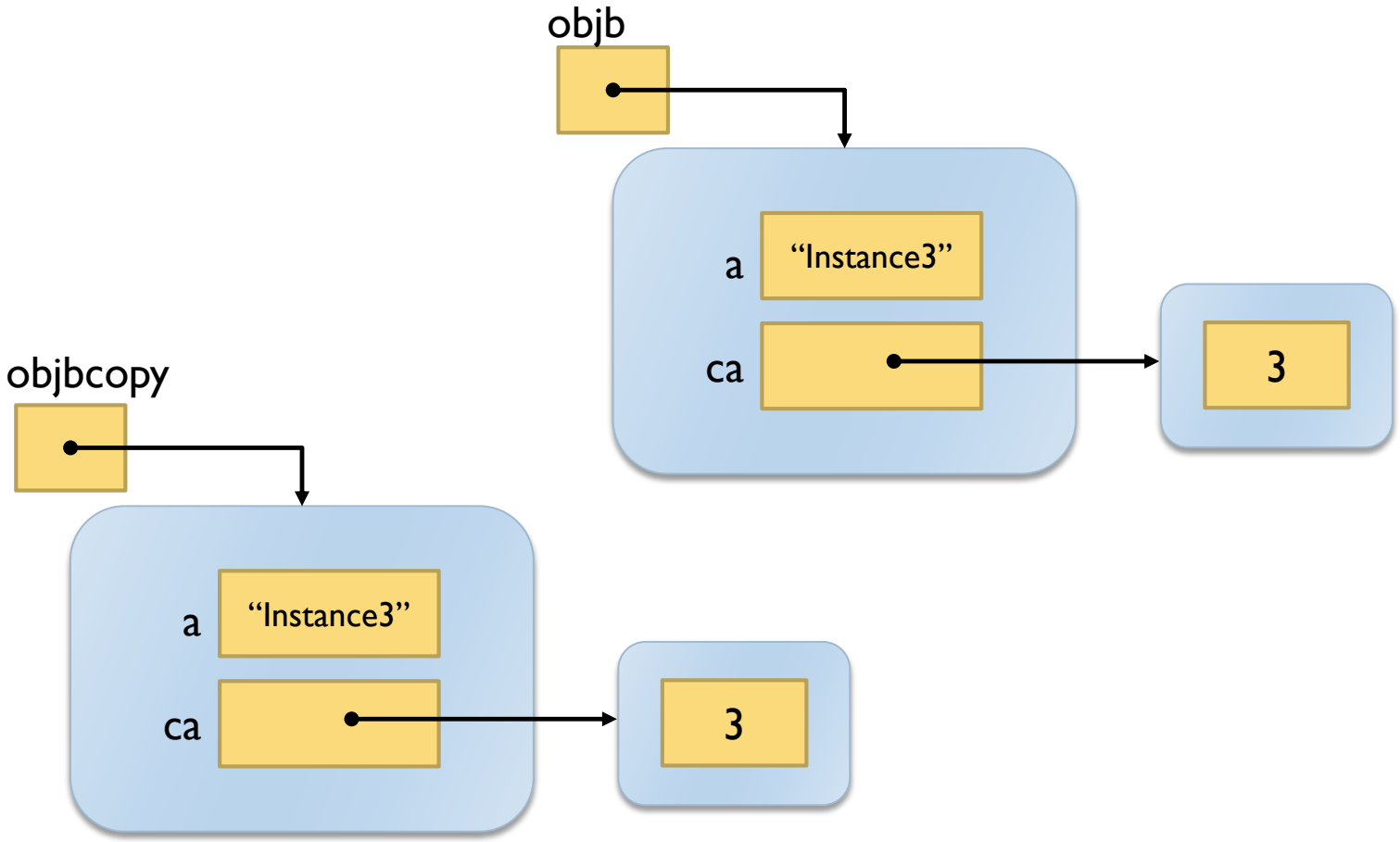
```
ClassA obja= new ClassA(3);
ClassB objb= new ClassB("Instance3", obja);
```



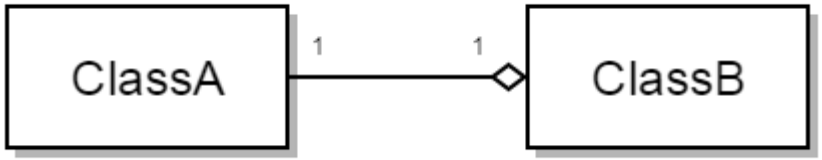
Notice that objb has its own copy of obja

Cont Comp. Ex.2

```
Classb objbcopy= new Classb(objb);
```



Aggregation Example 2



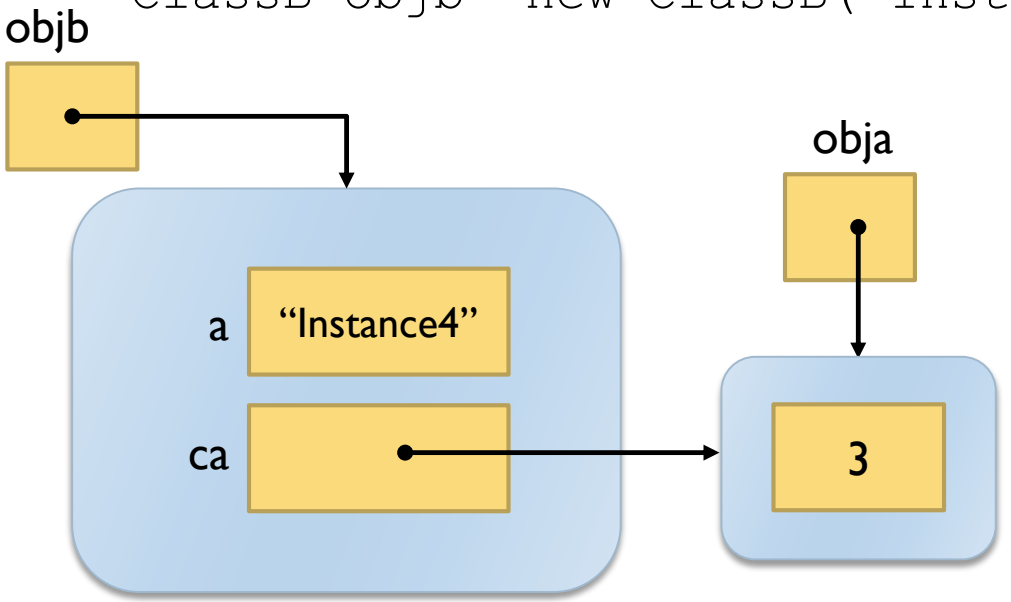
ClassB is an aggregation of one ClassA object

```
public class ClassB{

    String a;
    ClassA ca;

    public ClassB(String a, ClassA ca)
    {
        //initialise all attributes
        this.a=a;

        //a reference to the parameter is assigned
        this.ca= ca;//OR setCA(b.ca)
    }


    public ClassB(ClassB b)
    {
        copy(b);
        //OR
        //this(b.a, b.ca);
    }
```

```
    public void copy(ClassB b)
    {
        //copy primitive values directly
        a= b.a;

        //reference values depend on association
        //need a reference for aggregation
        ca= b.ca; //OR setCA(b.ca)
    }


    public void setCA(ClassA ca)
    {
        this.ca= ca;
    }



}
```

Cont Comp. Ex.2

```
ClassA obja= new ClassA(3);
ClassB objb= new ClassB("Instance4", obja);
```



Notice that objb references the same instance obja references

Cont Comp. Ex.2

```
Classb objbcopy= new Classb(objb);
```