```java
package uebung041;

abstract class Filter implements Sequence{
    protected Sequence zahlenquelle;
    public Filter(Sequence zahlenquelle){
        this.zahlenquelle = zahlenquelle;

    }
}
```

✓ 1/1

```java
package uebung041;

public class Primes extends Filter{

    public Primes(){
        super(new Naturals());
    }
    //Wie kann man die 1 bei den nur mit den gegeben Methoden löschen???
    @Override
    public int getNext() {
        return 0;
    }
}

```

im Constructor nach super zahlenquelle.getNext() aufrufen

1/3

```java
package uebung041;

public class Naturals implements Sequence{
    private int current = 1;

    public void setCurrent(int current) {
        this.current = current;
    }

    public int getNext() {
        int result = current;
        current++;
        return result;
    }
}

```

1/1

```java
package uebung041;

public interface Sequence {
    int getNext();
}

```

1/1

```java
package uebung041;

public class ZapMultiples extends Filter{

    private int basis;

    public ZapMultiples(int basis, Sequence zahlenquelle){
        super(zahlenquelle);
        this.basis = basis;
    }

    @Override
    public int getNext() {
        int result = zahlenquelle.getNext();
        while(result % basis == 0){
            result = zahlenquelle.getNext();
        }
        return result;
    }
}
```

2/2

```java
package uebung044;

public class Desk extends Table{

    public Desk(){
        super();
    }
}

```

```java
package uebung044;

import java.util.ArrayList;
import java.util.List;

public class Room {
    protected List<Furniture> furniture = new ArrayList<>();

    public Room(){

    }
    public Room(List<Furniture> furniture) {
        this.furniture = furniture;
    }

    public List<Furniture> getFurniture() {
        return furniture;
    }

    public void setFurniture(List<Furniture> furniture) {
        this.furniture = furniture;
    }
}

```

```java
package uebung044;

public class test {
    public static void main(String[] args) {
        Chair chair0 = new Chair();
        Chair chair1 = new Chair();

        Desk desk0 = new Desk();

        Office office = new Office(chair0, desk0);

        office.furniture.add(chair0);
        office.furniture.add(desk0);
        office.furniture.add(chair1);



    }
}

```

4/4

```java
package uebung044;

public class Chair implements Furniture{


    private String name;

    public Chair(){

    }
    public String getName() {
        return name;
    }

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

```

warum hat der Stuhl einen Namen?

```java
package uebung044;

public class Table implements Furniture{
    public Table() {

    }
}

```

```java
package uebung044;

import java.util.ArrayList;
import java.util.List;

public class Office extends Room {
    private List<Chair> chair;
    private List<Desk> desk;

    public Office(Chair chairnew, Desk newdesk) {
        chair = new ArrayList<>();
        chair.add(chairnew);
        desk = new ArrayList<>();
        desk.add(newdesk);
    }

    public List<Chair> getChair() {
        return chair;
    }
}

```

warum nicht beide Listen per Getter zur Verfügung stellen und dann über diese Einfügen?

```java
package uebung044;

public interface Furniture {

}

```