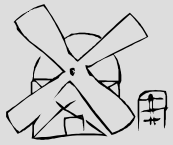
marc-oliver pahl



Die ObjektOrientierte Mühle

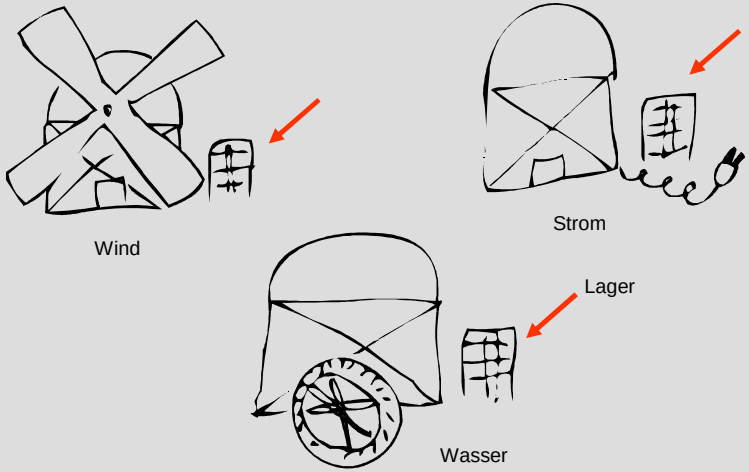
Das Beispiel soll noch einmal das Konzept der Objektorientiertheit erläutern.

Dabei werden außerdem Interfaces, Vererbung, abstrakte Methoden, Queues, SimplyLinked Lists, Exceptions, ... verwendet.

9.2.2004

verschiedene Muehlen

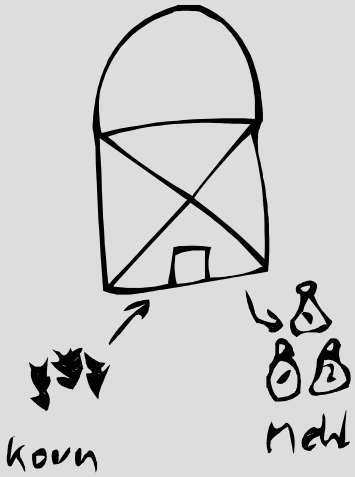
marc-oliver pahl



Muehle

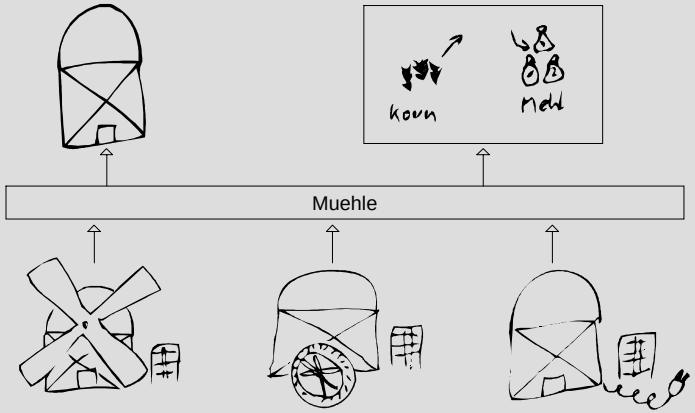
marc-oliver pahl

```
public interface IMuehle {

    public Id anliefern(Korn korn);

    public void mahlen();

    public MehlSack abholen();
}
```



-> UML

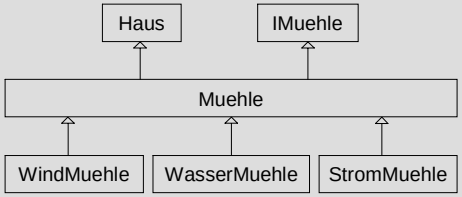
marc-oliver pahl



UML-Vererbungs-Diagramm

5

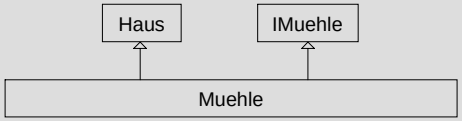
marc-oliver pahl



Muehle.java

6

marc-oliver pahl

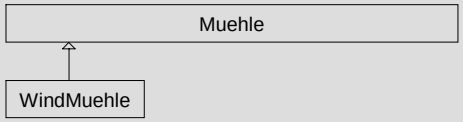



```
public abstract class Muehle extends Haus implements IMuehle {
   protected LagerQueue kornlager;  protected LagerQueue mehllager;

   public Id anliefern(Korn korn){
      new Id(); kornlager.push(theId);
      return theId;
   }

   public abstract void mahlen() throws MahlException;

   public MehlSack abholen(){
      return (MehlSack) mehllager.pop();
   }
}
```

WindMuehle.java

7

marc-oliver pahl

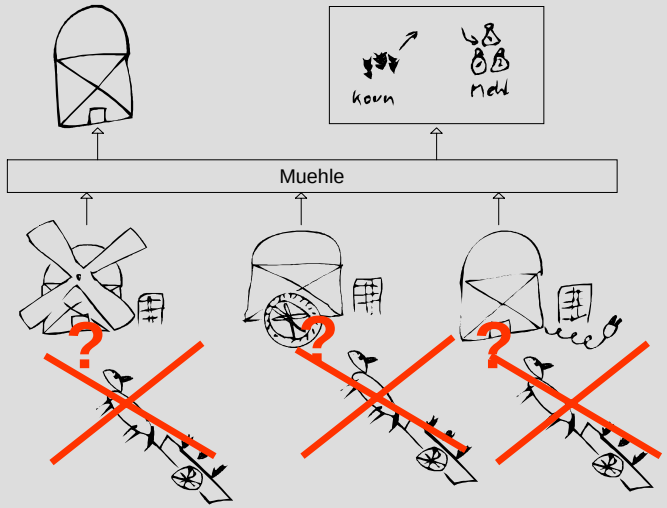



```
public class WindMuehle extends Muehle {
   private boolean wind = true;

   public WindMuehle(String name){
      super(name);
   }

   public void mahlen() throws KeinWindException{
      while(kornlager.top() != null){
         if (!wind) throw new KeinWindException();
         Id inBearbeitungId = (Id) kornlager.pop();
         // mahlen mit Wasser
         mehllager.push(new MehlSack(inBearbeitungId));
      }
   }
}
```

Vorteil des Interface

8

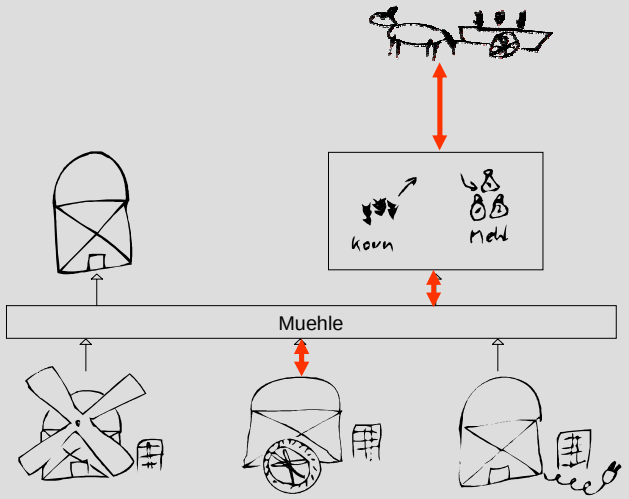
marc-oliver pahl



Vorteil des Interface

9

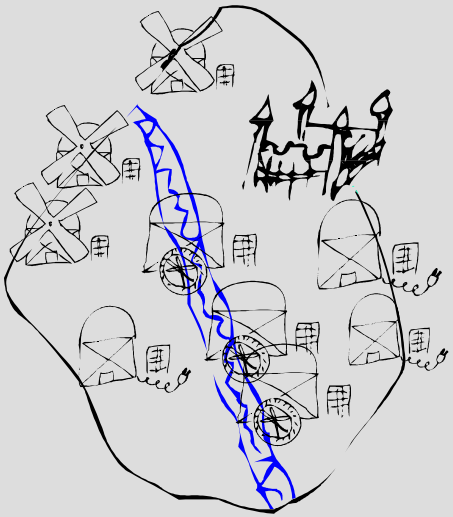
marc-oliver pahl



Vorteil der Instanzen/ Objekte

10

marc-oliver pahl



Der Test

11

marc-oliver pahl

```
public static void main(String[ ] args) {
      IMuehle[ ] meineMuehlen = new IMuehle[4];
      meineMuehlen[0] = new WindMuehle("WindMühle A");
      meineMuehlen[1] = new WindMuehle("WindMühle B");
      meineMuehlen[2] = new StromMuehle("StromMühle");
      meineMuehlen[3] = new WasserMuehle("WasserMühle");

      // 10x wird Korn an zufällig gewählte Mühlen geliefert.
      for(int i=0; i<10; i++){
         meineMuehlen[(int) (4*Math.random())].anliefern(new Korn());
      }

      // alle Mühlen mahlen lassen
      try{ for(int i=0; i<4; i++) meineMuehlen[i].mahlen();
      }catch(MahlException e){  /* irgendetwas sinnvolles tun */  };

      // Lager von allen Mühlen leeren
      MehlSack sack;
      for(int i=0; i<4; i++)  while ((sack = meineMuehlen[i].abholen()) != null);
}
```

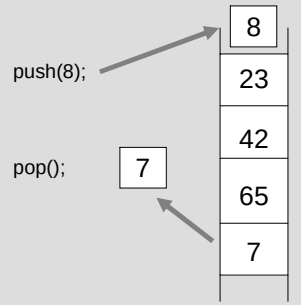
Die Queue

12

marc-oliver pahl

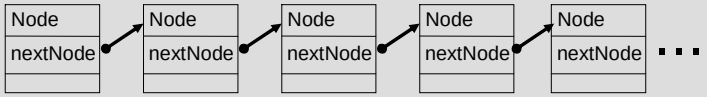
Queue: **F**irst**I**n**F**irst**O**ut:

```
public interface IObjectQueue {

   public void push(Object obj);

   public Object pop();

   public Object top();
}
```



SimplyLinked List

marc-oliver pahl




```
protected class Node{
   protected Node nextNode;
   protected Object nodeData;

   Node(Node next){ nextNode = next; }

   Node(Node next, Object data){ nextNode = next; nodeData = data; }

   void setNextNode(Node next){ nextNode = next; }

   void setData(Object data){ nodeData = data; }

   Node getNextNode(){ return nextNode; }

   Object getData(){ return nodeData; }
}
```

Die Queue

marc-oliver pahl

```
public class LagerQueue implements IObjectQueue {
protected Node head;

public void push(Object obj){
   if (head == null) head = new Node(null, obj);
   else {
      Node actNode = head;

      while(actNode.getNextNode() != null) actNode = actNode.getNextNode();
      // Jetzt ist actNode der letzte Knoten

      actNode.setNextNode(new Node(null, obj));
      // Jetzt ist der neue Knoten hinten angehängt
       }
   }
   ...
}
```

Die Queue

marc-oliver pahl

```
public class LagerQueue implements IObjectQueue {
protected Node head;

   public void push(Object obj){ ... }

   public Object top(){
      if (head == null) return null; else return head.getData();
    }

   public Object pop(){
      if (head == null) return null; // kein Knoten da

      Node res = head;
      head = res.getNextNode(); // ehemaliger Frontknoten ausgekettet
      return res.getData();          // Der ehemalig head wird garbagecollectet
   }
}
```