WEEK 9

MERRY CHRISTMAS

INHERITANCE

```
public class A {
   public int  min(C c, B b) { return 0; }    // Methode 1
   public void min(A a, B b) { }              // Methode 2
}
public class B extends A {
   public void min(A a1, A a2) { }            // Methode 3
}
public class C extends B {
   public B min(A a, C c) { return new B(); } // Methode 4
}
public class Poly {
   public static void main (String[] args) {
      A a = (B)(new C());
      B b = new B();
      C c = new C();
      c.min(a, c);          // Aufruf 1
      b.min(a, (B)c);       // Aufruf 2
      ((B)c).min(c, (B)c); // Aufruf 3
      ((A)b).min((B)a, b); // Aufruf 4
   }
}
```

1. The static types of all the Expressions of a Method Call
2. Signature of the Method when it is called
3. What the Object evaluates to
4. Which Method # is called

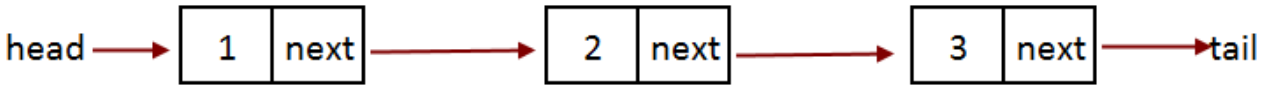
INHERITANCE – METHOD 4

```
public class A {
   public int  min(C c, B b) { return 0; }    // Methode 1
   public void min(A a, B b) { }              // Methode 2
}
public class B extends A {
   public void min(A a1, A a2) { }            // Methode 3
}
public class C extends B {
   public B min(A a, C c) { return new B(); } // Methode 4
}
public class Poly {
   public static void main (String[] args) {
      A a = (B)(new C());
      B b = new B();
      C c = new C();
      c.min(a, c);          // Aufruf 1
      b.min(a, (B)c);       // Aufruf 2
      ((B)c).min(c, (B)c); // Aufruf 3
      ((A)b).min((B)a, b); // Aufruf 4
   }
}
```

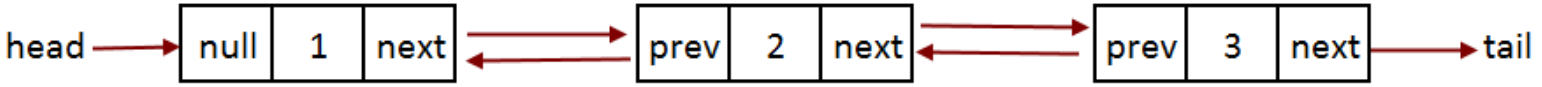
1. The static types of all the Expressions of a Method Call
   ABB
2. Signature of the Method when it is called
   min(B, B)
3. What the Object evaluates to
   B
4. Which Method # is called
   2

DOUBLY LINKED LISTS

DOUBLY LINKED LISTS



Singly Linked List



Doubly Linked List

DOUBLY LINKED LISTS

Singly Linked List	Doubly Linked List
Easy Implement	Not easy
Less memory	More Memory
Can traverse only in forward direction	Traverse in both direction, back and froth

DOUBLY LINKED LISTS – METHODS

- Constructor that initializes head with NULL
- Public int size() – Returns # of elems in the List
- Public void add(int info) – Appends an elem to the end of the list
- Public void add(int index, int info) – Make a new node that will become the $index element. If index is invalid do nothing
- Publc int remove(int index) – Removes an element at $index and returns its value. If there is an invalid index, Integer.MIN_VALUE is to be returned
- Public void shiftLeft(int index) – shifts all elements $index to the left. You are only allowed to change head, prev and next.
  - Ex: [0, 1, 2, 4] will become [2, 4, 0, 1] with shiftLeft(2)
  - Ex: [0, 1] will become [1, 0] with shiftLeft(1)
  - Shiftleft(0) does nothing

DOUBLY LINKED LISTS

- Unit test is provided on the website

DOUBLY LINKED LISTS – METHODS

- Constructor that initializes head with NULL
- Public int size() – Returns # of elems in the List
- Public void add(int info) – Appends an elem to the end of the list
- Public void add(int index, int info) – Make a new node that will become the $index element. If index is invalid do nothing
- Publc int remove(int index) – Removes an element at $index and returns its value. If there is an invalid index, Integer.MIN_VALUE is to be returned
- Public void shiftLeft(int index) – shifts all elements $index to the left. You are only allowed to change head, prev and next.
  - Ex: [0, 1, 2, 4] will become [2, 4, 0, 1] with shiftLeft(2)
  - Ex: [0, 1] will become [1, 0] with shiftLeft(1)
  - Shiftleft(0) does nothing

MODELING

MODELLING

- Create a Grundflaeche.java class with the method stubs of
  - umfang()
  - flaeche()
  - toString()

MODELLING

- Create the classes Kreis, Rechteck, NEck
  - These all extend Grundflaeche
  - These overwrite all functions of Grundflaeche

MODELLING

- Add a istQuadrat() method to all grundflaesche classes
  - Try to solve this by adding code only to two classes

MODELLING

- Create a Class Prisma with the methods
  - volume()
  - oberflaesche()
  - toString()

MODELLING

- Create a class Quadrat that extends Grundflaesche
- Also add a method zuQuadrat which returns a Quadrat that is identical (in length) to the current object. If it is not a Quadrat, return null

MODELLING

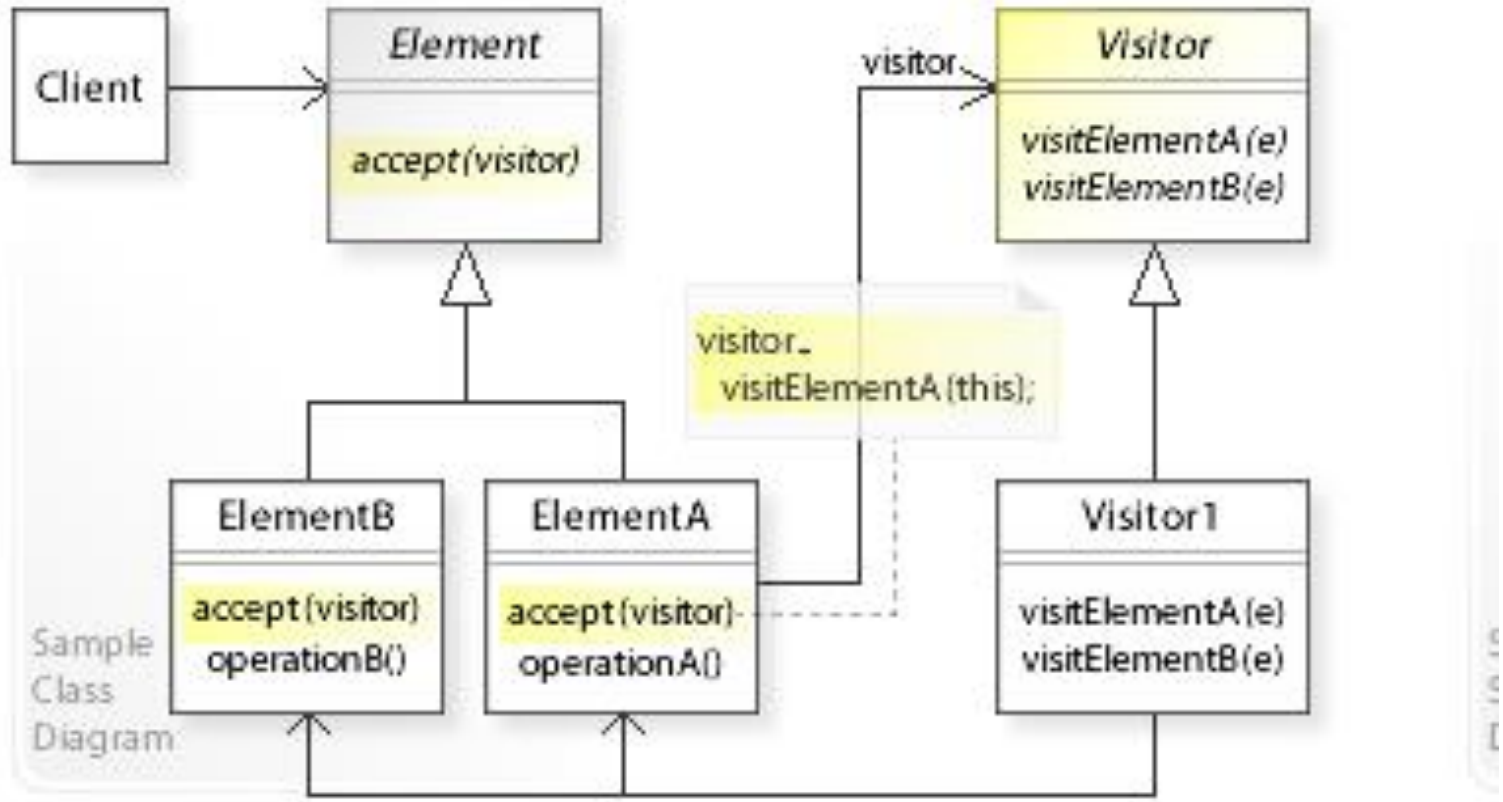
- Add a method to Prisma called istWuerfel(), that checks if the current Object is a cube

TIPS

- Plan it out on paper first
- get Methods may help
- Try to avoid repetition in methods and variables by putting things in the parent class
- Area for a regular n-Sided shape is (a is sidelength)
- You can use Math.pi and Math.tan $\frac{n*a^2}{4*\tan(\frac{\pi}{n})}$
- ÜL has provided a test class (not a JUNIT one however :S)

VISITOR

VISITOR


Client
Element
accept(visitor)
visitor
Visitor
visitElementA(e)
visitElementB(e)
visitor.
visitElementA(this);
ElementB
accept(visitor)
operationB()
ElementA
accept(visitor)
operationA()
Visitor1
visitElementA(e)
visitElementB(e)
Sample
Class
Diagram

VISITOR CLASS

- Add an accept() class as appropriate in each class
- Implement a class Visitor that has a visit() Method for each Class
  - public void visit(Class cName)
- Implement a class FlaeschenVisitor that extends Visitor
  - It should visit each object and get the flaesche