```java
package uebung051;

class Util {

```

Da Exceptions auch Klassen sind, sollten sie das Namensschema für Klassen in Java Einhalten (WertNullException)

```java
    // liefert die kleinste Zahl des uebergebenen Arrays
    public static int minimum(int[] values) throws wertnullException, emptyException {
        if(values ==null){
            throw new wertnullException("Wert darf nicht null sein");
```

gute Überlegungen

```java
        }
        if(values.length == 0){
            throw new emptyException("array darf nicht leer sein");
        }
        int min = values[0];
        for (int i = 1; i < values.length; i++) {
            if (values[i] < min) {
                min = values[i];
            }
        }
        return min;
    }

    // konvertiert den uebergebenen String in einen int-Wert
    public static int toInt(String str) throws wertnullException, emptyException {
        if(str == null){
            throw new wertnullException("Wert darf nicht null sein");
        }

        if(str.isEmpty()){
            throw new emptyException("Wert darf nicht leer sein:");
        }
        int result = 0, factor = 1;
        char ch = str.charAt(0);
        switch (ch) {
```

```
        case '-':
            factor = -1;
            break;
        case '+':
            factor = 1;
            break;
        default:
            result = ch - '0';
        }
        for (int i = 1; i < str.length(); i++) {
            ch = str.charAt(i);
            int ziffer = ch - '0';
            result = result * 10 + ziffer;
        }
        return factor * result;
    }

    // liefert die Potenz von zahl mit exp,
    // also zahl "hoch" exp (number to the power of exp)
    public static long power(long number, int exp) throws negativevalueException {
        if(exp < 0){
            throw new negativevalueException("Wert darf nicht kleiner 0 sein");
        }
        if (exp == 0) {
            return 1L;
        }
        return number * Util.power(number, exp - 1 );
    }
}

public class UtilTest {

    // Testprogramm
    public static void main(String[] args) {
        String eingabe = IO.readString("Zahl: ");
        try {
            int zahl = Util.toInt(eingabe);
```

hier können auch Probleme auftreten, was passiert etwa wenn im default case oder in der Schleife ch keine Ziffer ist? oder wenn der ganze String nur + oder - ist -1P

```
            System.out.println(zahl + " hoch " + zahl + " = "
                    + Util.power(zahl, zahl));
        }
        catch (wertnullException f){
                f.printStackTrace();

        }

        catch(emptyException f){
            f.printStackTrace();
        }

        catch(negativevalueException f){
            f.printStackTrace();
        }

        try{
            System.out.println(Util.minimum(new int[] { 1, 6, 4, 7, -3, 2 }));
            System.out.println(Util.minimum(null));
            System.out.println(Util.minimum(new int[0]));
        }catch(wertnullException e){
            e.printStackTrace();
        }
        catch(emptyException e){
            e.printStackTrace();
        }
    }
}


```

5/6

```
package uebung051;

public class emptyException extends Exception{
        public emptyException(){
            super();
        }

        public emptyException(String message){
            super(message);
        }

            public emptyException(Throwable cause){
            super(cause);
        }

        public emptyException(String message, Throwable cause){
            super(message, cause);
        }
}

```

```java
package uebung051;

public class negativevalueException extends Exception {
    public negativevalueException(){
        super();
    }

    public negativevalueException(String message){
        super(message);
    }

    public negativevalueException(Throwable cause){
        super(cause);
    }

    public negativevalueException(String message, Throwable cause){
        super(message, cause);
    }
}

```

```java
package uebung051;

public class wertnullException extends Exception{
    public wertnullException(){
        super();
    }

    public wertnullException(String message){
        super(message);
    }

    public wertnullException(Throwable cause){
        super(cause);
    }

    public wertnullException(String message, Throwable cause){
        super(message, cause);
    }
}
```

```
package uebung052;

public class Car implements CarComponent{
    private String name;
    private CarComponent[] components;
    public String getName(){
        return name;
    }


}
```

muss nicht noch der Zugriff auf die CarComponents irgendwie hergestellt werden?

```
package uebung052;

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

public final class Bill {

    private List<BillItem> items = new ArrayList<>();
    private double totalPrice;
    public double getTotalPrice(){
        return totalPrice;
    }

    public void addItems(BillItem item1, double price){
        items.add(item1);
        item1.setPrice(price);
    }

    public String toString(){
        String partresult = "";
        double result = 0;
        String resultpart2 = "";
        for(int i = 0; i < items.size(); i++){
                partresult =  partresult + "\n" + items.get(i).toString();
        }
        for(int i = 0; i < items.size(); i++){
            result = result + items.get(i).getPrice();
            resultpart2 = "\nTotal: " + result;
        }

        return partresult + resultpart2;

    }

    public class BillItem{
        private CarComponent item;
        public double price;
```

final class

totalprice ist kein Attribut von Bill sondern die Summe der Preise der BillItems, hier einfach die Werte aufsummieren

hier könnte man die getTotalPrice Methode einsetzen

das hier sollte vermutlich nicht in der Schleife sein? so wird result mehrmals im Ergebnis stehen

inner class

```java


        public double getPrice(){
        return price;
        }

        public CarComponent getItem(){
            return item;
        }

        public String toString(){
           // String pricestring = Double.toString(this.getPrice());
           return this.item.getName() + ":" + " " + price;
        }

        public void setPrice(double price) {
            this.price = price;
        }

        public void setItem(CarComponent c){
            this.item = c;
        }
    }
}

```

```
package uebung052;

import uebung052.carpart.Motor;
import uebung052.carpart.Seat;
import uebung052.carpart.Wheel;

public class test {
    public static void main(String[] args) {
        Bill bill1 = new Bill();

        Bill.BillItem motor = bill1.new BillItem();
       Motor m1 = new Motor("Rolls Royce (Motor )"
);
```

so nicht ganz richtig, es handelt sich um ein Car (Rolls Royce) mit einer Komponente (Motor)

```
       motor.setItem(m1);
       bill1.addItems(motor, 100000);

        Bill.BillItem seat1 = bill1.new BillItem();
       Seat seat1new = new Seat("Seat");
       seat1.setItem(seat1new);
       bill1.addItems(seat1, 1000);

        Bill.BillItem seat2 = bill1.new BillItem();
        Seat seat2new = new Seat("Seat");
        seat2.setItem(seat2new);
        bill1.addItems(seat2, 1000);

        Bill.BillItem seat3 = bill1.new BillItem();
        Seat seat3new = new Seat("Seat");
        seat3.setItem(seat3new);
        bill1.addItems(seat3, 1000);

        Bill.BillItem seat4 = bill1.new BillItem();
        Seat seat4new = new Seat("Seat");
        seat4.setItem(seat4new);
        bill1.addItems(seat4, 1000);


       System.out.println(bill1.toString());
    }
}

```

```java
package uebung052;

public class CarPart implements CarComponent{
    protected String name;
    private CarComponent[] components;
    public String getName(){
        return name;
    }


}
```

```java
package uebung052;

    public interface CarComponent {
    String getName();

}

```

```java
package uebung052.carpart;

import uebung052.CarPart;

public class Seat extends CarPart {
    public Seat(String name){
        this.name = name;
    }
}

```

Package vorhanden

```java
package uebung052.carpart;

import uebung052.CarPart;

public class Motor extends CarPart {
    public Motor(String name){
        this.name = name;
    }
}

```

```
package uebung052.carpart;

import uebung052.CarPart;

public class Wheel extends CarPart {
    public Wheel(String name){
        this.name = name;
    }
}

```

static class fehlt, 3,5/6 Punkte

```java
package uebung053;

import java.util.ArrayList;

public class Group<T extends Older<T>>{
    private ArrayList<T> listmember;

    public Group() {
        listmember = new ArrayList<T>();
    }
    public void add( T member){
        listmember.add(member);
    }

    public T getOldest(){
        if (listmember.isEmpty()) {
            return null;
        }

        T oldest = listmember.get(0);
        for (int i = 1; i < listmember.size(); i++) {
            if (listmember.get(i).isOlder(oldest)) {
                oldest = listmember.get(i);
            }
        }
        return oldest;
    }
}

```

gut

```java
package uebung053;

public interface Older<T>{
    public boolean isOlder(T other);
}

```

```java
package uebung053;

public class Person implements Older<Person>{
    private String name;
    private int age;
    public Person(String name, int age){
        this.name =  name;
        this.age = age;
    }
    public String getName() {
        return name;
    }

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

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public boolean isOlder(Person other){
        if(other.age < this.age){
            return true;
        }
        else{
            return false;
        }
    }

}

```

korrekt, 8/8

```
package uebung053;

public class TestGroup{
        public static void main(String [] args) {
            Group<Person> group = new Group<>();
            group.add(new Person("Alice", 25));
            group.add(new Person("Bob", 23));
            group.add(new Person("Carl", 26));
            System.out.println(group.getOldest().getName());
        }
}

```