

Informatik I – Kapitel 7

„Klassen und höhere Datentypen“

Zusammenfassung des Kapitel 7

Küchlin, Weber, Einführung in die Informatik, 2.Auflage

25.1.2004

- Zum Testen der Date-Klasse habe ich im Projekt noch die folgende Main-Klasse angelegt.
Analog könnt Ihr die Klassen, die Ihr schreibt, testen!

```
public class Main {

    public static void main(String[] args) {
        Date geburtsDatum = new Date(10, 11, 1980);
        Date defaultDatum = new Date();
        System.out.println("geburtsDatum: "+geburtsDatum);
        System.out.println("defaultDatum: "+defaultDatum);
        // hier wird automatisch die .toString-Methode aufgerufen
    }
}
```

Das zip-File mit dem Beispielcode befindet sich auf der Webseite.

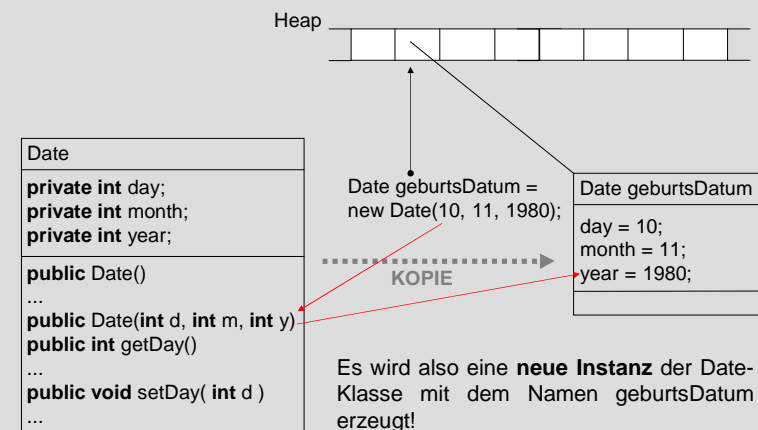
```
public class Date{
    private int day, month, year;
    private static final byte D = 1; private static final byte M = 1;
    private static final short Y = 1900;

    // constructors
    public Date(){ this(D, M, Y); }
    public Date(int y){ this(D, M, y); }
    public Date(int m, int y){ this(D, m, y); }
    public Date(int d, int m, int y){ setDay(d); setMonth(m); setYear(y); }

    // selectors (the selectors are also class-methods)
    public int getDay(){ return day; }
    public int getMonth(){ return month; }
    public int getYear(){ return year; }

    public void setDay(int d){ day=d; }
    public void setMonth(int m){ month=m; }
    public void setYear(int y){ year=y; }

    // methods (for information about the toString method see JDK-Doc „Object“)
    public String toString(){ return (day+"."+month+"."+year); }
}
```



Klassen-/ Instanzvariablen

5
marc-oliver pahl

```

class Id
{
    private static int counter=X; X 4
    private int id;
    public Id(){id = counter++;}
    public int getId(){return id;}
}
    
```

```

class Id idA
{
    counter
    id = 1
}
    
```

```

class Id idB
{
    counter
    id = 2
}
    
```

```

class Id idC
{
    counter
    id = 3
}
    
```

Heap

```

class Test{
    public static void main(String[] args){
        Id idA = new Id();
        Id idB = new Id();
        Id idC = new Id();

        System.out.println( "idA: "+idA.getId() );
        System.out.println( "idB: "+idB.getId() );
        System.out.println( "idC: "+idC.getId() );
    }
}
    
```

Assertions

7
marc-oliver pahl

- generieren abhängig von einer Bedingung zur Laufzeit Meldungen und dienen
 - der aktiven Dokumentation, also der Ausgabe von Informationen über den aktuellen Zustand, indem sie z.B. die Schleifenbedingungen überprüfen...
 - der Fehlersuche (Debugging)

```

public static int sum_rek(int n){
    assert n>=0 : "\ !!(n>=0) n="+n;
    if ( n == 0 ) return 0;
    return ( n + sum_rek( n-1 ) )
}
    
```

zu dem Beispiel siehe auch Folien sum_java

Exceptions

6
marc-oliver pahl

- dienen der kontrollierten Ausnahmebehandlung

Die Exception:

```

public class MyException extends Exception{
    public MyException(String val){ super(val); }
}
    
```

Methode, die die Exception wirft:

```

public static void myProcedure() throws MyException{
    throw new MyException("Ich bin eine Exception");
}
    
```

Code, der die Methode aufruft:

```

public static void main(String[] args) {
    try{ myProcedure(); }
    catch(MyException e){ System.out.println("MyException: "+e.getMessage()); }
}
    
```

einfach verkettete Listen (simple linked lists)

8
marc-oliver pahl

```

SimplyLinkedList
{
    private Node startNode;
    ...
    public SimplyLinkedList()
    ...
    public T getDataAt(int i)
    ...
}
    
```

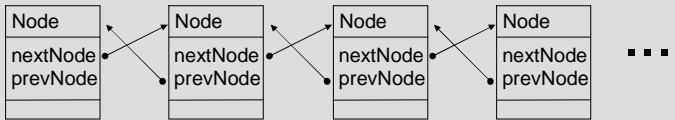
```

Node
{
    private Node nextNode;
    private T value;
    ...
    public Node(T value)
    ...
    public T getValue()
    ...
}
    
```

T steht jeweils für einen Datentyp, z.B. int, String, Date, ...

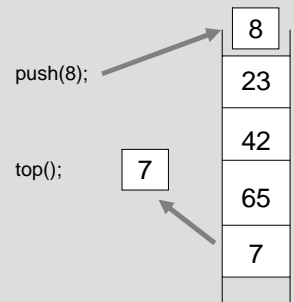
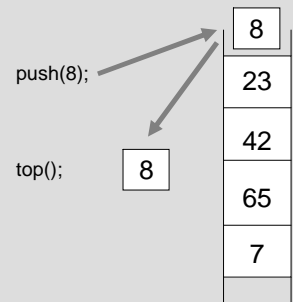
```
DoublyLinkedList  
private Node startNode;  
...  
public SimplyLinkedList()  
...  
public T getDataAt(int i)  
...
```

```
Node  
private Node nextNode;  
private Node prevNode;  
private T value;  
...  
public Node(T value)  
...  
public T getValue()  
...
```



Stack: LastInFirstOut:

Queue: FirstInFirstOut:



Die Anordnung der Werte von oben nach unten ist willkürlich gewählt und könnte genau so gut andersrum sein.