Wie entwerfe ich ein Programm?

Welche Objekte brauche ich?

Flussdiagramme für Programmablauf

Vorcode

Testcode

Hauptcode

Wir spielen Lotto!

Was für Klassen/ Objekte brauche ich?

1. Vorcode

```
class Benutzereingabe {
  // Eingabe von Tastatur
  // einlesen
}
```

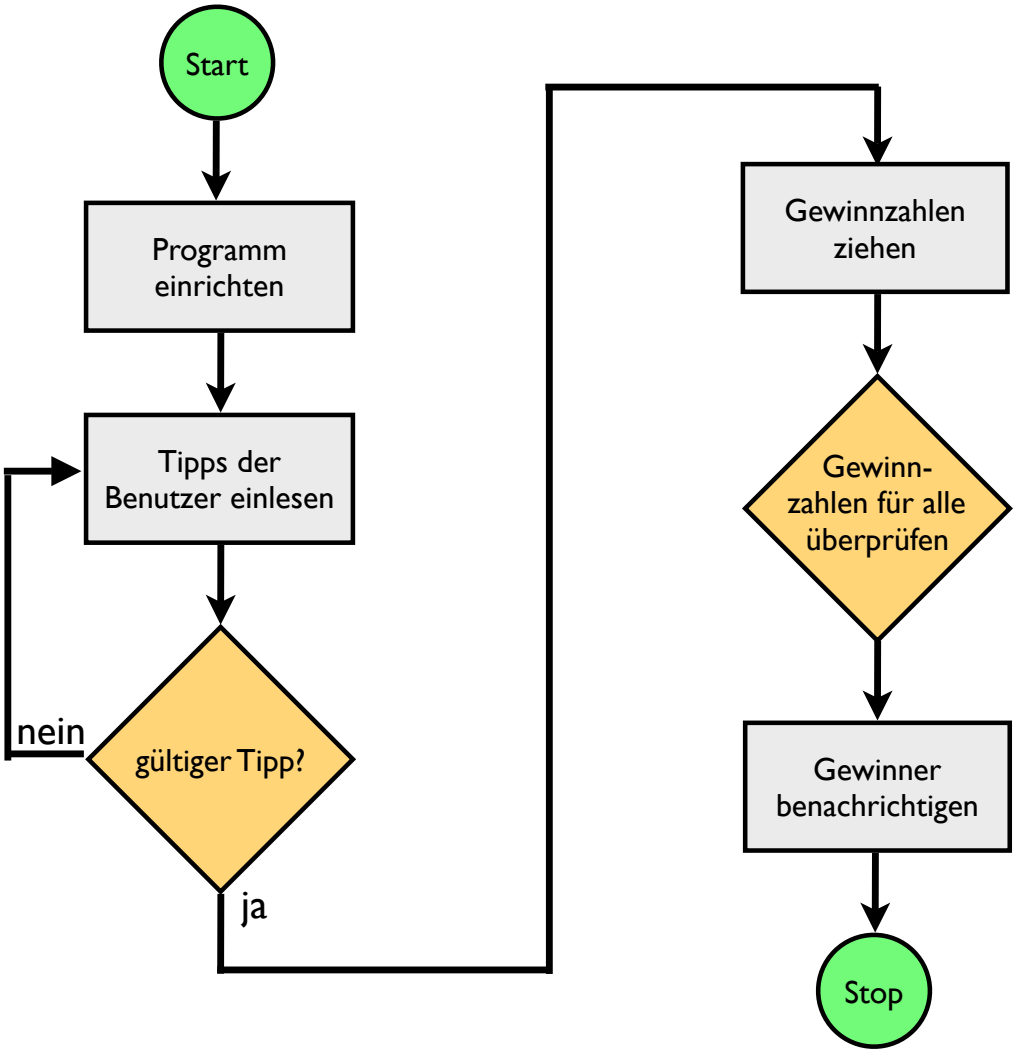
```
class LottoTipp {
  // speichert Tipp und
  // Namen des Spielers
}
```

```
class LottoGesellschaft {
  // Gewinnzahlen ziehen,
  // mit Tipp vergleichen
```

```
class LottoSpiel {
  // Spieler verwalten,
  // Hauptcode testen
```

Flussdiagramm

1. Vorcode



```
// Testet den Hauptcode und managt die Spieler
class LottoSpiel {
   ...                                                    2. Testcode
   // Objekte erstellen
   LottoGesellschaft westLotto = new LottoGesellschaft ();
   Benutzereingabe eingabe = new Benutzereingabe ();
   LottoTipp JansTipp = new LottoTipp ();
   ...
   int[] lottoZahlen = new int[6];     // für Benutzereingabe
   ...
   JansTipp.setTipp (lottoZahlen);    // Setter
   westLotto.zieheGewinnzahlen ();
   westLotto.pruefeTipp (JansTipp.getTipp ());    // Getter
   ...
}
```

3. Hauptcode


```
// speichert einen Tipp und den Namen des Spielers
class LottoTipp {
  private int[] tipp = new int[6];
  private String name;

  // Setter und Getter
  public void setTipp (int[] numbers) {...}
  public int[] getTipp () { return tipp; }
  public void setName (String spieler) {...}
  public String getName () { return name; }
}
```

Ein Code-Fertiggericht!

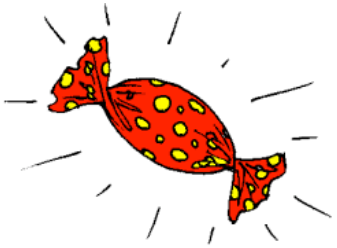
3. Hauptcode

```
public class Benutzereingabe {
  public String getBenutzereingabe (String prompt) {

    // Gibt eine Aufforderung (prompt) aus,
    // liefert die Eingabe von der Tastatur als String

  }
}
```

Aber nicht alles ist schon vorgekocht...

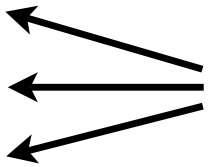
Hier ein paar Zutaten, die wir noch brauchen:

Wrapper-Klassen für elementare Datentypen



```
Benutzereingabe eingabe = new Benutzereingabe ();
String tipp = eingabe.getBenutzereingabe ("Ihr Tipp?");

// Wir brauchen einen int-Wert!

int tippAlsInt = Integer.parseInt (tipp);  // String → int
```

Byte, Short, Integer, Long
Float, Double
Character
Boolean



Wrapper-Klassen

Elementare Typen casten

```
byte klein = 17;                  // Literale sind int!
short gross = klein;              // implizite Umwandlung

short gross = 12461;
byte klein = gross;               // geht nicht
byte klein = (byte) gross;        // explizite Umwandlung
                                  // klein wird zu -83

int preis   = (int) 19.99f;       // 19
float preis = (int) 19.99f;       // 19.0
```

Elementare Ganzzahlen casten

Boolesche Ausdrücke verknüpfen

Und: &&
Oder: ||

Nicht: !

(Nicht verwechseln mit !=)

```
if (alter >= 6 && alter < 60) {
   ermäßigung = false;
}

if (alter < 6 || alter >= 60) {
    ermäßigung = true;
}

if ( (alter != 99 && !name.equals.("Cubbi")) || alter > 500 ) { ... }
```

Auswertungsreihenfolge unklar?
Klammern schaffen Klarheit

Noch ein paar nützliche Operatoren...

```
// Post-Inkrement-Operator
anzahl ++;                  // verwende Wert, erhöhe ihn

// Prä-Inkrement-Operator
++ anzahl;                  // erhöhe Wert, verwende ihn

// Modulo für ganzzahligen Rest der Division
rest = 10 % 3;              // rest = 1
rest = 27 % 4;              // rest = 3
```

?

Welche Werte haben x und y am Ende von A bzw. B?

Gewinnegewinnegewinne!

SMS mit Ergebnis an:

A:
```
x = 1;
y = x++*3;
```

B:
```
x = 1;
y = ++x*3;
```

Die SMS soll nur genau 4 Zahlen enthalten:
x und y für A
und
x und y für B
(in dieser Reihenfolge)

Wiederholung aus der ersten Vorlesung: Die **while**-Schleife

Solange eine Bedingung **wahr** ist, wiederhole.

while (Schleifenbedingung) { Anweisungsblock }

Die Schleifenbedingung wird zu einem Boolschen Wert ausgewertet:
true oder **false**

Boolsche Tests prüfen den Wert einer Variablen, z.B.
<, >, ==, >=, <=

```
int summe = 0;
int zahl = 1;

// wir nur ausgeführt, wenn Bedingung true ist
while (zahl <= 100) {
      summe = summe + zahl;
      zahl ++;    // entspricht zahl = zahl + 1
}

System.out.println ("Die Summe ist " + summe);
```

Besser, wenn man die Anzahl Durchläufe schon kennt: Die **for-Schleife**

Initialisierung *Boolescher Test* *Wiederholungsausdruck*

```
for (int zahl = 0; zahl < 6; zahl++) {
    // etwas berechnen...
    ...
    ...
    // Erst nach allen Anweisungen in der Schleife
    // erhöht die JVM hier den Wert von zahl.
}
```

Variablenwerte von zahl:

```
int zahl;

for (zahl = 0; zahl < 6; zahl++) {
   // am Anfang
   ...
   // am Ende
}

// nach der Schleife
System.out.println (zahl);   // 6
```

am Anfang	am Ende
0	**1**
1	**2**
2	**3**
3	**4**
4	**5**
5	**6**

Enhanced **for**-Schleife

Spart Tipparbeit und Fehler!

```
int[] gewinnZahlen = new int[6];
```

Iterationsvariable "in" Element-Collection, z.B. ein Array

```
for (int zahl  :  gewinnZahlen) {
    // Die Variable zahl enthält in jeder Iteration das
    // nächste Element des Arrays.
}
```

1. Iteration: zahl == gewinnZahlen[0];
2. Iteration: zahl == gewinnZahlen[1];
3. Iteration: zahl == gewinnZahlen[2];
4. Iteration: zahl == gewinnZahlen[3];
5. Iteration: zahl == gewinnZahlen[4];
6. Iteration: zahl == gewinnZahlen[5];


```
// Gewinnzahlen ziehen, mit Tipp vergleichen
class LottoGesellschaft {
  private int[] gewinnZahlen = new int[6];

  private void zieheGewinnzahlen () {
      // wie geht das?
  }

  // Anzahl Treffer zurückliefern
  public int pruefeTipp (int [tipp]) { // Übungsaufgabe }
}
```

Die Java-API...
...hat viele Fertiggerichte!

double Math.random () // liefert Zahl aus [0...1[

```
// Gewinnzahlen ziehen
private void zieheGewinnzahlen () {
  for (int zahl = 0; zahl < 6; zahl++) {
    gewinnZahlen[zahl] = (int) (Math.random() * 49) + 1;
  }
}
```

It's demo time!

javac LottoSpiel.java
java LottoSpiel

Für Sie als Übungsaufgaben...

Wertebereich der Zahlen prüfen, Tipps und Gewinnzahlen vegleichen... und noch viel meeehr...

Nein, die Lösung gibt's nicht bei Media Markt!

Die Java-API und Pakete

double Math.random ()

Klasse Methode

System.out.println ()

im Paket **java.lang**

ArrayList

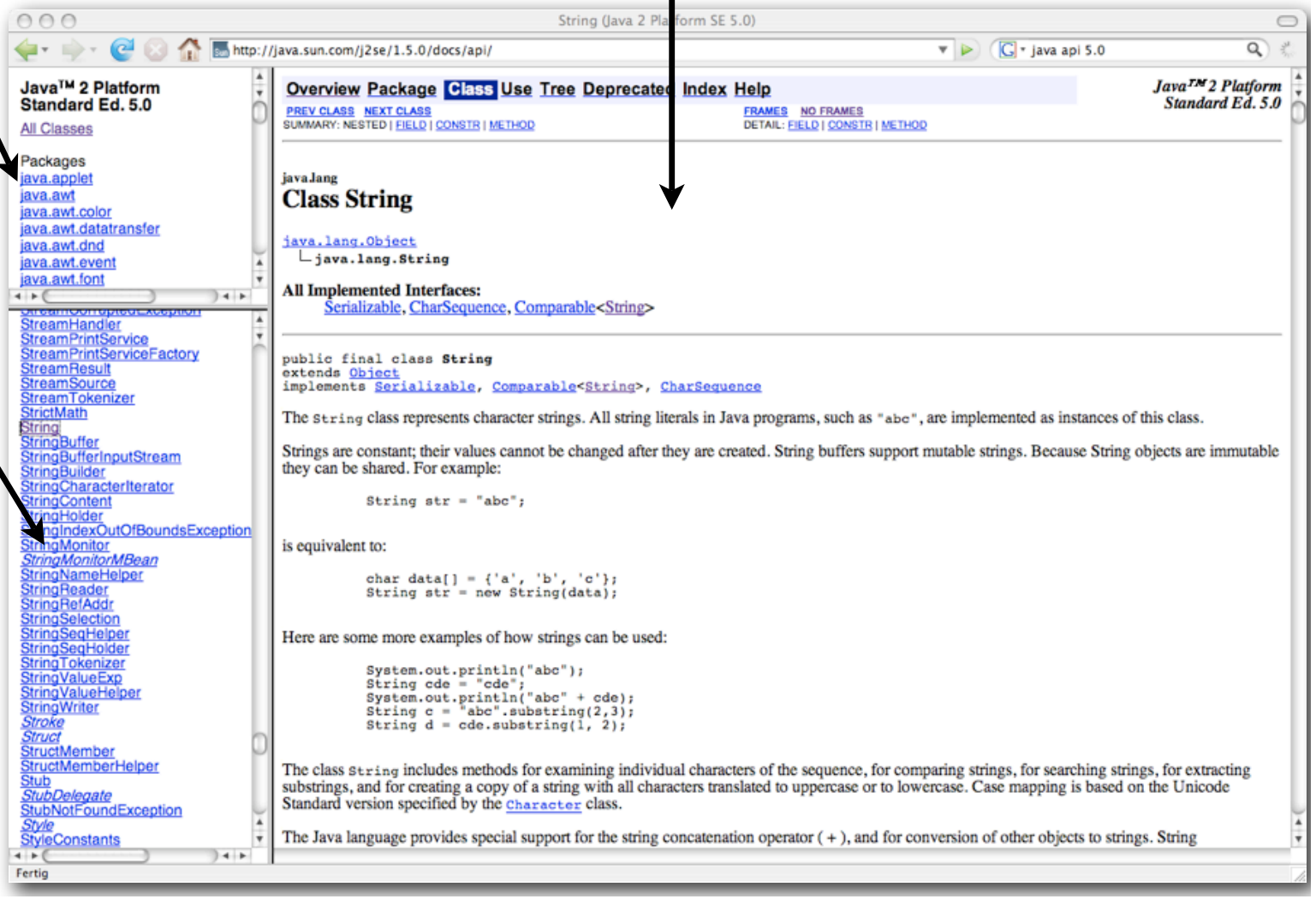
im Paket **java.util**

http://java.sun.com/j2se/1.5.0/docs/api/

alle **Pakete**

alle **Methoden** in einer Klasse

alle **Klassen** im einem Paket



Pakete importieren


```
// import Klassenname;
import java.lang.*;      // importiert der Compiler für Sie
import java.io.*;        // benutzt von Benutzereingabe.java
import java.util.ArrayList;      // eine Klasse importieren

// ohne import müssten sie jedesmal tippen!
java.lang.System.out.println ("Eine Methode aus java.lang");

public java.util.ArrayList<LottoTipp> nurVerlierer (...) { ... }
```

Paketname

Arrays

```
LottoTipp deinTipp = new LottoTipp ();          // nur ein Spieler
LottoTipp[] alleTipps = new LottoTipp[2000];    // viele Spieler
```

Größe nicht veränderbar!

ArrayList <Typ>

```
import java.util.ArrayList;
ArrayList<LottoTipp> alleTipps = new ArrayList<LottoTipp> ();
```

Typ-Parameter

Liste wächst und schrumpft dynamisch

Speichert nur LottoTipps!

LottoTipp-Objekt

```
alleTipps.add (deinTipp);
                                        alleTipps.add (meinTipp);
                                    alleTipps.remove (meinTipp);
    int anzahl = alleTipps.size ();
ArrayList<LottoTipp> alleTipps = new ArrayList<LottoTipp> ();
  int index = alleTipps.indexOf (JansTipp);
                                boolean istLeer = alleTipps.isEmpty ();
   boolean istDarin = alleTipps.contains (meinTipp);
```

Enhanced For-Schleife... ...und Objekte


```
ArrayList<LottoTipp> alleTipps = new ArrayList<LottoTipp> ();

// LottoTipp Objekt                                    // Getter
                    ↘                                      ↘
for (LottoTipp einTipp :  alleTipps) {
    int treffer = westLotto.pruefeTipp (einTipp.getTipp());
    if (treffer >=3) { // ein Gewinner !!! }
}
```

Jetzt sind Sie wieder an der Reihe!

Die Details stehen in den **KapiteIn 5 + 6**.