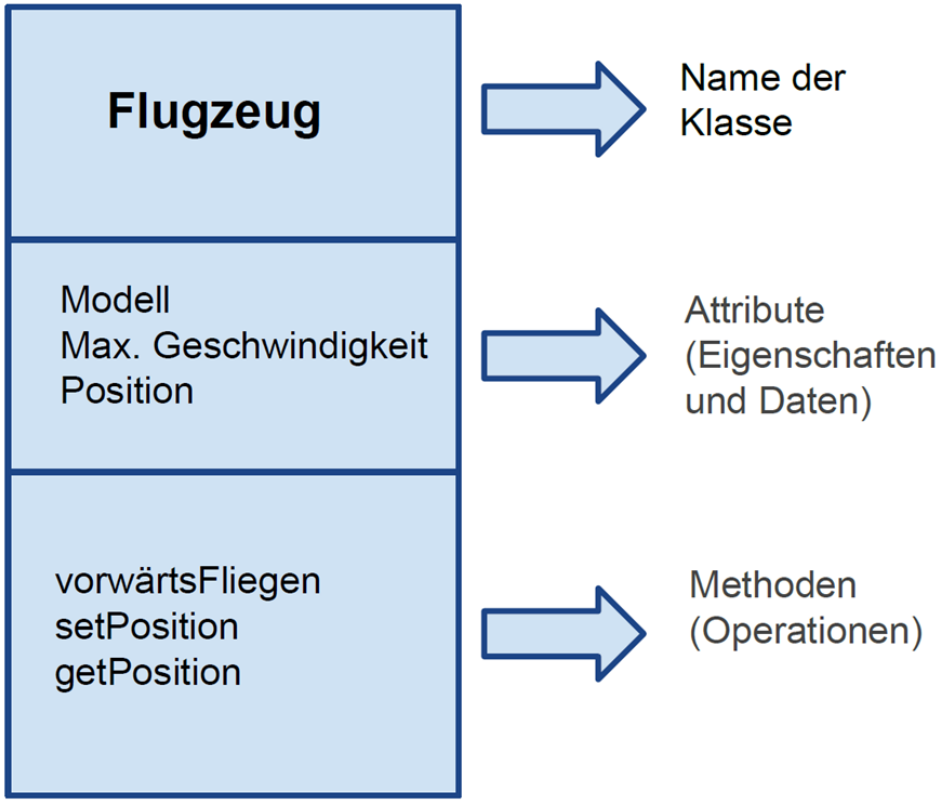
Flugzeug Beispiel

(Handout orientiert sich an: http://www.itslot.de/2013/12/java-objektorientierung-lernen-beispiel.html)

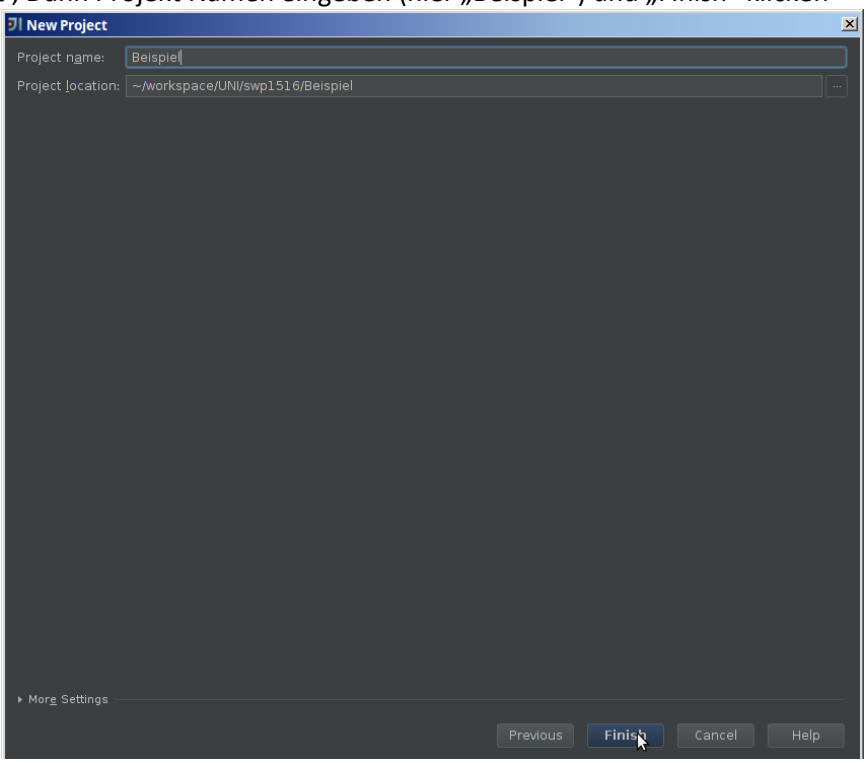
Szenario:

Wir werden mit unserem Programm diverse Flugzeuge bauen und mit denen anschließend fliegen.
Jedes Flugzeug hat folgende Eigenschaften/Daten:
Modell, Maximale Geschwindigkeit, Position im Himmel.
Mit der Position ist die Entfernung von der Startposition gemeint. Jedes Flugzeug startet mit der Position 0 km.

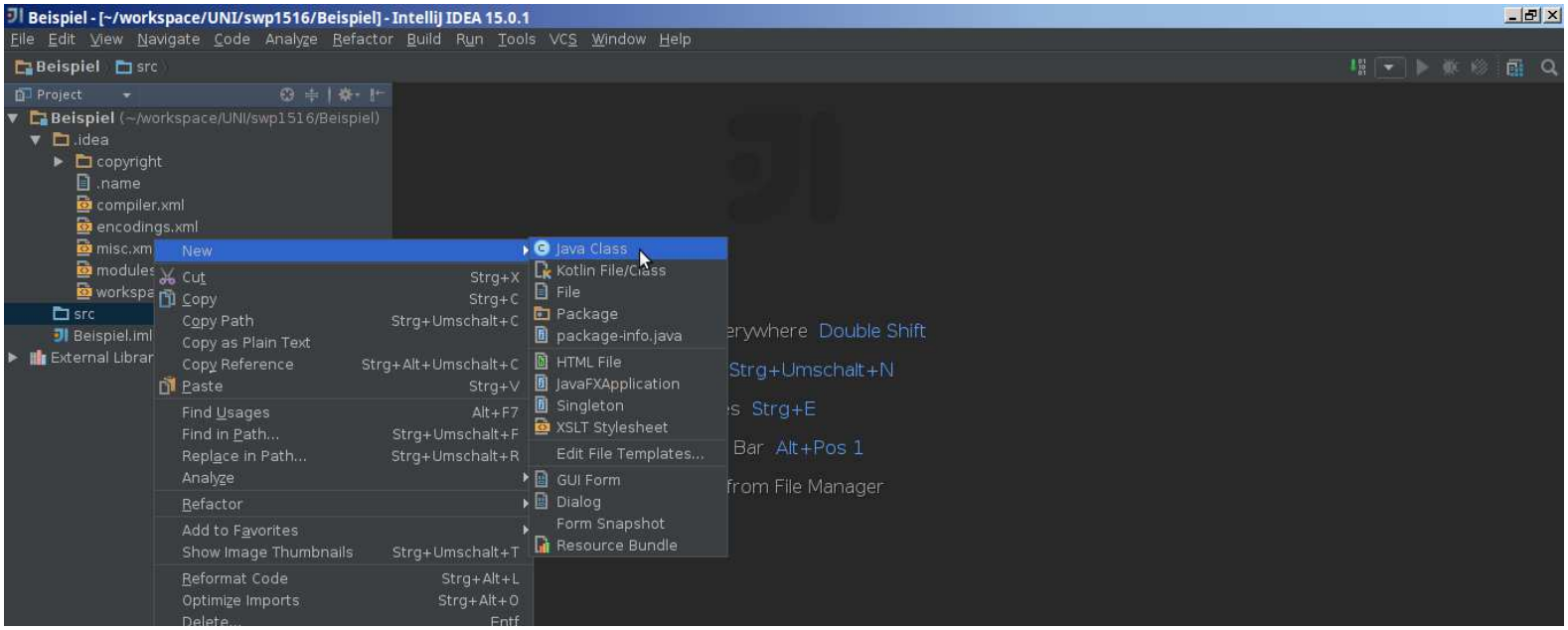


Legen wir nun los…

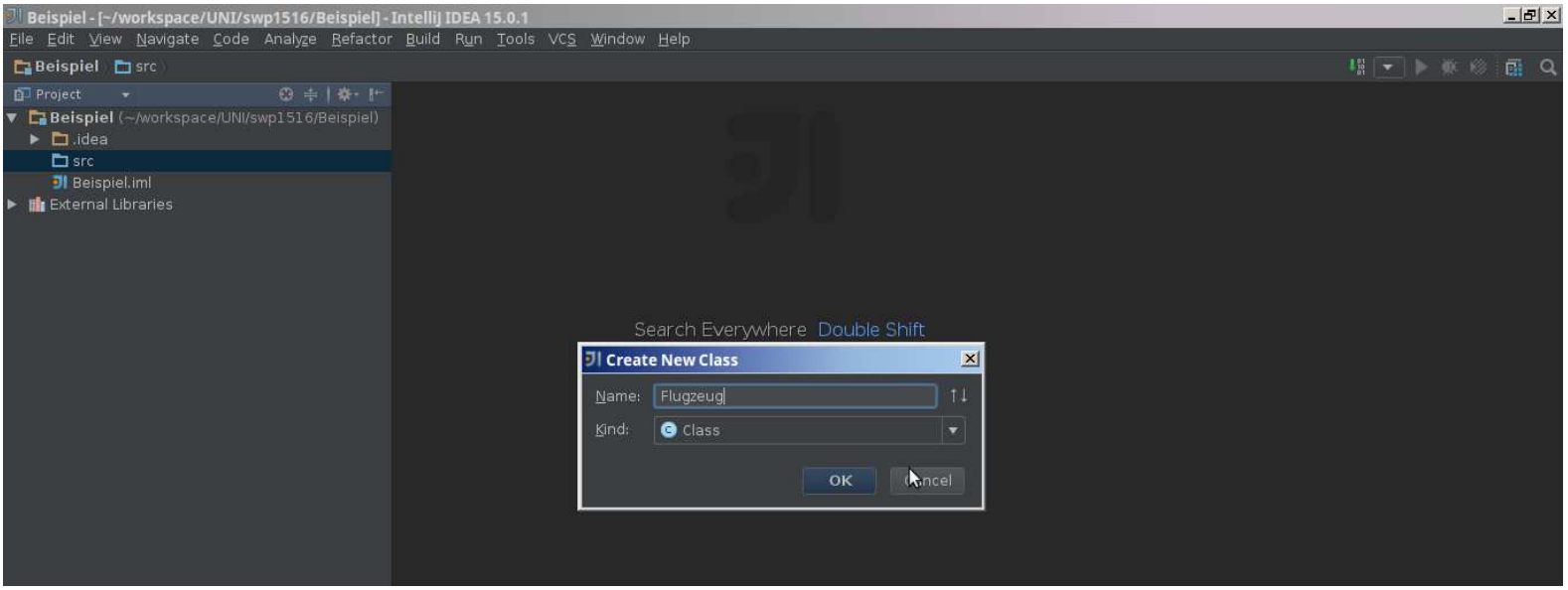
Als erstes startet ihr intelliJ und öffnet ein Projekt:
New –> Project ; Dann Projekt Namen eingeben (hier „Beispiel“) und „Finish“ klicken



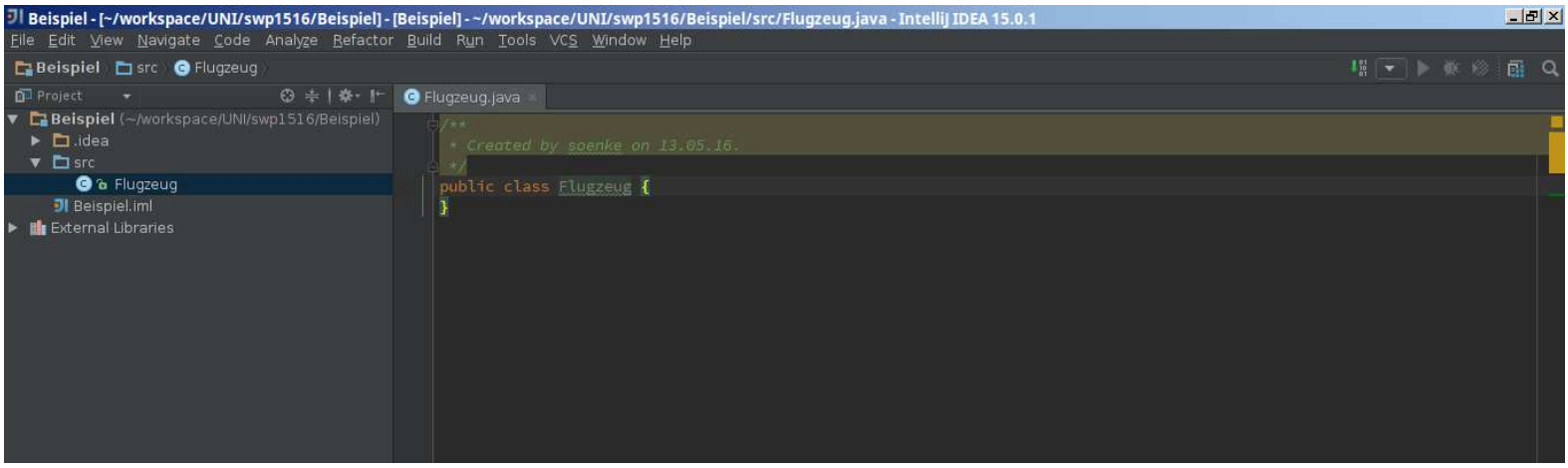
Nun haben wir ein Projekt erstellt. Jetzt brauchen wir nur noch eine Java Klasse, in der wir programmieren können. Das geht folgendermaßen:
Rechtsklick auf „src“ im Projekt Ordner (linke Seite) –> New –> Java Class



Es wird nun eine leere Klasse erstellt und man kann den Namen der Klasse eingeben („Flugzeug“).



Die Flugzeug Klasse sieht nun folgendermaßen aus:



Sieht noch ziemlich leer aus…
Es ist also an der Zeit, die Klasse mit Leben zu füllen!

Als erstes erstellen wir die Attribute:

```
//Attribute
private String flugzeugModell;
private int maxSpeed;
private int Position;
```

Das schreiben wir direkt hinter die geschweifte Klammer von: public class Flugzeug {

Was bedeutet ein Attribut in Java?
Attribute: Hier werden die Eigenschaften und Daten definiert, die unsere Flugzeuge haben.
private => Das Attribut ist außerhalb dieser Klasse (class Flugzeug) nicht veränderbar/erreichbar.

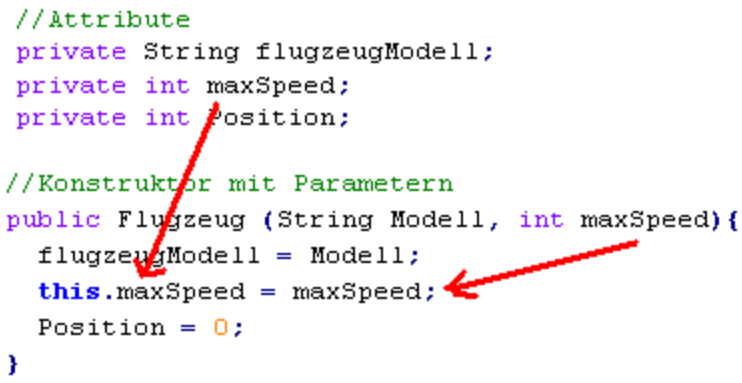
Nun erstellen wir die Konstruktoren, dafür schreiben wir nach den Attributen:

```
//Konstruktor mit Parametern
public Flugzeug (String Modell, int maxSpeed){
  flugzeugModell = Modell;
  this.maxSpeed = maxSpeed;
  Position = 0;
}

//Konstruktor ohne Parametern
public Flugzeug (){
  flugzeugModell = "default";
  maxSpeed = 100;
  Position = 0;
}
```

Was bedeutet ein Konstruktor in Java?
Konstruktor ist eine Methode, die beim Erzeugen eines Objektes (mit new-Operator) aufgerufen wird.
Ein Konstruktor hat immer den gleichen Namen wie die Klasse.
Konstruktor hat keinen Rückgabewert.

Im Konstruktor kann man die Anfangswerte vom Objekt festlegen (z.B. Position = 0;).

this.maxSpeed => "this." wird dann geschrieben, wenn der Parameter gleich wie das Attribut heißt.
Um das einmal zu verbildlichen:



Wir haben in unserem Beispiel 2 Konstruktoren, d.h. das Objekt kann mit dem 1. oder 2. Konstruktor erstellt werden.
Wenn beim Erzeugen des Objekts die Parameter (flugzeugModell, maxSpeed) nicht angegeben werden [z.B Flugzeug Flugzeug1 = new Flugzeug ()], (←-Dazu später mehr)
wird das Objekt mit dem Konstruktor ohne Parameter erstellt. Dabei werden dem Objekt Standartwerte zugewiesen:
flugzeugModell = "default"; maxSpeed = 100; Position = 0;
Wenn wir ein Objekt mit Parametern erstellen:
[z.B. Flugzeug Flugzeug2 = new Flugzeug (Stealth, 500,)], werden die Werte der Parameterliste dem Objekt zugewiesen.
(String Modell, int maxSpeed) => Das sind die Parameter.

Kommen wir nun zu den Methoden, schreibt dafür:

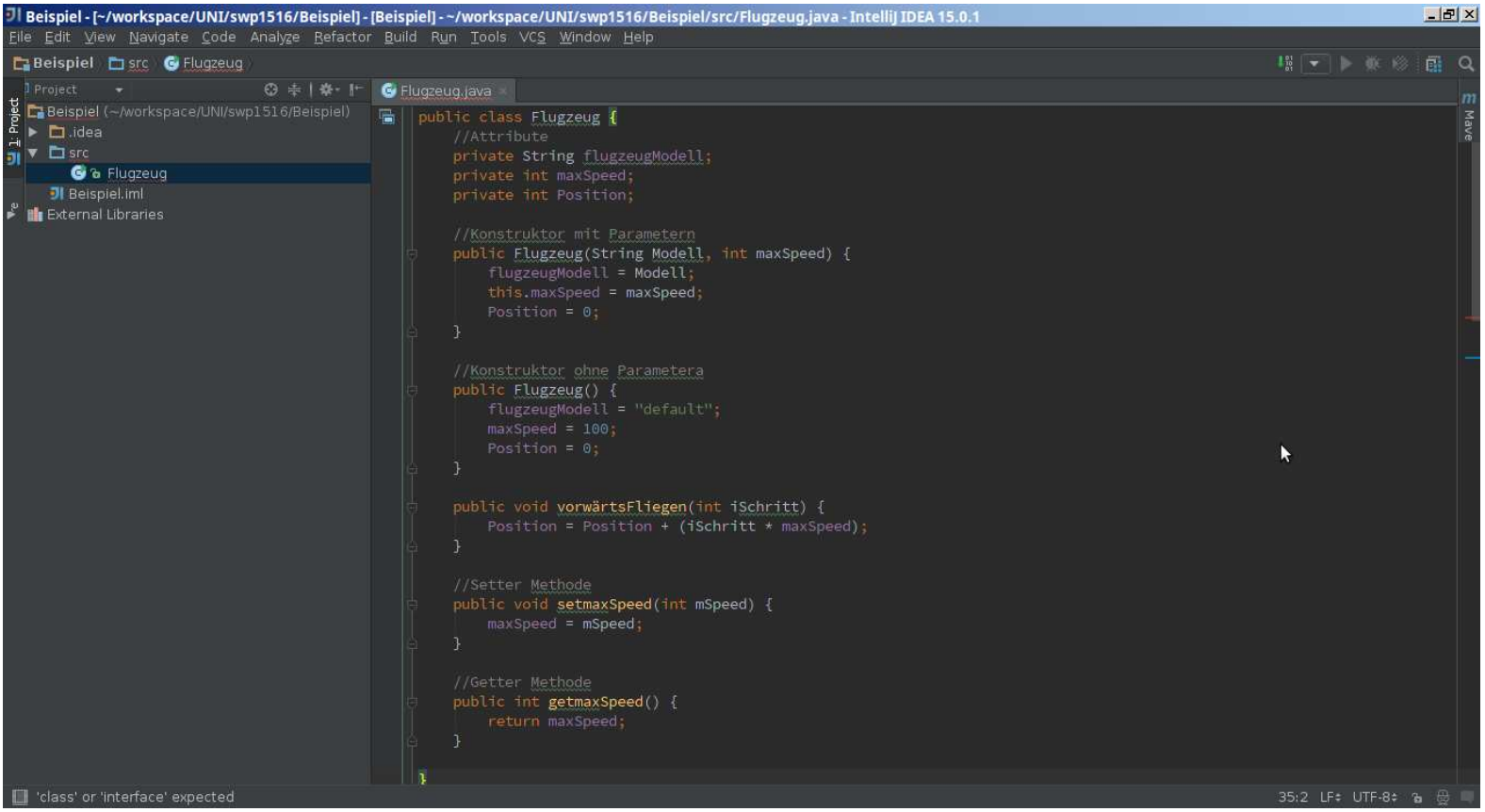
```
//Methode
// Jedes Flugzeug hat ja unterschiedliche max. Geschwindigkeit (maxSpeed),
//deswegen wird bei der Berechnung der Position die Anzahl der Schritte
//mit maxSpeed multipliziert.
  public void vorwärtsFliegen(int iSchritt){
    Position = Position + (iSchritt * maxSpeed);
  }

  //Setter Methode
  public void setmaxSpeed (int mSpeed){
    maxSpeed = mSpeed;
  }

  //Getter Methode
  public int getmaxSpeed (){
    return maxSpeed;
  }
```

Was bedeutet eine Methode in Java?
Eine Methode ist in der Objektorientierung ein Unterprogramm, das für das Hauptprogramm bestimmte Berechnungen/Änderungen durchführt oder Daten abfragt.
Die Beispiele von Unterprogrammen:

- vorwärtsFliegen (ändert die Position vom Flugzeug),
- setmaxSpeed (ändert max. Geschwindigkeit vom bestimmten Flugzeug),
- getmaxSpeed (liefert den Wert von der max. Geschwindigkeit vom bestimmten Flugzeug)

Ihr seht hier schon zwei unterschiedliche Arten von Methoden, die eine heißen Setter und die andere Getter Methoden. Die Setter Methoden setzen einen Wert. Bei der Getter Methode, bekommen wir einen Wert zurück, wir fragen also einen Wert ab.

Ok, soweit so gut. Wie sieht es denn bisher im Editor aus?
The Road so far:



Gut, jetzt fehlt nur noch eine Start bzw. Einstiegsmethode.
In der Präsentation habt ihr gerade gelernt, dass ein Programm immer mit Main beginnt, darum schreiben wir nun die Main Methode:

```
public static void main(String[] args) {
  // TODO Auto-generated method stub

  //Erstellen des Objekts mit dem Konstruktor ohne Parameter
  Flugzeug Flugzeug1 = new Flugzeug();
  System.out.println("Modell = " + Flugzeug1.flugzeugModell);
  System.out.println("maxSpeed = " + Flugzeug1.maxSpeed + " km/h");
  System.out.println("Position = " + Flugzeug1.Position + " km");
```

Würde man das schon mal in der Konsole ausführen lassen, bekommen wir folgendes Ergebnis:
Achtung: Das Programm ist noch nicht fertig, die geschlossene geschweifte Klammer ist ja auch noch nicht gesetzt und wie man kompiliert zeigen wir später erst, aber nur um die Ergebnisse kurz zu verstehen, nehmen wir an, wir haben es gestartet.

// Ergebnis in der Konsole:
Modell = default
maxSpeed = 100 km/h
Position = 0 km

Tippen wir nun weiter ein:

```
//Wir fliegen mit dem Flugzeug1 5 Schritte.
Flugzeug1.vorwärtsFliegen(5);
System.out.println("Position = " + Flugzeug1.Position + " km");
```

//Würde man es wieder in der Konsole aufrufen, würde nun dort stehen:
Position = 500 km

Zur Erklärung: Also 5 Schritte * maxSpeed (100 km/h) werden zur Position 0 dazugerechnet = 500 km

Tippen wir weiter ein:

```
//Erstellen des Objekts mit dem Konstruktor mit Parametern
Flugzeug Flugzeug2 = new Flugzeug("stealth", 1000);
System.out.println("Modell = " + Flugzeug2.flugzeugModell);
System.out.println("maxSpeed = " + Flugzeug2.maxSpeed + "km/h");
System.out.println("Position = " + Flugzeug2.Position + " km");
```

// Ergebnis in der Konsole wäre:
Modell = stealth
maxSpeed = 1000km/h
Position = 0 km

Fügen wir weiter zu Klasse nun folgendes hinzu:

```
//Wir fliegen mit dem Flugzeug2 5 Schritte.
Flugzeug2.vorwärtsFliegen(5);
System.out.println("Position = " + Flugzeug2.Position + " km");
```

// Ergebnis in der Konsole:
Position = 5000 km

Nochmal zur Erklärung: 5 Schrtte * maxSpeed (1000 km/h) = 5000 km

Wir kommen fast zum Ende, jetzt tippen wir noch folgendes ein:

```
//Wir können die maximale Geschwindigkeit vom Flugzeug1 nachträglich auf 10000 km/h ändern:
Flugzeug1.setmaxSpeed(10000);

//Aktuelle maximale Geschwindigkeit anzeigen
System.out.println("maxSpeed = " + Flugzeug1.getmaxSpeed() + "km/h");
```

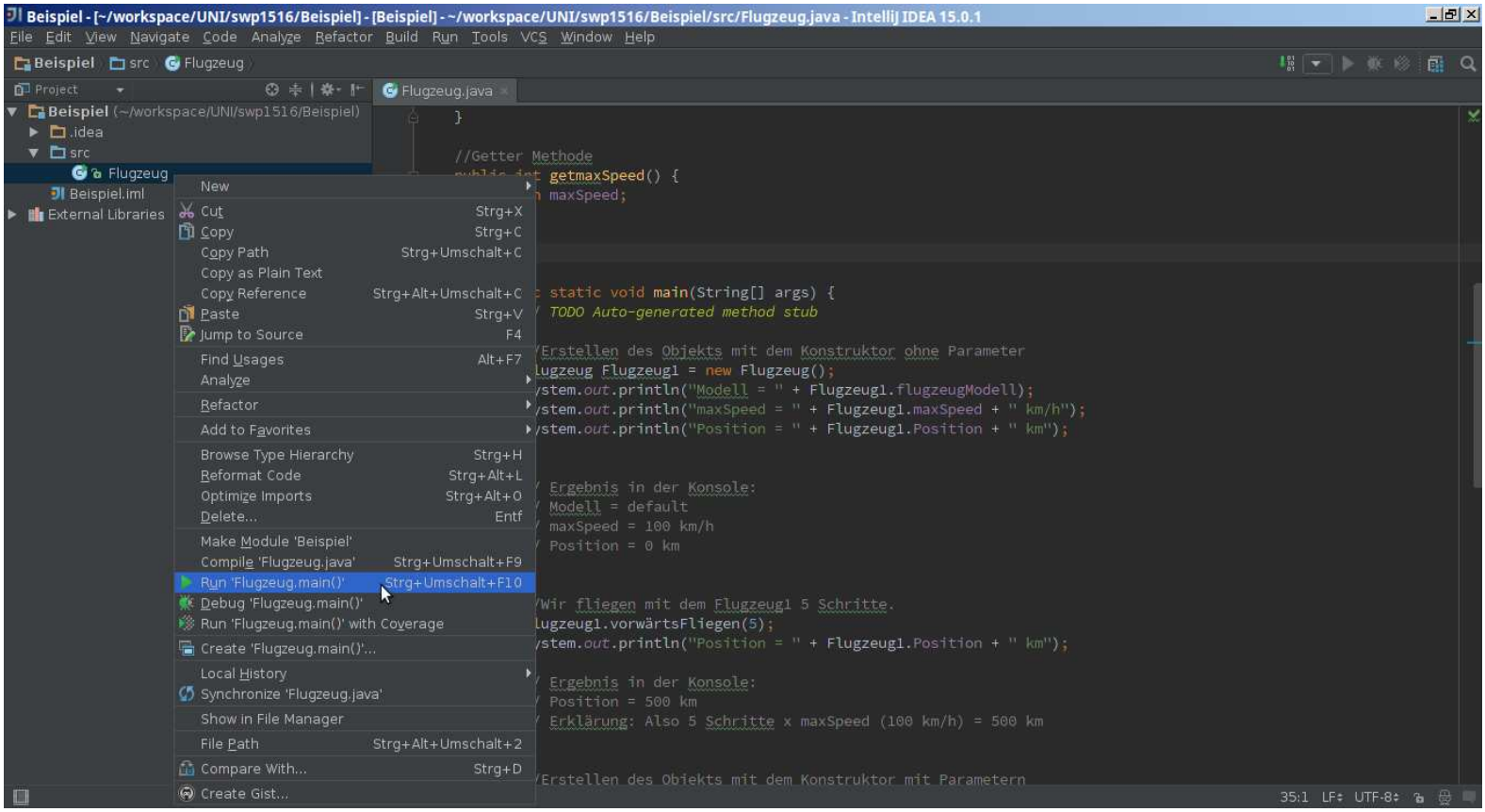
// Ergebnis in der Konsole:
maxSpeed = 10000km/h

Zum Schluss müssen wir die Blöcke nun auch noch mit den geschweiften Klammern schließen:

```
    }

}
```

Dabei steht die erste geschweifte Klammer für die Main Methode und die zweite steht für den Block in der Klasse selber.

Nun wollen wir unser Flugzeug auch mal testen, also müssen wir es kompilieren und ausführen.
Das geht mit Run: Rechtsklick auf die Java Klasse -> Run „Flugzeug.main()“ auswählen



Jetzt wird das Projekt kompiliert und die Main Methode aufgerufen.
Ihr solltet nun eine Ausgabe in der (intelliJ) Konsole sehen: