Scala Cheatsheet

```scala
package intro

/* Interface Person */
trait Person {
  /* - Abstrakte Methode (Signatur ohne Rumpf) (wie abstrakte Methode in Java)
   * - Parameter msg hat den Typ String
   * - Return Typ der Funktion ist Unit (wie void in Java)
   * - Definitionen in Scala: Name ':' Typ
   */
  def speak(msg: String): Unit
}

/* Class Hacker
 * - name und age sind Konstruktorparameter und Variablen auf einen Streich
 * - val ist immutable (final)
 * - var ist mutable
 * - Definitionen in Scala: Name ':' Typ
 */
class Hacker(val name: String, var age: Int) extends Person {
  /* Konstruktor Code */
  println("Constructing Hacker: " + toString())

  def speak(msg: String): Unit = {
    /* Semikolon ';' erlaubt aber nicht notwendig */
    val slang: String = msg.toUpperCase()
      .replace('A', '4')
      .replace('E', '3')
      .replace('T','7')
      .replace('L', '1')
    /* Wie 'System.out.println' in Java */
    println(name + "#" + slang)
  }

  /* - override um Methode zu überschreiben
   * - 'return' ist nicht notwendig. Das Resultat des letzten Ausdrucks wird zurückgegeben
   * - Geschwungene Klammern '{}' sind nicht notwendig für einen einzelnen Ausdruck
   */
  override def toString(): String =
    name + " " + age
}

/* Singleton Instanz
 * - Alle Members darin sind wie 'static' in Java
 */
object ScalaCheatSheet {
  /* main Methode (Einstiegspunkt)
   * - Array ist ein normaler Collection Typ
   * - Zwischen den eckigen Klammern stehen Typparameter (Java: List<String>)
   */
  def main(args: Array[String]): Unit = {
    /* val ist eine 'final' Variable */
    val sophia = new Hacker("Sophia", 16)

    /* age ist mutable und kann somit verändert werden */
    sophia.age = 10
    sophia.speak("MeltDown")

    /* Der Typ kann häufig inferiert werden */
    val nicolai = new Hacker("Nicolai", 8)

    val crew = List(sophia, nicolai)

    /* Aufruf von foreach mit einer anonymen Funktion */
    crew.foreach(h => h.speak("I am " + h.age + " years old" ))

    var maxAge: Int = 0
    /* Wie in Java 'for(a : as) {...}' */
    for(hacker <- crew) {
      /* if ist ein Ausdruck mit Resultat wie 'cond ? a : b' in Java */
      maxAge = if(hacker.age > maxAge) hacker.age else maxAge
    }

    var cnt = 1
    /* while Schleifen sind wie in Java */
    while(cnt <= nicolai.age) {
      println("Happy Birthday " + cnt)
      cnt += 1
    }
  }
}
```