Scala in 15 Minutes

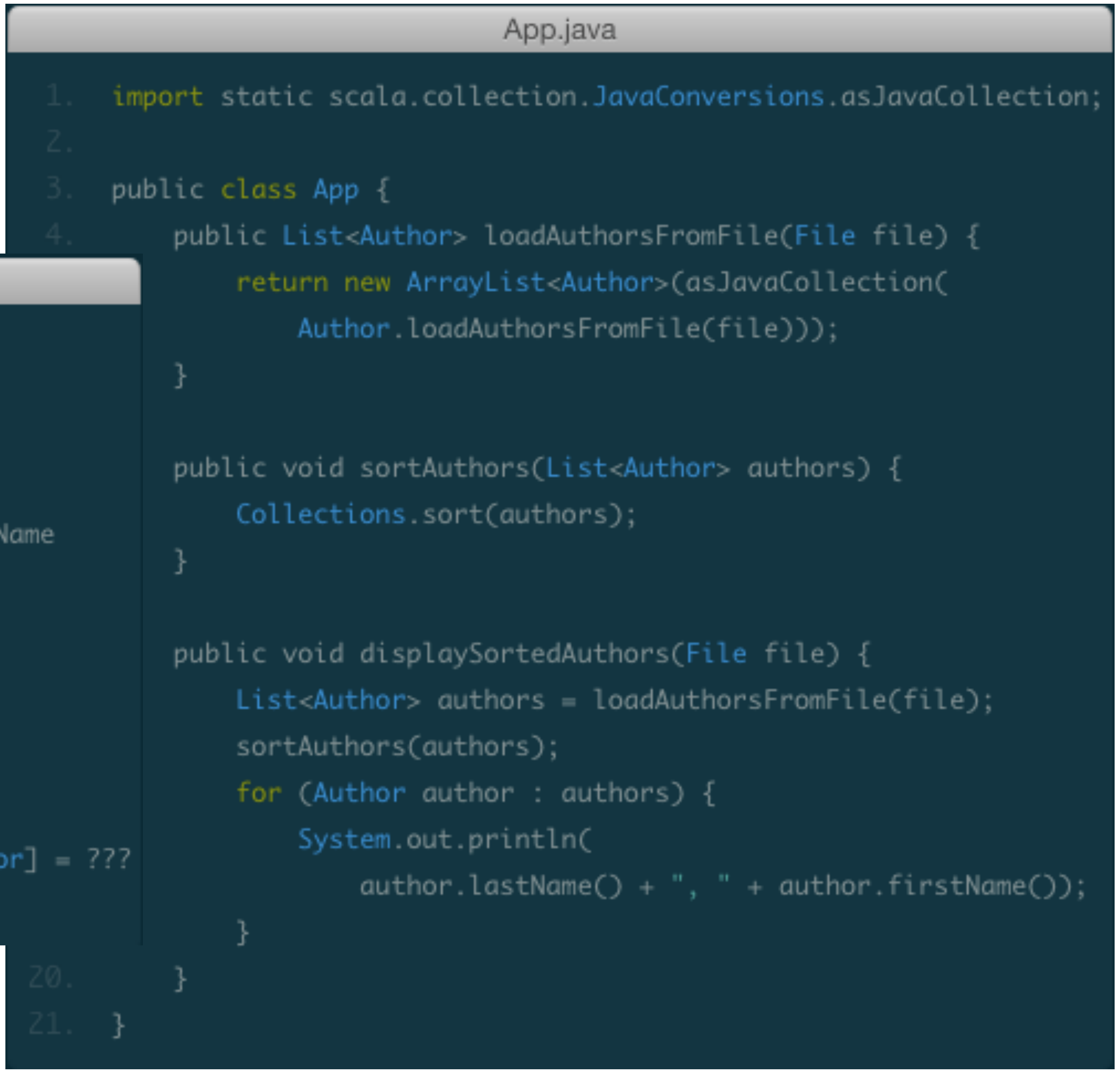
Was ist Scala?

- funktional meets objektorientiert
- steile Lernkurve
- Scalable Language

Über sich selbst

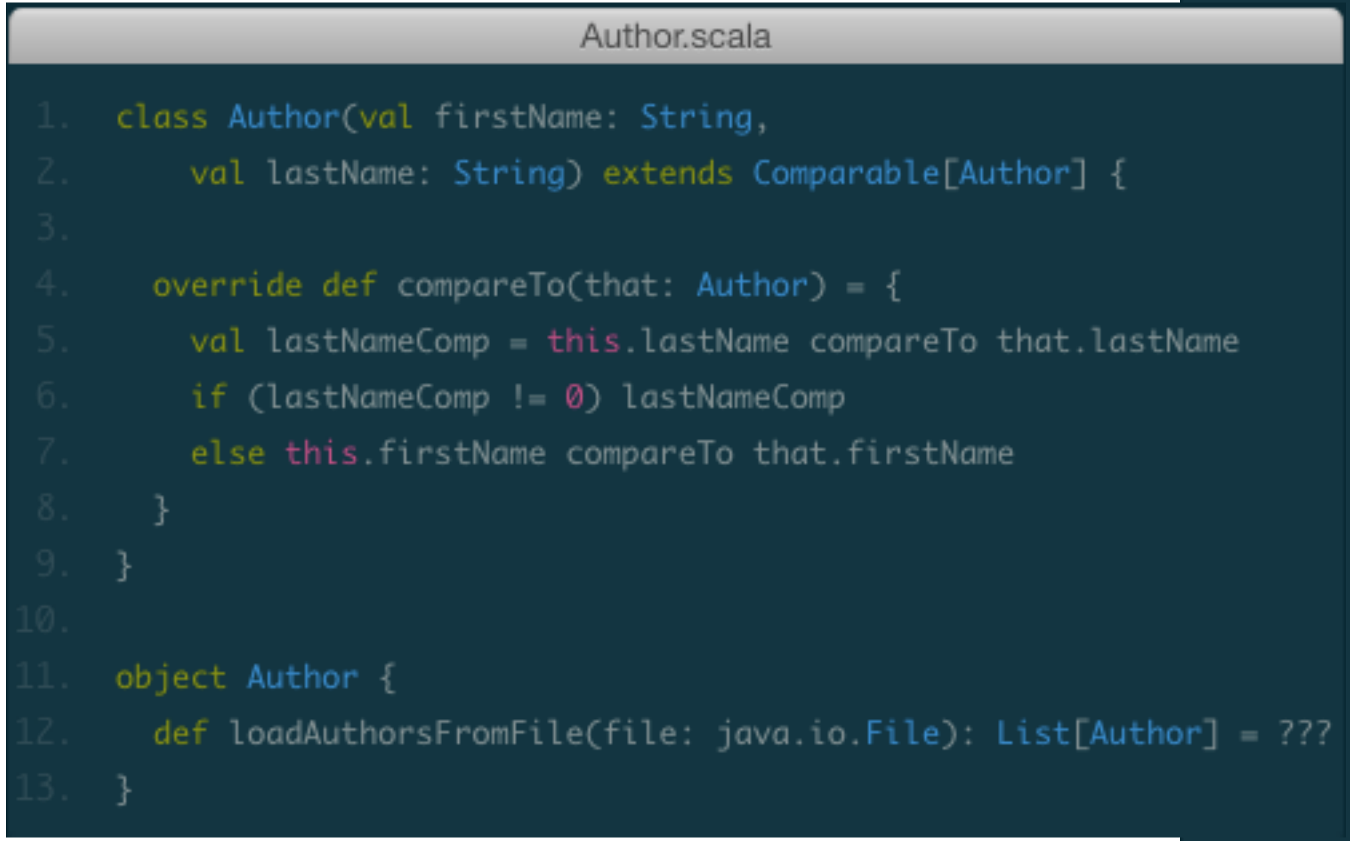
- Seamless java integration
- Traits
- Type Interference
- Concurrency & Distribution
- Pattern matching
- Higher Order Functions

Seamless Java Integration



App.java

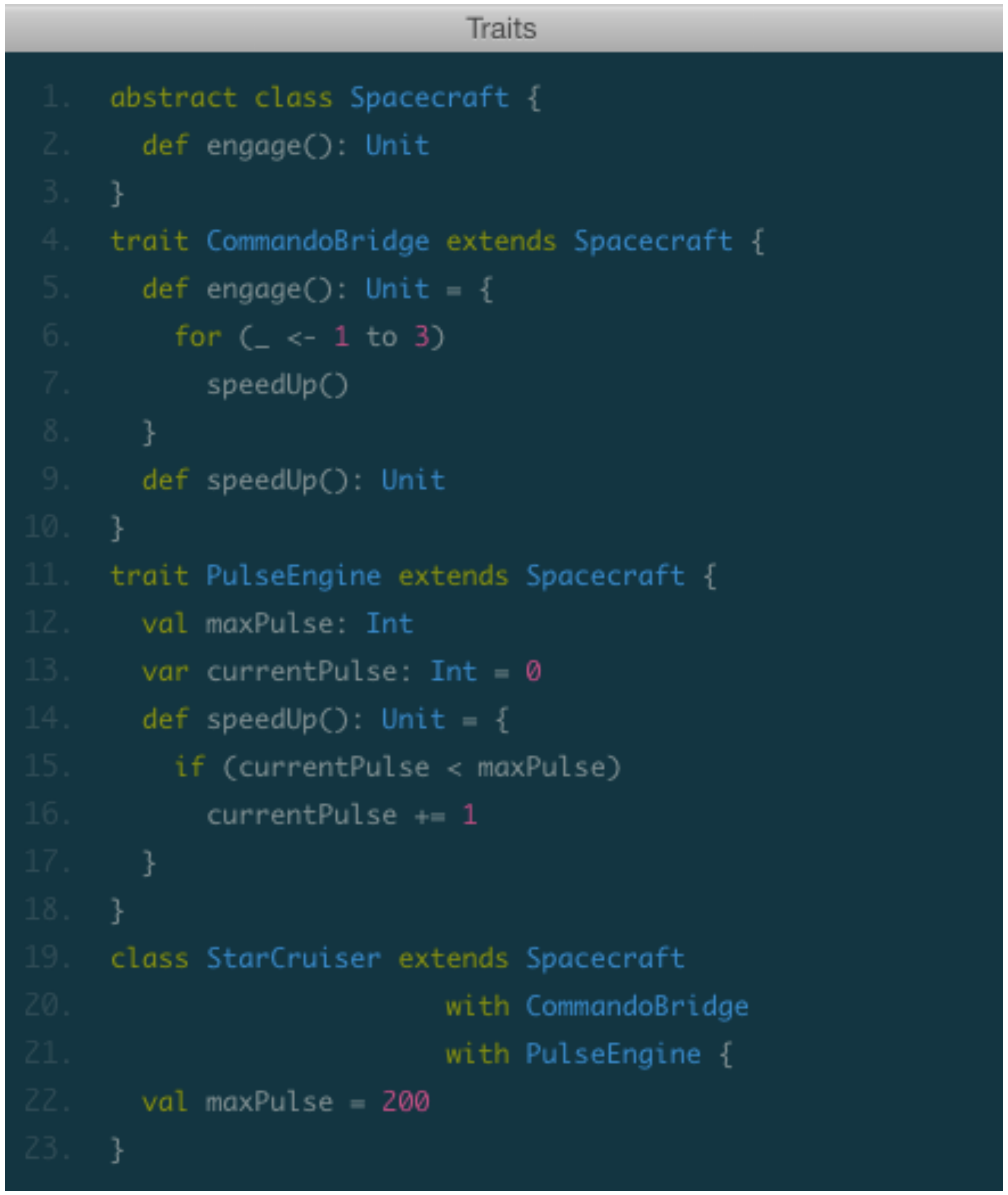
```java
import static scala.collection.JavaConversions.asJavaCollection;

public class App {
    public List<Author> loadAuthorsFromFile(File file) {
        return new ArrayList<Author>(asJavaCollection(
            Author.loadAuthorsFromFile(file)));
    }

    public void sortAuthors(List<Author> authors) {
        Collections.sort(authors);
    }

    public void displaySortedAuthors(File file) {
        List<Author> authors = loadAuthorsFromFile(file);
        sortAuthors(authors);
        for (Author author : authors) {
            System.out.println(
                author.lastName() + ", " + author.firstName());
        }
    }
}
```



Author.scala

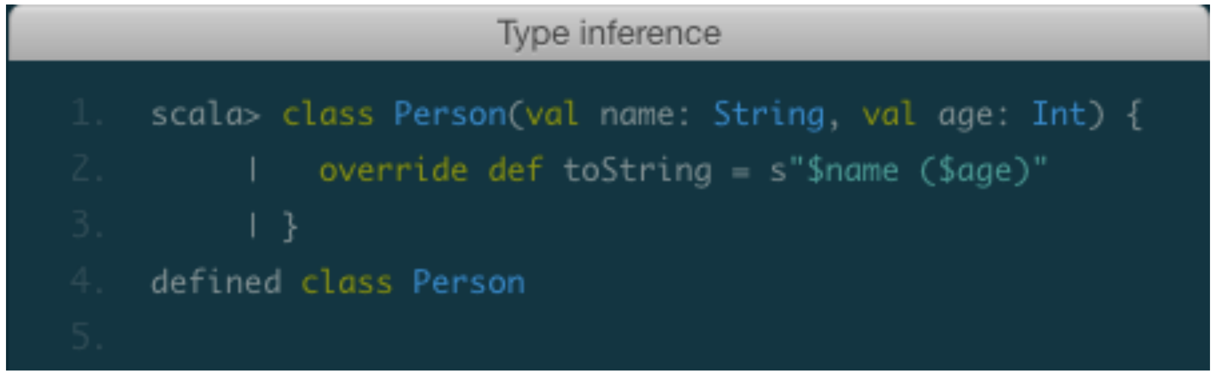
```scala
class Author(val firstName: String,
    val lastName: String) extends Comparable[Author] {

  override def compareTo(that: Author) = {
    val lastNameComp = this.lastName compareTo that.lastName
    if (lastNameComp != 0) lastNameComp
    else this.firstName compareTo that.firstName
  }
}

object Author {
  def loadAuthorsFromFile(file: java.io.File): List[Author] = ???
}
```

Traits



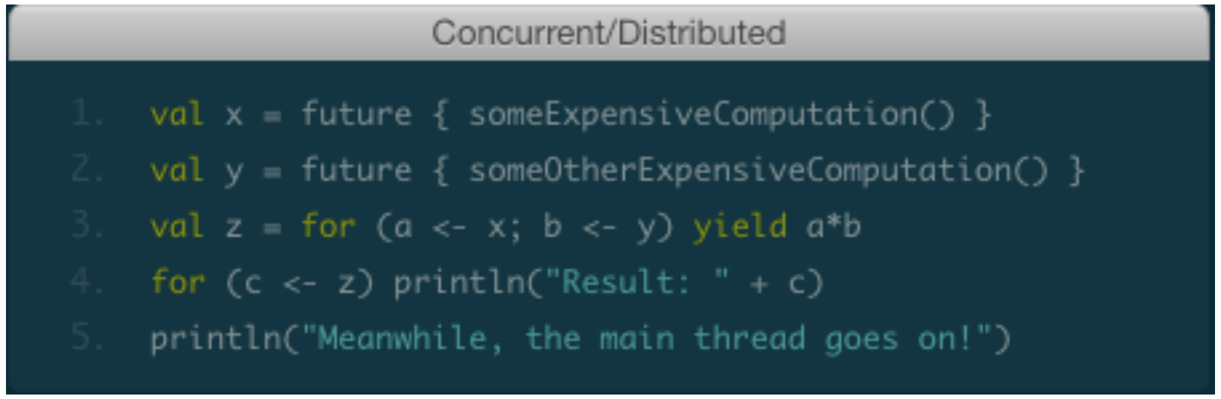
```scala
abstract class Spacecraft {
  def engage(): Unit
}
trait CommandoBridge extends Spacecraft {
  def engage(): Unit = {
    for (_ <- 1 to 3)
      speedUp()
  }
  def speedUp(): Unit
}
trait PulseEngine extends Spacecraft {
  val maxPulse: Int
  var currentPulse: Int = 0
  def speedUp(): Unit = {
    if (currentPulse < maxPulse)
      currentPulse += 1
  }
}
class StarCruiser extends Spacecraft
                     with CommandoBridge
                     with PulseEngine {
  val maxPulse = 200
}
```

Type Interferance



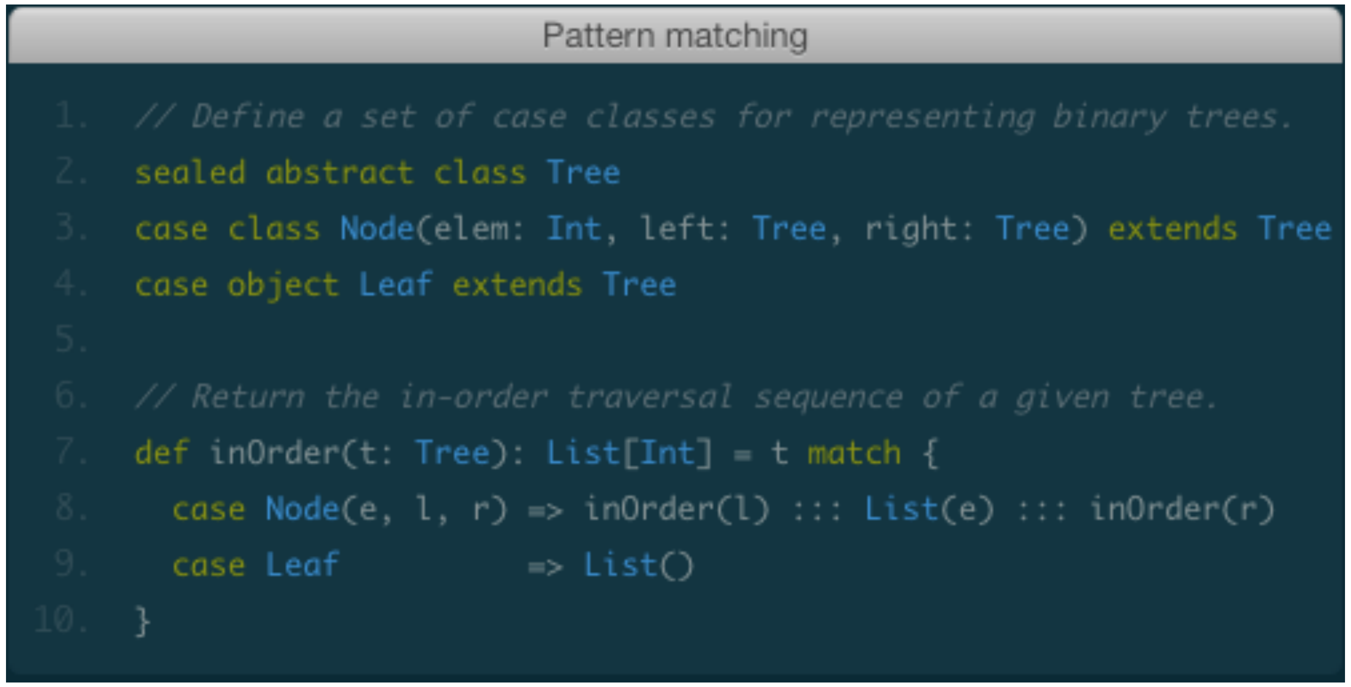
```
scala> class Person(val name: String, val age: Int) {
     |   override def toString = s"$name ($age)"
     | }
defined class Person
```

Concurrency & Distribution



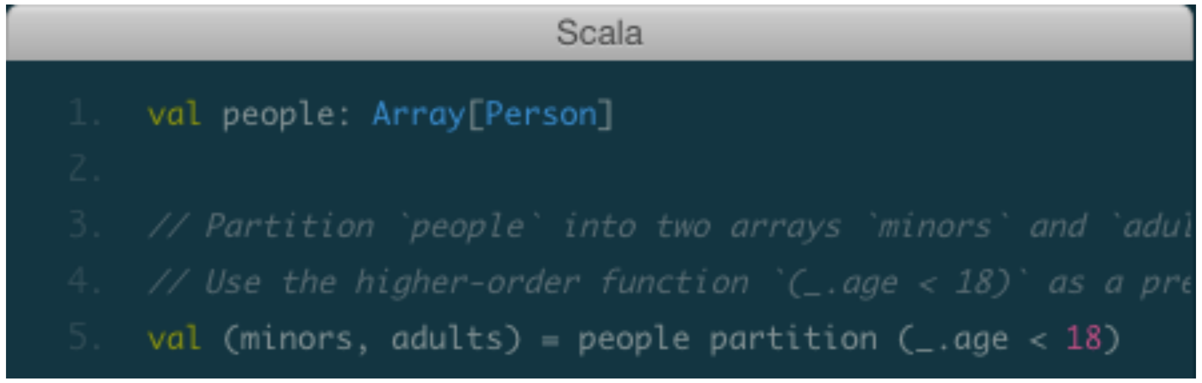
```scala
val x = future { someExpensiveComputation() }
val y = future { someOtherExpensiveComputation() }
val z = for (a <- x; b <- y) yield a*b
for (c <- z) println("Result: " + c)
println("Meanwhile, the main thread goes on!")
```

Pattern Matching

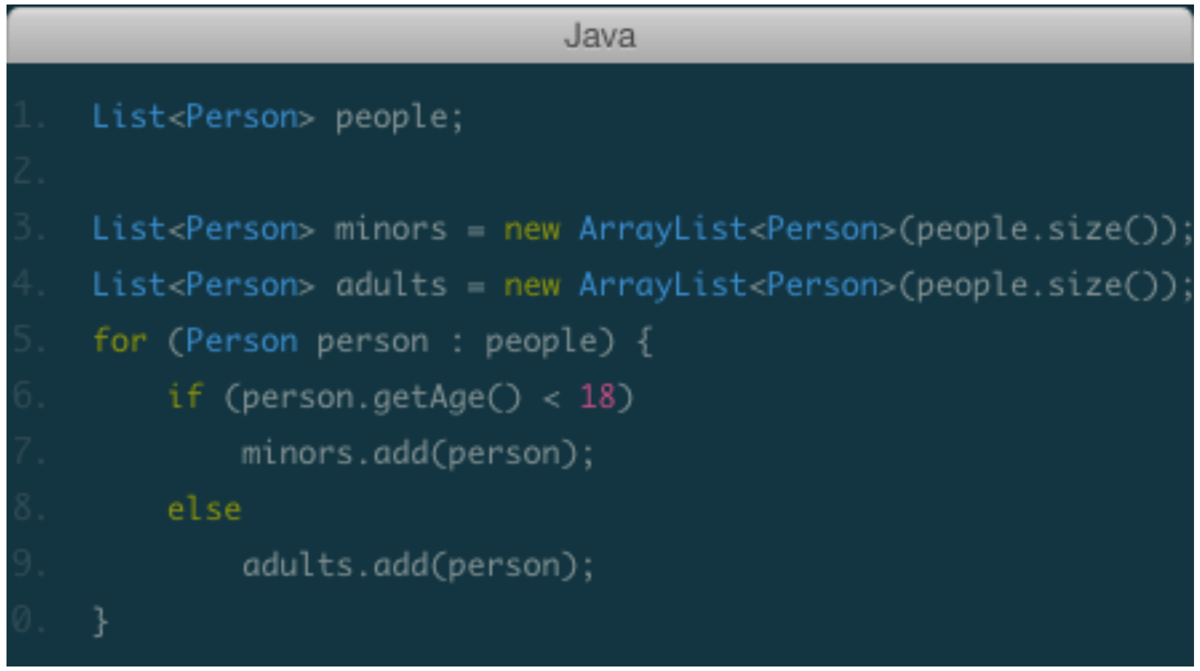

Pattern matching

```
// Define a set of case classes for representing binary trees.
sealed abstract class Tree
case class Node(elem: Int, left: Tree, right: Tree) extends Tree
case object Leaf extends Tree

// Return the in-order traversal sequence of a given tree.
def inOrder(t: Tree): List[Int] = t match {
  case Node(e, l, r) => inOrder(l) ::: List(e) ::: inOrder(r)
  case Leaf          => List()
}
```

Higher-Order Functions


Scala

```scala
val people: Array[Person]

// Partition `people` into two arrays `minors` and `adul
// Use the higher-order function `(_.age < 18)` as a pre
val (minors, adults) = people partition (_.age < 18)
```


Java

```java
List<Person> people;

List<Person> minors = new ArrayList<Person>(people.size());
List<Person> adults = new ArrayList<Person>(people.size());
for (Person person : people) {
    if (person.getAge() < 18)
        minors.add(person);
    else
        adults.add(person);
}
```

Let’s switch to the command line