Scala Basics

;

is omitted almost everywhere

Variables

Scala:

```
val s = “Hello World”

var i = 1

private var j = 3
```

Java:

```
public final String s = “Hello World”;

public int i = 1;

private int j = 3;
```

Methods

Scala:

```
def add(x: Int, y: Int): Int = {
  x + y
}  // braces are unnecessary

def sub(x: Int, y: Int) = x - y

def doSomething(text: String) { }
```

Java:

```
public int add(int x, int y) {
  return x + y;
}
public int sub(int x, int y) {
  Return x – y;
}
public void doSometing(String text) { }
```

Method Invocation

Scala:

```
myObject.myMethod(1)
myObject myMethod(1)
myObject myMethod 1

myObject.myOtherMethod(1, 2)
myObject myOtherMethod(1, 2)

myObject.myMutatingMethod()
myObject.myMutatingMethod
myObject myMutatingMethod
```

Java:

```
myObject.myMethod(1);

myObject.myOtherMethod(1, 2);

myObject.myMutatingMethod()
```

Method Overriding

Scala:

```
override def toString = ...
```

Java:

```
@Override
public String toString() {...}
```

Classes and Constructors

Scala:

```
class Person(val name: String)
```

Java:

```
public class Person {
  private final String name;
  public Person(String name)
    this.name = name;
  }
  public String getName()
    return name;
  }
}
```

Traits (= Interface + Code)

Scala:

```
trait Shape {
  def area: Double
}

class Circle extends Shape
```

Java:

```
interface Shape {
  public double area();
}

public class Circle extends Object
  implements Shape
```

No **static** in Scala

Scala:

```
object PersonUtil {
  val AgeLimit = 18
  def countPersons(persons:
    List[Person]) = ...
}
```

Java:

```
public class PersonUtil {
  public static final int AGE_LIMIT = 18;
  public static int
    countPersons(List<Person> persons) {
      ...
    }
}
```

if-then-else

Scala:

```
if (foo) {

  ...

} else if (bar) {

  ...

} else {

  ...

}
```

Java:

```
if (foo) {

  ...

} else if (bar) {

  ...

} else {

  ...

}
```

For-loops

Scala:

```
for (i <- 0 to 3) {
    ...
}

for (s <- args) println(s)
```

Java:

```
for (int i = 0; i < 4; i++) {
    ...
}

for (String s : args) {
    System.out.println(s);
}
```

While-loops

Scala:

```
while (true) {
    ...
}
```

Java:

```
while (true) {
    ...
}
```

Exceptions

Scala:

```
throw new Exception(“...”)

try { ...

} catch {

  case e: IOException => ...

} finally { ...

}
```

Java:

```
throw new Exception(“...”)

try {

} catch (IOException e) {

  ...

} finally { ...

}
```

Varargs

```
def foo(values: String*){ }

foo("bar", "baz")

val arr = Array("bar", "baz")
foo(arr: _*)
```

```
public void foo(String... values){ }

foo("bar", "baz");

String[] arr = new String[]{"bar", "baz"}
foo(arr);
```

Scala Is An Expression Language

In Scala, (almost) everything is an expression

```
val res = if (foo) x else y
val res = for (i <- 1 to 10) yield i    // List(1, ... , 10)
val res = try { e1 } catch { ...; e2 } finally { }
```

Collections – List

Scala:

```
/* val numbers = List(1, 2, 3)
   // one possible notaton */
val numbers = 1 :: 2 :: 3 :: Nil
   // another possible notation



numbers(0)
=> 1
```

Java:

```
List<Integer> numbers =
  new ArrayList<Integer>();
numbers.add(1);
numbers.add(2);
numbers.add(3);


numbers.get(0);

=> 1
```

Collections – Map

Scala:

```
var m = Map(1 -> “apple”)
m += 2 -> “orange”
/* (2 → 3) is special notation
   For the pair (2, 3) but it
   is required in most contexts
   Involvings Map and Scala
   Abbreviation */
m(1)
=> “apple”
```

Java:

```
Map<Int, String> m =
    new HashMap<Int, String>();
m.put(1, “apple”);
m.put(2, “orange”);
```

```
m.get(1);
=> apple
```

Generics

Scala:

```
List[String]
```

Java:

```
List<String>
```

Tuples

Scala:

```
val tuple: Tuple2[Int, String] =
  (1, “apple”)
// (1 → “apple”) is synonymous
val quadruple =
  (2, “orange”, 0.5d, false)
```

Java:

```
Pair<Integer, String> tuple =
  new Pair<Integer, String>(1,
    “apple”)
```

... yeah right... ;-)

Packages

Scala:

```
package mypackage
...
```

Java:

```
package mypackage;
...
```

Imports

Scala:

```
import java.util.{List, ArrayList}

import java.io._

import java.sql.{Date => SDate}
```

Java:

```
import java.util.List
import java.util.ArrayList

import java.io.*

???
```

Nice to know

Scala:

```
Console.println(“Hello”)
println(“Hello”)

val line = Console.readLine()
val line = readLine()

error(“Bad”)

1 + 1
1 .+(1)

1 == new Object
1 eq new Object

"""A\sregex""".r
```

Java:

```
System.out.println(“Hello”);

BufferedReader r = new BufferedReader(new
InputStreamRead(System.in)
String line = r.readLine();

throw new RuntimException(“Bad”)

new Integer(1).toInt() + new  Integer(1).toInt();

new Integer(1).equals(new Object());
new Integer(1) == new Object();

java.util.regex.Pattern.compile(“A\\sregex”);
```