Scala

Introduction

1

Scala

- Scala was proposed by Professor Martin Odersky and his group at EPFL in 2003 to provide a high-performance, concurrent-ready environment for
  - functional programming and
  - object-oriented programming on the Java Virtual Machine (JVM) platform
- A famous project based on Scala is Apache Spark
  - A popular framework for big data analysis

2

Main Characteristics of Scala

- Object oriented and functional
  - Everything is an object (also integers, floats, doubles, chars, etc.)
- Statically typed
- Java compatible
  - Scala complies to Java bytecode (and CLR)
  - Existing Java libraries/frameworks can be used in Scala programs

3

Scala vs Java

- Scala
  - Is more compact/less verbose than Java
    - Thanks to implicit type inference and other features
    - This is an advantage
    - … but also a disadvantage
      - Sometimes the Scala code becomes cryptic
  - Supports functional programming

4

Scala vs Java: Example

- A Java class representing a person with two attributes/variables: age and name

```
public class Person {
  private int age;  private String name;

  public Person(int age, String name) {
    this.age = age;
    this.name = name;
  }

  public int getAge() {
    return this.age;
  }

  public void setAge(int age) {
    this.age = age;
  }

  public String getName() {
    return this.name;
  }

  public void setName(String name) {
    this.name = name;
  }
}
```

5

Scala vs Java: Example

- The same code based on Scala

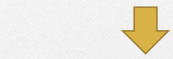
```
class Person(var age: Int, var name: String)
```

```
public class Person {
  private int age;  private String name;

  public Person(int age, String name) {
    this.age = age;
    this.name = name;
  }

  public int getAge() {
    return this.age;
  }

  public void setAge(int age) {
    this.age = age;
  }

  public String getName() {
    return this.name;
  }

  public void setName(String name) {
    this.name = name;
  }
}
```

6

Scala vs Java: Example

- The same code based on Scala



```
class Person(var age: Int, var name: String)
```

Name of the class

```
public class Person {
  private int age;  private String name;

  public Person(int age, String name) {
    this.age = age;
    this.name = name;
  }

  public int getAge() {
    return this.age;
  }

  public void setAge(int age) {
    this.age = age;
  }

  public String getName() {
    return this.name;
  }

  public void setName(String name) {
    this.name = name;
  }
}
```

7

Scala vs Java: Example

- The same code based on Scala

```
class Person(var age: Int, var name: String)
```

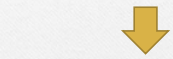
Definition of the two attributes/variables of the class and signature of the primary constructor

```
public class Person {
  private int age;  private String name;

  public Person(int age, String name) {
    this.age = age;
    this.name = name;
  }

  public int getAge() {
    return this.age;
  }

  public void setAge(int age) {
    this.age = age;
  }

  public String getName() {
    return this.name;
  }

  public void setName(String name) {
    this.name = name;
  }
}
```

8

Scala vs Java: Example

- The same code based on Scala



```
class Person(var age: Int, var name: String)
```

Definition of the two attributes/variables of the class and signature of the primary constructor

Also the gets and sets methods are automatically defined

```
public class Person {
  private int age;  private String name;

  public Person(int age, String name) {
    this.age = age;
    this.name = name;
  }

  public int getAge() {
    return this.age;
  }

  public void setAge(int age) {
    this.age = age;
  }

  public String getName() {
    return this.name;
  }

  public void setName(String name) {
    this.name = name;
  }
}
```

9

Scala vs Java: Example

- Split a list of persons in two sub lists
  - Kids (age<18)
  - Adults (age>=18)

```
List<Person> persons = ...

List<Person> adults = new
LinkedList<Person>();

List<Person> kids = new
LinkedList<Person>();

for (Person person : persons) {
  if (person.getAge() < 18) {
    kids.add(person);
  } else {
    adults.add(person);
  }
}
```

10

Scala vs Java: Example

- The same code in Scala

```
val person = …
val (kids, adults) =
  persons.partition(_.age < 18)
```

```
List<Person> persons = ...
List<Person> adults = new
LinkedList<Person>();
List<Person> kids = new
LinkedList<Person>();
for (Person person : persons) {
  if (person.getAge() < 18) {
    kids.add(person);
  } else {
    adults.add(person);
  }
}
```

11

Scala vs Java: Example

- The same code in Scala

```
val person = …
val (kids, adults) =
  persons.partition(_.age < 18)
```

Split condition

```
List<Person> persons = ...
List<Person> adults = new
LinkedList<Person>();
List<Person> kids = new
LinkedList<Person>();
for (Person person : persons) {
  if (person.getAge() < 18) {
    kids.add(person);
  } else {
    adults.add(person);
  }
}
```

12

Scala vs Java: Example

- The same code in Scala

```
val person = …
val (kids, adults) =
  persons.partition(_.age < 18)
```

Definition of the sub lists storing the result of the partition

```
List<Person> persons = ...
List<Person> adults = new
LinkedList<Person>();
List<Person> kids = new
LinkedList<Person>();
for (Person person : persons) {
  if (person.getAge() < 18) {
    kids.add(person);
  } else {
    adults.add(person);
  }
}
```

13

Scala: Example of cryptic code

```
val myList = List(1, 2, 3)
val res = (10/:myList)(_+_)
```

14

Scala: Example of cryptic code

```
val myList = List(1, 2, 3)
val res = (10/:myList)(_+_)
```

Definition of a list of integers

15

Scala: Example of cryptic code

```
val myList = List(1, 2, 3)
val res = (10/:myList)(_+_)
```

?????

16

Scala: Example of cryptic code

```
val myList = List(1, 2, 3)
val res = (10/:myList)(_+_)
```

What is the data type of res?

What is the content of res ?

17

Command line shell and IDE

18

REPL - Read eval print loop

- REPL is
  - A command line shell for on-the-fly execution of Scala statements
  - Run bin/scala to open the command line shell

19

IDE

- Several IDEs have ad-hoc plugins for Scala
  - Netbeans
  - IDEA
  - Eclipse

20