PEP Scala (1)

Email:	christian.urban at kcl.ac.uk
Office:	N7.07 (North Wing, Bush House)
Slides & Code:	KEATS
Scala Office Hours:	Thursdays 11 – 13

Why Scala?


twitter


LinkedIn


Morgan Stanley


CREDIT SUISSE

...


EDF
ENERGY


Novell


foursquare


HSBC

...

Why Scala?

- compiles to the JVM
  (also JavaScript, native X86 in the works)
- integrates seamlessly with Java
- combines **functional** and **object-oriented** programming
- it is a bit on the "mathematical" side
  (no pointers, no `null`)
- often one can write very concise and elegant code

alternatives:
Elm, Haskell, Ocaml, F♯, *Erlang*, *ML*, *Lisp* (*Racket*), . . .

Java vs Scala

Java

```
public class Point {
  private final int x, y;

  public Point(int x, int y) {
    this.x = x;
    this.y = y;
  }

  public int x() { return x; }

  public int y() { return y; }
}
```

Scala

```
class Point(val x: Int, val y: Int)
```

First Steps: Scala Tools

- there is a plugin for Eclipse (called Scala IDE)
- there is also a plugin for IntelliJ
- there is a worksheet mode in Eclipse and IntelliJ
- I use Sublime or venerable Emacs ;o)

Why Scala?

Scala, Elm, Haskell, Ocaml, F♯, *Erlang*, *ML*, *Lisp* (*Racket*), . . .

Why Functional Programming?

Scala, Elm, Haskell, Ocaml, F♯, *Erlang*, *ML*, *Lisp* (*Racket*), . . .

Why Functional Programming?

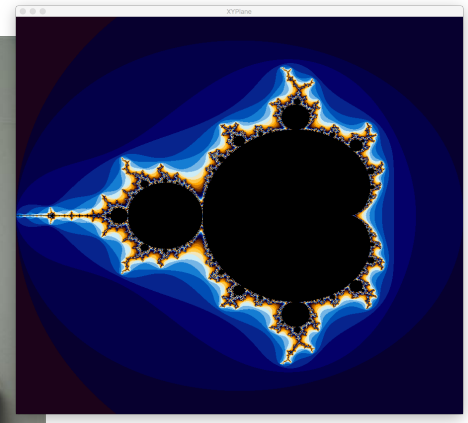
"If you want to see which features will be in mainstream programming languages tomorrow, then take a look at functional programming languages today."

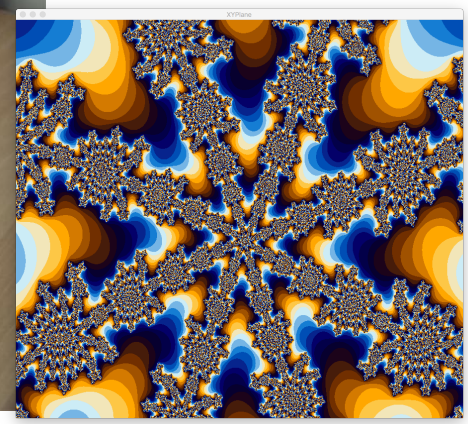
—Simon Peyton Jones (works at Microsoft)
main developer of the Glasgow Haskell Compiler

Scala, Elm, Haskell, Ocaml, F♯, *Erlang*, *ML*, *Lisp* (*Racket*), . . .

1986

3 days





64K RAM, no HD, no monitor, lots of cables

1986

1988, C

1986

1988, C

1992, Linux

1986

1988, C

1992, Linux

1996

1986

1988, C

1992, Linux

2000

1996

1986

1988, C

1992, Linux

1996

2000

2012?

1986

1988, C

1992, Linux

1996

2000

2012?

2017

1986

1988, C

1992, Linux

1996

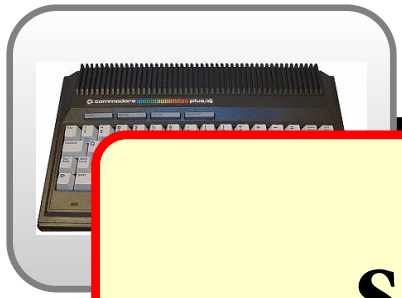
2000

2012?

2017

1986: no Internet
no Amazon
no FB, no mobiles,...

1986



1988, C

1992, Linux

2012?

Speedup by Moore's Law

1986:	3 days	**1996:**	135 mins
1988:	1.5 days	**1998:**	67 mins
1990:	18 hs	**2000:**	33 mins
1992:	9 hs	**2002:**	16 mins
1994:	4.5 hs		???

Every two years, computers got twice as powerful.

2017

no Amazon

no FB, no mobiles,...

1986

1988, C

1992, Linux

1996

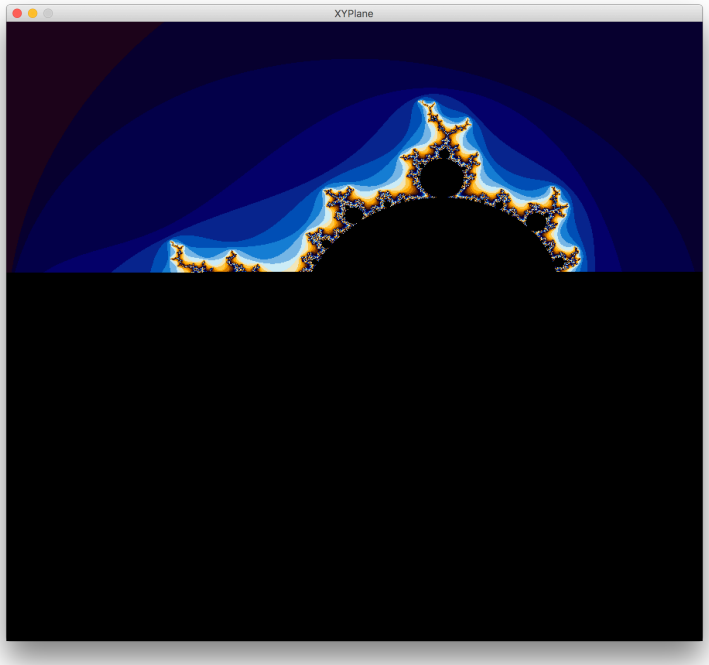
2000

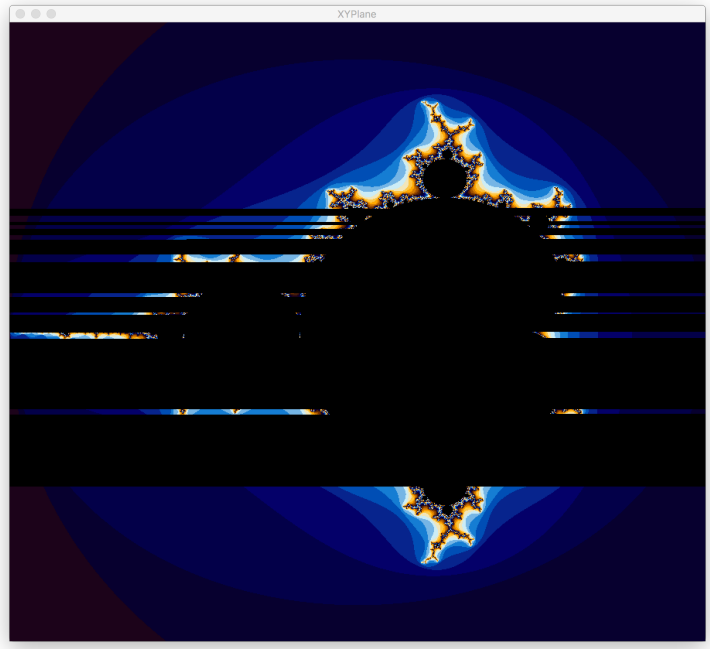
2012?

2017

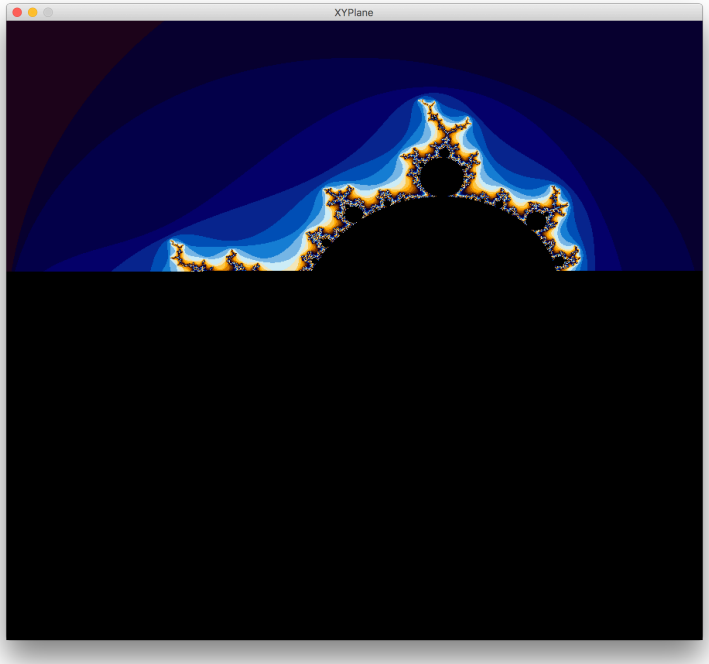
1986: no Internet
no Amazon
no FB, no mobiles,...

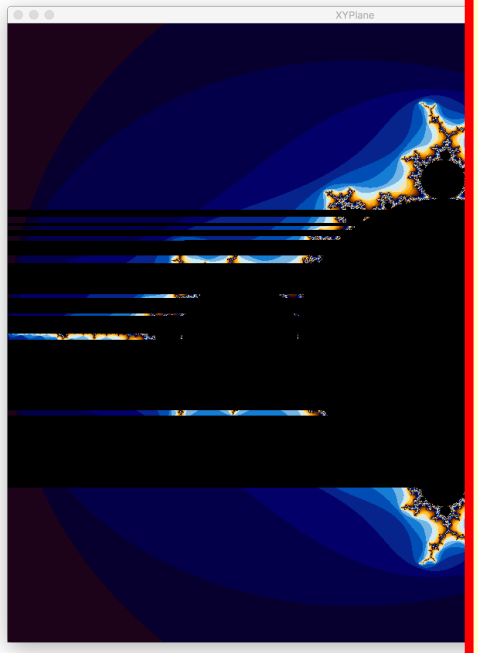
Seq vs Par



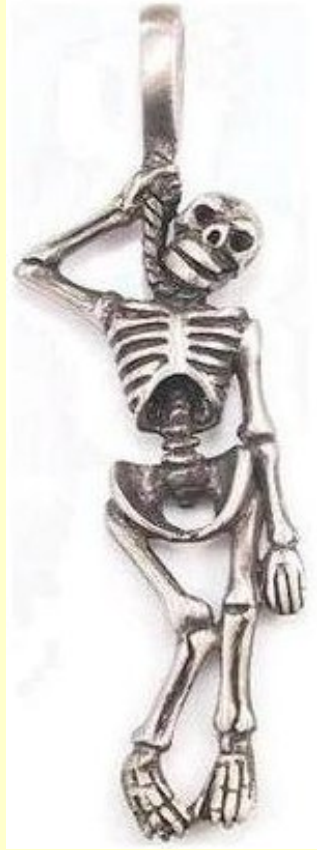


Seq vs Par

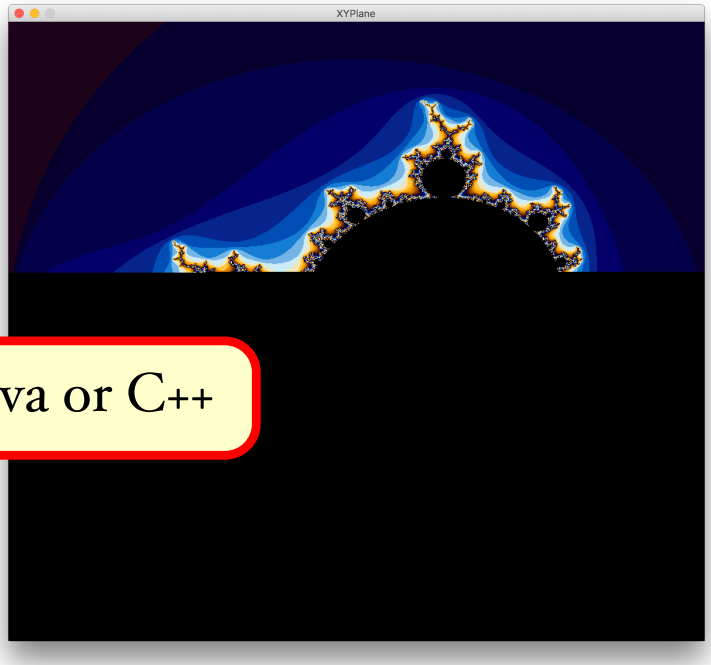




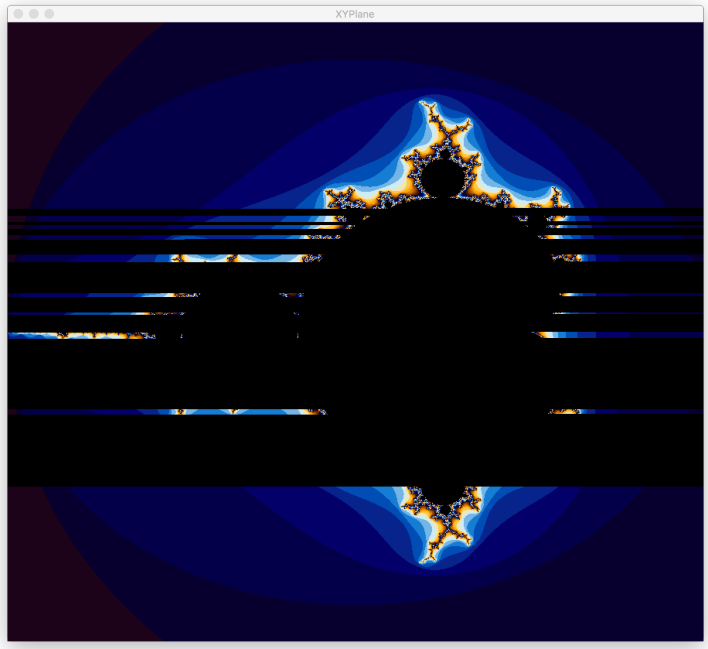
in Java or C++



Seq vs Par



in Java or C++



Types

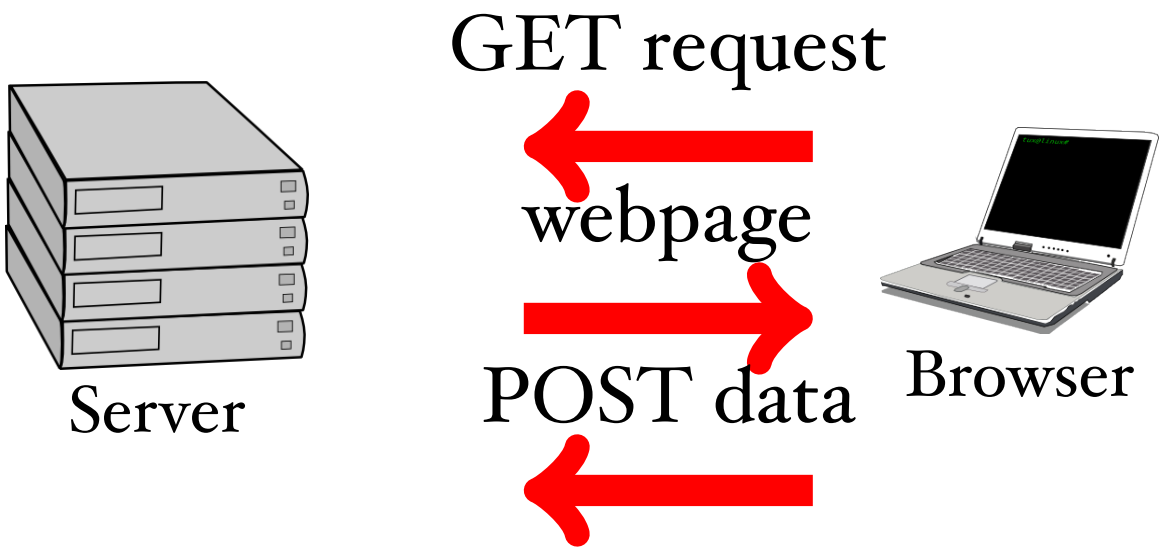
- Base types

  `Int`, `Long`, `BigInt`, `Float`, `Double`
  `String`, `Char`
  `Boolean`

- Compound types

`List[Int]`	lists of Int's
`Set[Double]`	sets of Double's
`(Int, String)`	Int-String pair
`List[(BigInt, String)]`	lists of BigInt-String pairs
`List[List[Int]]`	list of lists of Int's

An Http Request



```
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Scanner;

public class URLReader {

    public static String readURL(String sUrl) {
        StringBuilder buf = new StringBuilder();
        Scanner in = null;

        try {
            URL url = new URL(sUrl);
            in = new Scanner(url.openStream());

            while (in.hasNextLine()) {
                buf.append(in.nextLine() + "\n");
            }
            return buf.toString();

        } catch (MalformedURLException e) {
            System.err.println(e);
        } catch (IOException e) {
            System.err.println(e);
        } finally {
            if (in != null) {
                in.close();
            }
        }
        return null;
    }
}
```

```
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Scanner;

public class URLReader {

    public static String readURL(String sUrl) {
        StringBuilder buf = new StringBuilder();
        Scanner in = null;

        try {
            URL url = new URL(sUrl);
            in = new Scanner(url.openStream());

            while (in.hasNextLine()) {
                buf.append(in.nextLine() + "\n");
            }
            return buf.toString();

        } catch (MalformedURLException e) {
            System.err.println(e);
        } catch (IOException e) {
            System.err.println(e);
        } finally {
            if (in != null) {
                in.close();
            }
        }
        return null;
    }
}
```

Conclusion

- Scala is still under heavy development (the compiler is terribly slow)
- `http://www.scala-lang.org/`
- it is a rather **deep** language...i.e. gives you a lot of rope to shoot yourself
- hope you have fun with the coursework

Questions?