Test.scala

```scala
package com.crazyjvm.week3

/**
 * Created by chenchao on 14-6-10.
 */
object LocalFunction extends App{

  def add3(x : Int, y : Int, z : Int ) : Int = {
    def add2(x : Int, y : Int) : Int = {
      x + y
    }
    add2(add2(x,y),z)
  }

  println(add3(1,2,3))
}
```

嵌套函数

Def 里有 def

---

ek4 Bounds.scala

Bounds.scala × Comparable.java ×

```scala
package com.crazyjvm.week4

/**
 * Created by chenchao on 14-6-17.
 */
object Bounds extends App{
//  val p = new Pair(1,2)
//  println(p.smaller)

  val s1 = new Student("A")
  val s2 = new Student("B")

  val pair1 = new Pair(s1,s2) //Pair[Student,Student]

  val p1 = new Person("AA")
  println(pair1.replaceFirst(p1))
}

class Pair[T] (val first : T, val second : T){
  //def smaller = if(first.compareTo(second) < 0) first else second

  def replaceFirst[R >: T](newFirst : R) = new Pair(newFirst,second)

}

class Person(val name : String)

class Student(name : String) extends Person(name)
```

P246

泛型例子 2

Person 是 Student 的超类

Bounds.scala × Comparable.java × RichInt.scala ×

```scala
object Bounds extends App{
//  val p = new Pair("A","B")
//  println(p.smaller)

  val s1 = new Student("A")
//  val s2 = new Student("B")
 //
 //  val pair1 = new Pair(s1,s2) //Pair[Student,Student]
//
  val p1 = new Person("AA")
 //  println(pair1.replaceFirst(p1))

  new X(s1)
  new X(p1)

}

class Pair[T <% Comparable[T]] (val first : T, val second : T){
  def smaller = if(first.compareTo(second) < 0) first else second

  //def replaceFirst[R >: T](newFirst : R) = new Pair(newFirst,second)

}

class Person(val name : String)

class Student(name : String) extends Person(name)

//T <% V ------>  T到V的隐式转换
//[T:M]  ------>  M[T]


class X[-T](val x : T){

}
```

型变

\+ 协变

\- 逆变

隐式转换

Function1.scala × Bounds.scala × ImplicitDemo.scala × Comparable.ja

```scala
package com.crazyjvm.week4

import java.io.File
import scala.io.Source

/**
 * Created by chenchao on 14-6-17.
 */
object ImplicitDemo extends App{

  import Context._
  new File("xxx").read

}

class RichFile(val file : File){
  def read = Source.fromFile(file.getPath).mkString
}

object Context{
  implicit def file2RichFile(f : File) = new RichFile(f)
}
```

Import ~

Function1.scala × Bounds.scala × ImplicitDemo.scala × Compar

```scala
//  new File("xxx").read
  //import Context._
  //AAA.print("Jack")
  //println(1.add2)

  val p1 = new Pair2(1,2)
  val p2 = new Pair2("A","B")

  println(p1.smaller)
  println(p2.smaller)
}

class Pair2[T : Ordering](val first : T, val second : T){
  def smaller(implicit ord : Ordering[T]) =
    if (ord.compare(first,second) < 0 ) first else second
}

class Line(val len : Double){
  override def toString() = "Length of lone : " + len
}
```

上下文界定 1

P330

ek4 ImplicitDemo.scala

Function1.scala × Bounds.scala × ImplicitDemo.scala × Comparable

```scala
//  import Context._
//  new File("xxx").read
  //import Context._
  //AAA.print("Jack")
  //println(1.add2)

  implicit object Line extends LineOrdering
  val p1 = new Pair2(1,2)
  val p2 = new Pair2("A","B")

  val l1 = new Line(1)
  val l2 = new Line(2)

  val p3 = new Pair2[Line](l1,l2)

  println(p1.smaller)
  println(p2.smaller)
  println(p3.smaller)
}

class Pair2[T : Ordering](val first : T, val second : T){
  def smaller(implicit ord : Ordering[T]) =
    if (ord.compare(first,second) < 0 ) first else second
}

trait LineOrdering extends Ordering[Line]{
  override def compare(x : Line, y : Line) = {
    if(x.len < y.len) -1
    else if(x.len == y.len) 0
    else 1
  }
}

class Line(val len : Double){
  override def toString() = "Length of line : " + len
}
```

上下文界定 2