



# **DATA STRUCTURE FOR RECURSION**

# LIST

- Immutable
- Linked list

```
object ListExample {  
    val myList: List[Int] = List()  
    val listNum = List(1, 2, 3, 4, 5)  
    val listStr: List[String] = List("John", "Robin", "Richard")  
  
    def main(args: Array[String]): Unit = {  
        println(myList)  
        println(listNum)  
        println(listStr)  
    }  
}
```

```
List()  
List(1, 2, 3, 4, 5)  
List(John, Robin, Richard)
```

# LIST ACCESS

```
object ListAccess {  
  val myList: List[Int] = List()  
  val listNum = List(1, 2, 3, 4, 5)  
  val listStr: List[String] = List("John", "Robin", "Richard")  
  
  def main(args: Array[String]): Unit = {  
    println(listStr(0))  
    println(listStr(1))  
    println(listStr(2))  
    println(listStr(3))  
  }  
}
```

```
listStr(2) = "DD"
```



not compile  
immutable

```
John  
Robin  
Richard
```

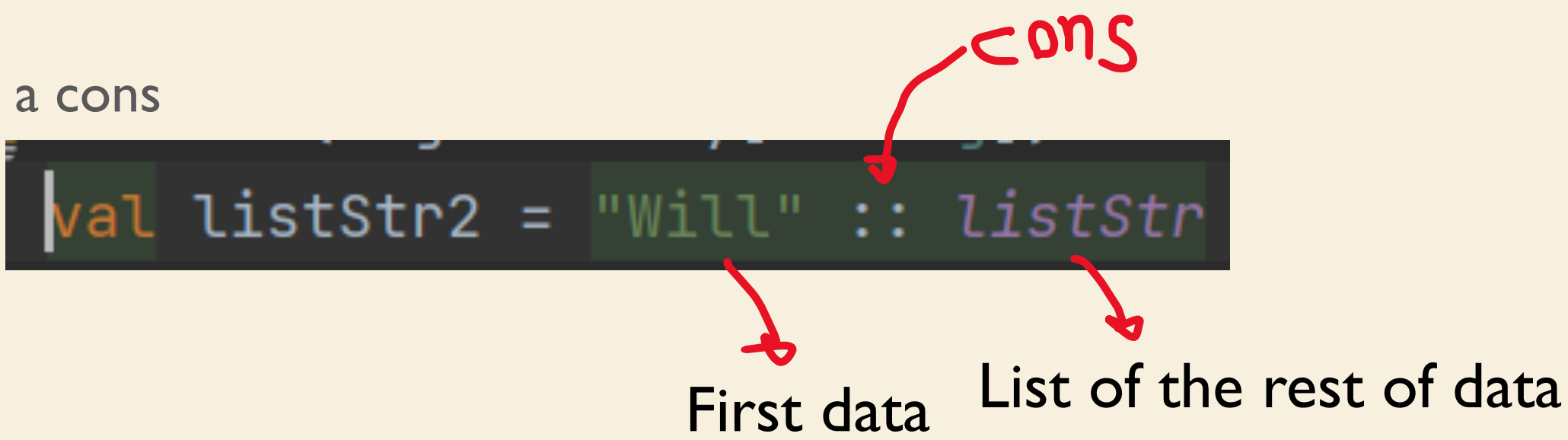
```
Exception in thread "main" java.lang.IndexOutOfBoundsException: Cre  
    at scala.collection.LinearSeqOps.apply(LinearSeq.scala:117)  
    at scala.collection.LinearSeqOps.apply$(LinearSeq.scala:114)  
    at scala.collection.immutable.List.apply(List.scala:79)
```

# HOW TO DEFINE A LIST?

```
val listStr: List[String] = List("John", "Robin", "Richard")
```

- Use a cons

```
val listStr2 = "Will" :: listStr
```



First data

List of the rest of data

```
val listNum2 = 9 :: 6 :: 17 :: Nil
```

```
List(9, 6, 17)
```

Anything in front or  
between it must be a data.

```
val listNum = List(1, 2, 3, 4, 5)
```

```
val listNum2 = 9 :: 6 :: 17 :: Nil
```

```
println(listNum ++ listNum2)
```

```
List(1, 2, 3, 4, 5, 9, 6, 17)
```

# LIST METHODS

```
object ListMethods {  
  val myList: List[Int] = List()  
  val listNum = List(1, 2, 3, 4, 5)  
  val listStr: List[String] = List("John", "Robin", "Richard")  
  
  def main(args: Array[String]): Unit = {  
    println(listStr.head)  
    println(listNum.tail)  
    println(myList.isEmpty)  
    println(listNum.reverse)  
    println(List.fill(10)(1))  
    println(listStr.max)  
  }  
}
```

```
John  
List(2, 3, 4, 5)  
true  
List(5, 4, 3, 2, 1)  
List(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)  
Robin
```

# EXERCISE (ONLY ISEMPY, LENGTH, HEAD, TAIL, ::, ++ AVAILABLE)

```
def member(x:Any , l :List[Any]): Boolean = {
```

```
def sorted(l: List[Int]): Boolean = {
```

```
def delete(x:Any,l:List[Any]):List[Any] = {
```

```
def length(l:List[Any]):Int = {
```

# EXERCISE - CONT

```
def myReverse(l: List[Any]): List[Any] = {
```

```
def dot(l1: List[Int], l2: List[Int]): Int = {
```

```
def max(l: List[Int]): Int = {
```

```
def setify(l: List[Any]): List[Any] = {
```



# LIST ITERATION

```
def main(args: Array[String]): Unit = {  
    println(listNum.foreach(println))  
  
    for(name <- listStr){  
        println(name)  
    }  
  
    var sum = 0  
    listNum.foreach(sum += _)  
    println(sum)  
  
    println(listNum(4))  
    // println(listNum(5)) IndexOutOfBoundsException
```

```
1  
2  
3  
4  
5  
( )  
John  
Robin  
Richard  
15  
5
```

# ITERATE TO MODIFY A LIST?

- Cannot be done because list is immutable.
- We have to produce a new list.

```
def add(s:List[Int], a:Int): List[Int] = {  
  if(s.isEmpty) {  
    return List()  
  }  
  
  (s.head+a) :: add(s.tail,a)  
}
```

```
println(add(listNum,10))
```

```
List(11, 12, 13, 14, 15)
```

# HIGHER ORDER METHODS

## MAP

```
object MyMapOnList {  
  val myList: List[Int] = List()  
  val listNum = List(1, 2, 3, 4, 5)  
  val listStr: List[String] = List("John", "Robin", "Richard")  
  
  def addCurry(x: Int): Int => Int = {  
    (y: Int) => x + y  
  }  
  
  def main(args: Array[String]): Unit = {  
    println(listNum.map(_ * 2))  
    println(listNum.map(x => x * 2))  
    println(listNum.map(addCurry(100)(_)))  
  }  
}
```

List(2, 4, 6, 8, 10)

List(2, 4, 6, 8, 10)

List(101, 102, 103, 104, 105)

# FLATTEN

```
object Flatten {  
  val myList: List[Int] = List()  
  val listNum = List(1, 2, 3, 4, 5)  
  val listNum2 = List(10, 20, 30, 40, 50)  
  val listStr: List[String] = List("John", "Robin", "Richard")  
  
  def addCurry(x:Int): Int => Int = {  
    (y:Int) => x+y  
  }  
  
  def main(args: Array[String]): Unit = {  
    println(List(listNum, listNum2))  
    println(List(listNum, listNum2).flatten)  
  }  
}
```

```
List(List(1, 2, 3, 4, 5), List(10, 20, 30, 40, 50))  
List(1, 2, 3, 4, 5, 10, 20, 30, 40, 50)
```

# FILTER

```
object Filter {  
    val myList: List[Int] = List()  
    val listNum = List(1, 2, 3, 4, 5)  
    val listNum2 = List(10, 20, 30, 40, 50)  
    val listStr: List[String] = List("John", "Robin", "Richard")  
  
    def main(args: Array[String]): Unit = {  
        println(listNum.filter(x => x%2 ==0))  
    }  
}
```

List(2, 4)