الغلاف الخارجى للبحث

A close up of a logo

Description automatically generated

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| أولاً: البيانات الخاصة بالطالب | | | | | | | | | |
| **الفرقة الدراسية** | **الثانية منقول** | | | **التخصص** | | | **عام** | | |
| **اسم القسم** | **عام** | | | | | | | | |
| **اسم المقرر** | **لغات الحاسب – 3 | programming languages -3** | | | | | | | | |
| **استاذ المقرر** | **د/مروة عبد الفتاح** | | | | | | | | |
| ثانياً: البيانات الخاصة بالبحث | | | | | | | | | |
| **عنوان البحث** | **Comparing between Haskell and Scala** | | | | | | | | |
| **طبيعة المشاركة** | **بحث فردى**  ✓ | | | | **بحث جماعى** | | | | |
| **ارسال البحث** | **بواسطة البريد الالكتروني** | | | | | | | | |
| **اسماء الطلاب المشاركين فى البحث**  **(يكتب الاسم رباعيا)** | **م** | **الاسم رباعى** | | | **رقم الجلوس** | | | | **الرقم القومى** |
| **1** | **سيف هشام سالم سيد** | | | **20180284** | | | | **30008042100434** |
| **2** |  | | |  | | | |  |
| **3** |  | | |  | | | |  |
| **4** |  | | |  | | | |  |
| **5** |  | | |  | | | |  |
| **تاريخ الإرسال** | **1 / 6 / 2020** | | | | | | | | |
| ثالثاً: البيانات الخاصة بالكونترول | | | | | | | | | |
| **النتيجة** | | | **ناجح** | | | **راسب** | | | |
| **أعضاء لجنة تقييم البحث** | **الاسماء** | | | | | | | **التوقيع** | |
| **1** |  | | | | | |  | |
| **2** |  | | | | | |  | |
| **3** |  | | | | | |  | |

|  |  |
| --- | --- |
| **فى حالة عدم قبول البحث يرجى ذكر الأسباب** | * **..............................................................................................................................................** * **..............................................................................................................................................** * **..............................................................................................................................................** * **.............................................................................................................................................** |

Haskell and Scala, A brief comparison!

# Introduction:

* Haskell is a statically typed programming language with strictly functional programming functionality with lazy evaluation and inference styles.
* This was created and developed by Lennart Augustsson, John Hughes, Paul Hudak, John Launchbury, Philip Wadler, Simon Peyton Jones, and Erik Meijer.
* The typing discipline is solid, static, assumed, and does not have strict Semantics.
* It had been licensed under the BSD certificate of Clause 3. Its characteristics are lazy programming and is non-strict and flexible. The first release was in 2010.
* Scala is also a general-purpose programming language, provides the features of functional programming and had a strong static type system which had object-oriented features.
* Scala was written and created using Java, and it uses Java virtual machine (JVM) to compile its source codes and it runs on the JVM.
* So that it gives it a great advantage to run anywhere. It was licensed under Clause 3 BSD license.
* It was designed by Martin Odersky. It was developed in “EPFL” in Switzerland.

So, we are going to discuss differences between Haskell and Scala to show up the differences between them.

# Syntax and Datatypes:

In Haskell:

* while the syntax is strong and static so every expression and function in Haskell has a *type.* For example, the value True has the type Bool, while the value “foo” has the type string.
* Static type system means that the compiler knows the type of every value and expression at compile time before any code is executed.
* Type inference means that compiler can automatically deduce the types of almost all expressions in a program.
* Some of basic datatypes in Haskell: Char, Bool, Int, Integer, Double

and there is another useful composite data types like lists and Tuples

ghci> :type “a”

"a" :: [Char]

ghci> :type [1,2,3,4]

[5,2,4] :: Num a => [a]

ghci> head [1,2,3,4]

1

ghci> :type (99, 88)

(99, 88) :: (Num a, Num b) => (a, b)

ghci> fst (99, 88)

99

* To Define functions in Haskell it easy, we type first the type of the function and its arguments and body of function is written in that way:

square :: Integer -> Integer

square x = x \* x

* Pattern matching consists of specifying patterns to which some data should comply and checking to see whether they do, deconstructing the data according to these patterns and defining separate function bodies for different patterns. You can design any form of data starting with the numbers, characters, lists, tuples and so on, one of most useful examples in how to use pattern matching.

*-- To sum vectors, it very easy to perform using patter matching*

AddVectors :: (Num a) => (a, a) -> (a, a) -> (a, a)

AddVectors (a1, b1) (a2, b2) = (a1 + a2, b1 + b2)

*-- Another example in how to use pattern matching*

SumList :: [Int] -> [Int]

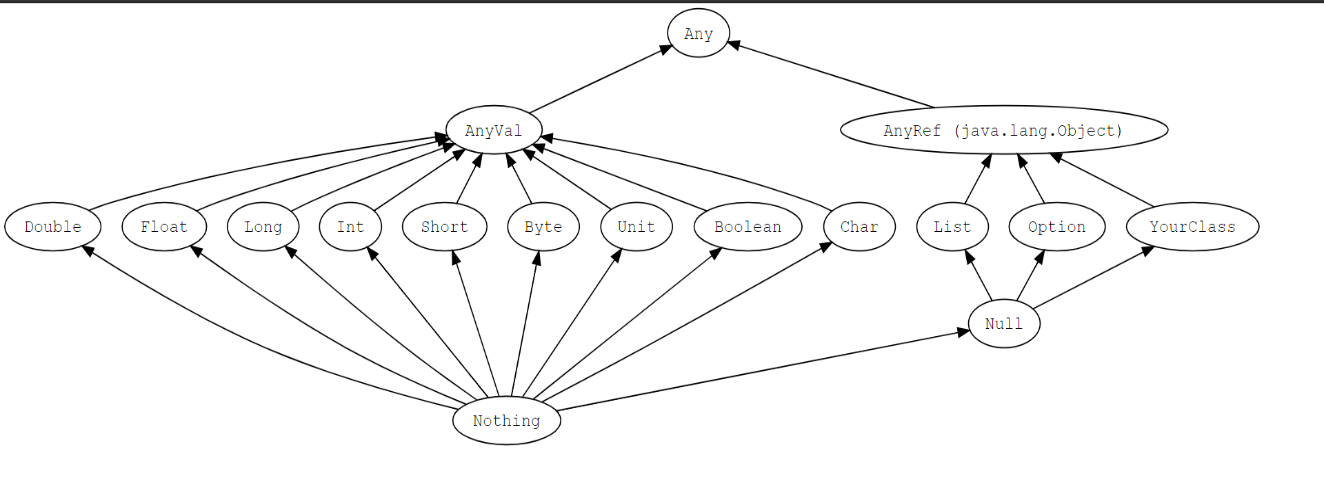
SumList (x:xs) = x + sumList xs

SumList [] = 0

In Scala:

* While Scala is a blend of object-oriented and functional programming concepts and a statically typed language, so that is making Scala’s functional programming construct make it easy to build interesting things quickly from simple parts, and makes it easy to structure larger systems and to adapt the new demands.
* Scala datatypes are the same with Haskell like:

Nothing, Double, Float, Long, Int, Short, Byte, Unit, Boolean, Char, Null, Lists, Options, Classes, Any



[Scala-lang.org](https://docs.scala-lang.org/tour/unified-types.html)

* Variables are made in Scala into that way:

scala> val sayhi = “Hello World!!”

val sayhi:String = “Hello World!!”

scala> val PI:Double = 22/7.0

val PI: Double = 3.142857142857143

Note: the number 7 is typed 7.0 to cast it into Double datatype and all datatypes in both languages are capitalized.

* And to define functions in Scala we will type, in example to find maximum number between two numbers:

scala> def max(x: Int, y: Int): Int = {

| if (x > y) x

| else y }

def max(x: Int, y: Int): Int

scala> max(5, 7)

val res0: Int = 7

* Pattern matching is like that one in Haskell, but we will show that by examples to see what differs from one to another and we will use the same examples in Haskell section.

*// To find sum of using pattern matching*

scala> val vectors = List(((1,2), (3,4)))

val obj: List[((Int, Int), (Int, Int))] = List(((1,2),(3,4)))

scala> vectors.map { case ((a, b),(c, d)) => (a+c, b+d)}

val res1: List[(Int, Int)] = List((4,6))

*// To find sum of list using pattern matching*

scala> def sumList(xs: List[Int]): Int = {

| if (xs.isEmpty) 0

| else xs.head + sumList(xs.tail) }

def sumList(xs: List[Int]): Int