Advanced Software Paradigms

Assignment 1. Ilyas Karimov

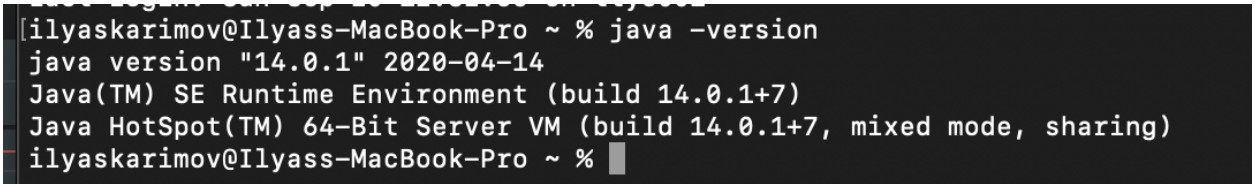
Task 4.

Go:

To install Go, I used this instruction/Link:
https://golang.org/doc/install?download=go1.15.2.darwin-amd64.pkg

Scala:

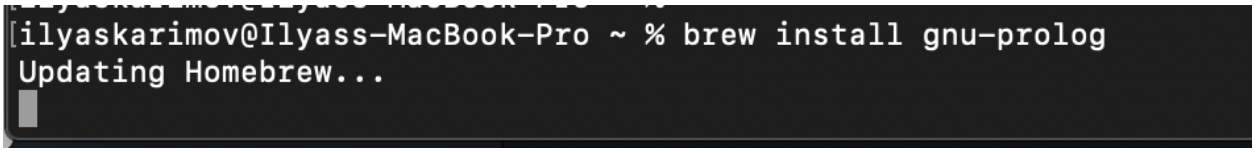
To install Scala, I used this instruction:
https://www.scala-lang.org/download/
To use Scala, I made sure I have minimum version 1.8 or 11 installed. I downloaded



I installed scala compiler by running **brew install sbt.**

SWI-Prolog:

To install Prolog, I used this instruction:
https://www.swi-prolog.org/download/stable
I installed prolog by running **brew install gnu-prolog**.



Compiling/Source Codes:

Go:

I used the Command-Line tool to compile the code, but I read that I could have used Microsoft Visual Code as an IDE for Go. I installed version go1.15.2 darwin/amd64. File extension is **.go**

Reference: https://gobyexample.com/hello-world

```
ilyaskarimov@Ilyass-MacBook-Pro hw1 % echo    > hello-world.go
ilyaskarimov@Ilyass-MacBook-Pro hw1 % vim hello-world.go
ilyaskarimov@Ilyass-MacBook-Pro hw1 % go run hello-world.go
hello world
ilyaskarimov@Ilyass-MacBook-Pro hw1 %
```

Source Code in hello-world.go file:

```
import "fmt"

func main() {
    fmt.Println("hello world")
}
```

Explanation:

1. Fmt comes from the word format, since Go was inspired a lot from C. fmt performs I/O functions, specifically, scanf() and printf() in C.
2. Func stands for function and it is accepted as a type in Go like in JavaScript.
3. But what I realized is semi-colons (;) are not used in Go language to end the statements.

Scala:

I downloaded IntelliJ for Scala, but then I had some problems, consequently, I used command-line. Installed version is 2.13.3. File extension **.scala**

```
ilyaskarimov@Ilyass-MacBook-Pro hw1 % vim hello-world.scala
ilyaskarimov@Ilyass-MacBook-Pro hw1 %
```

Source Code in hello-world.scala file:

```
object ScalaHelloWorld {
     def main(args: Array[String]): Unit = {
          println("Hello World")
     }
}
```

Explanation:

1. I learnt that **object** is a named instance with members such as fields and methods. Object and class have the same name.
2. It looks like Java. Def here means a function; we pass the argument. As far as I understood, Unit is a type like Void, and here, no value is to be returned.
3. Another difference is no semi-colons are used for end statements.
4. Rules for curly braces are the same {}.

```
==> Summary
🍺  /usr/local/Cellar/scala/2.13.3: 41 files, 22.8MB, built in 4 seconds
ilyaskarimov@Ilyass-MacBook-Pro hw1 % scalac hello-world.scala
hello-world.scala:4: error: '}' expected but eof found.
        }
         ^
1 error
ilyaskarimov@Ilyass-MacBook-Pro hw1 % vim hello-world.scala
ilyaskarimov@Ilyass-MacBook-Pro hw1 % scalac hello-world.scala
ilyaskarimov@Ilyass-MacBook-Pro hw1 % ls
IlyasKarimov_hw1_part2.docx     hello-world.go                  ~$yasKarimov_hw1_part2.docx
ScalaHelloWorld$.class          hello-world.pl
ScalaHelloWorld.class           hello-world.scala
ilyaskarimov@Ilyass-MacBook-Pro hw1 % scala ScalaHelloWorld
Hello World
ilyaskarimov@Ilyass-MacBook-Pro hw1 %
```

Prolog:

Reference: https://www.youtube.com/watch?v=SykxWpFwMGs

I used Command-Line for prolog as well. Normally, we store variables and operate on them, but in Prolog, I learnt that we store relationships in prolog. File extension is **.pl**

Source Code in hello-world.pl file:

```
write('Hello World'), nl.
```

Explanation:

1. Write means printing and nl means new line.
2. The statement ends with the period sign (.)

I also learnt that in order to stop the process, I need to use the command **halt.**

```
ilyaskarimov@Ilyass-MacBook-Pro hw1 % gprolog
GNU Prolog 1.4.5 (64 bits)
Compiled Aug 20 2018, 15:27:00 with clang
By Daniel Diaz
Copyright (C) 1999-2018 Daniel Diaz
| ?-
```

```
ilyaskarimov@Ilyass-MacBook-Pro hw1 % gprolog
GNU Prolog 1.4.5 (64 bits)
Compiled Aug 20 2018, 15:27:00 with clang
By Daniel Diaz
Copyright (C) 1999-2018 Daniel Diaz
| ?- write('Hello World'), nl.
Hello World

yes
| ?-
```

P.S. I wrote the code in hello-world.pl file and it showed that I can run the file with the consult command, yet it gave me errors. The same code worked like a charm in gprolog. So, I submitted that, the right one is above, the problem that I had is below:

```
| ?- consult('hello-world.pl').
compiling /Users/ilyaskarimov/Desktop/ADA-GWU/Software Paradigms/hw1/hello-world.pl for byte code...
/Users/ilyaskarimov/Desktop/ADA-GWU/Software Paradigms/hw1/hello-world.pl compiled, 1 lines read - 340 bytes written, 4 ms
error: /Users/ilyaskarimov/Desktop/ADA-GWU/Software Paradigms/hw1/hello-world.pl:1: native code procedure write/1 cannot be rede
fined (ignored)
```