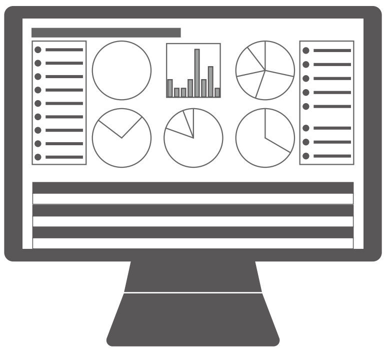


COLLEGE OF COMPUTING TECHNOLOGY - DUBLIN
BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY

MULTI-PARADIGM PROGRAMMING AND SCRIPTING

Assignment No. 2

Adelo Vieira
Student Number: 2017279
Lecturer: Dr. Muhammad Iqbal
December 22, 2019

Contents

1 Question 1 . . . . . . **1**

1.1 Java Integer data type 2

1.2 Java String data type 2

2 Question 2 . . . . . . **3**

2.1 Advantages and disadvantage of this approach 3

3 Question 3 . . . . . . **4**

4 Question 4 . . . . . . **5**

5 Question 5 . . . . . . **6**

Bibliography **8**

List of Figures

1.1 Numbers in Decimal and Binary systems 1

3.1 Java code snippet 1. Static languages 4

3.2 Python code snippet 1. Dynamic languages 4

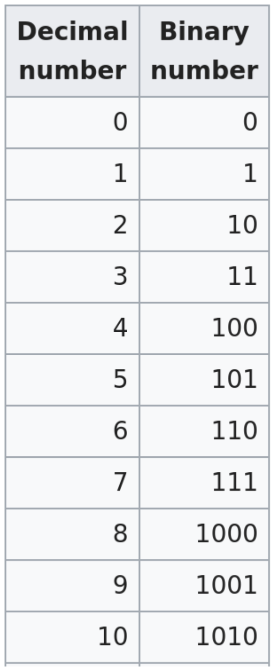
4.1 Lisp-tree 6

1 Question 1

The languages Scheme, C++, Java, and Python have an integer data type and a string data type. Explain how values of these types are abstractions of more complex data elements, using at least one of these languages as an example.

In computers, everything consists of binary data. So, in order to make a computer to understand what a number is (an integer, for instance) we first need to have a binary representation of the integer.

In Figure 1.1 we show the representation of numbers in Decimal and Binary systems. If you are a human, you will easily understand what a 5 means. Computers, on the contrary, would understand its binary representation: 1 0 1.



Decimal number	Binary number
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010

Figure 1.1: Numbers in Decimal and Binary systems [wikipedia.org]

So, when you write a 5 in your code, the programming language is acting as a layer of abstraction of the binary representation that the computer would understand (1 0 1).

When it comes to negative integer, the structure is a little more complexe. We usually represent negative numbers in decimal system adding the prefix $-$. This method is know as "signed number representations".

In computer, we cannot use any extra symbol to sign a number. Everything has to be represented as a sequence of bits. There are several methods to represent a negative number in binary system. One of the most used in programming languages is the Two's complement.

To obtain the Two's complement of a binary number, you have to inverse the binary number and add one.

Let's explain the Two's complement with an example:

- Decimal: 5
- Binary (using 16 bits): 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
- Inversion (one complement): 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0
- Two's complement: 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1

So, a negative integer (-5) would be represented as 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 on 16 bits using the two's complement method. The most important reason of using this method is that arithmetic operations $(+, -, *, /)$ are consistent.

1.1 Java Integer data type

In Java, Integer is a Primitive Types.

In java, according to its official documentation:

- int: By default, the int data type is a 32-bit signed two's complement integer, which has a minimum value of -2^{31} and a maximum value of $2^{31} - 1$. [oracle.com]

1.2 Java String data type

String, is not a primitive type in Java. A *String* is an class (*java.lang.Stringclass*) defined by a sequence of *character* objects; *character*, in turn, is defined from the *char* primitive type. [oracle.com]

In this sense, **a *String* type in java is an abstraction of a sequence of *char***.

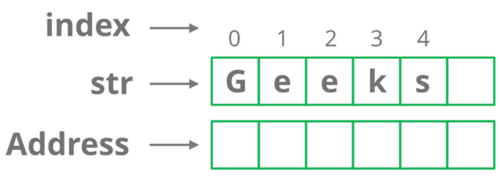


Figure 1.2: [geeksforgeeks.org]

char: "The *char* data type is a single 16-bit Unicode character. It has a minimum value of $u0000$ (or 0) and a maximum value of $uffff$ (or 65,535 inclusive)." [oracle.com]

2 Question 2

Java and Python programs are translated to byte code that runs on a virtual machine. Discuss the advantages and disadvantages of this implementation strategy, as opposed to that of C++, whose programs translate to machine code.

Traditionally, the compiler of a high level programming language, such as *Fortan* or *C*, translate the high level code into machine languages, which is used for the computer through the Operating System. This way, when you compile a code using C, you obtain an executable program that can be understood by the platform in which the code was compiled. Therefore, if you compile a *C* program in a particular platform, the executable file won't work in another platform.

However, *Java* implement a different method of compiling the code. When you compile using Java, the code is not translated into the Machine Language that can be directly understood by the Operating System. Instead, it is translated into an intermediate code (usually called, the *byte code*) that is going to be executed by a component called the *Java Vitual Machine* (in case of Java), which is in charge of compiling the *ByteCode* into the native machine language of the platform.

This way, a code compiled with *Java* can be executed in any platform where the Java virtual Machine is installed.

In the case of Python, it is usually described as an Interpreted language, but this is only parcially correct. Python actually translated the code into *bytecode* (*.pyc*), which is usually into a directory called ___pycache___. Then, the Python Virtual Machine will execute the *bytecodd*. [Bennett (2018)]

2.1 Advantages and disadvantage of this approach

The most important advantage of the approach, is that a program created using *Java* works in any platform using any operating system that runs a *JVM*.

The most important disadvantage of this scheme is that this intermediate layer represented by the *VM* has to translate the byte code into the machine language of the platform and of course this computing operation would have an impact on the speed of the program execution. Therefore, the execution of a byte code is slower compare with a native code execution.

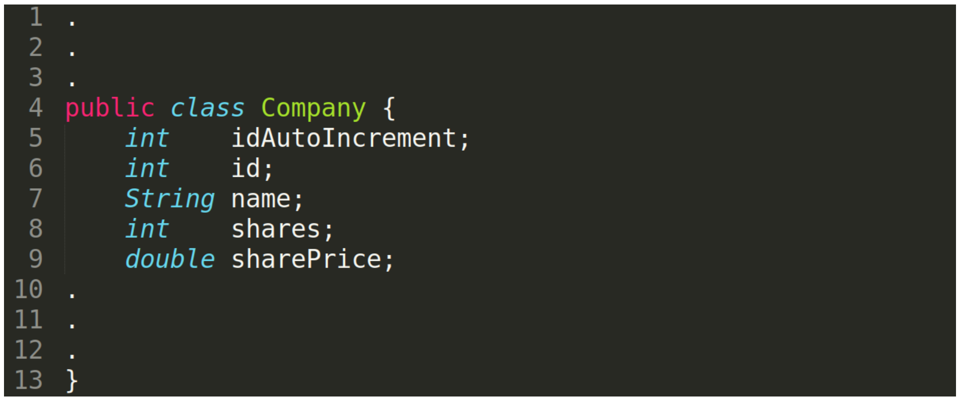
3 Question 3

Should a language require the declaration of variables? Languages such as Lisp and Python allow variable names to be used without declarations, while C, Java, and Ada require all variables to be declared. Discuss the requirement that variables should be declared from the point of view of readability, writability, efficiency, and security.

Before starting mentioning advantages or disadvantage of type declaration, we think that is important to define the two kind of Languages in the sense of the context of the question: Dynamci vs. static Langages:

Static languages (Ex. *C*, *Java*): We say that a language is static when the variable types have to be defined at compile time. So, when we write the code, we need to declare the type of the variables we are defining. This way, a variable of type *String* can only hold data of type *String*.

Java example:

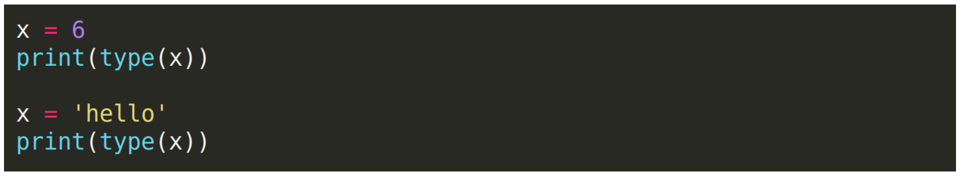


```java
.
.
.
public class Company {
    int    idAutoIncrement;
    int    id;
    String name;
    int    shares;
    double sharePrice;
.
.
.
}
```

Figure 3.1: Java code snippet 1. Static languages

Dynamic languages (Ex. *Python*, *JavaScript*): In the other hand, we say that a language is dynamic, when you don't need to define the variable type when you write the code. Therefore, the variable type is determined at runtime.

Python example:



```python
x = 6
print(type(x))

x = 'hello'
print(type(x))
```

Figure 3.2: Phyton code snippet 1. Dynamic languages.

Now, which one is better, should a language require the declaration of variable?. Let's point out some key differences between these two kind of languages:

- **Efficiency:** The most important difference is that in Static languages, the compiler is able to detect data type error before the code is executed. For example, if by mistake, you make an addition where a variable of type String is involved, the compiler of a static language would warn you about this error. So you cannot run the code before the error is corrected. That is not the case of a dynamic languages, where you will not be notified of this error until the code is executed. [thecodeboss.dev (2015)]
- **Writability:** A practical difference is that dynamic languages can be easier to learn and the code easier to write compare to static languages. This because you don't need to worry about data types or spend time learning strict differences between the data types when you use a dynamic programming language. [Papiewski (2016)]. However, when it comes to a large programs, the fact of not having data declaration can be negative aspect that can affect **Writability**. This could be solved in static languages by a properly use of comments.
- **Readability:** In a static language, event if you have to spend more time declaring variables, the resulting code is more readable and easier to understand. Thus, we can say that for code maintenance purposes, a static language is more suitable.
- **Refactoring:** When it comes to code maintenance, refactoring could be an important tool. Refactoring works much better in Static typed languages.

4 Question 4

Sheme's basic data structure is actually a little more general than the lists. Indeed, since the car of a list can be a list or an atom, it is also possible for the cdr to be an atom as well as a list. In the case when the cdr is not a list, the resulting structure is called a pair or S-expression instead of a list and is written (a . b), where a is the car and b is the cdr. Thus, (cons 2 3) = (2 . 3), and (cdr (2 . 3)) = 3. The list (2 3 4) can then be written in pair notation as (2 . (3 . (4 . ()))) Discuss any advantages you can think of to this more general data structure.

S-expressions are usually use as notation for tree-structured data. The advantage of the s-expression notation is that it gives a graphical representation of the data structure.

For example, the list (A B C) can be written in s-expression notation as (A . (B . (C . ()))), which give a more better graphical representation of the Lisp-tree in Figure 4.1.

```
        .
       / \
      A   .
         / \
        B   .
           / \
          C   empty
```

Figure 4.1: Lisp-tree corresponding to the (A . (B . (C . ()))) s-expression [wiki.c2.com (2013)]

Scheme uses mostly Lists. S-expressions that are list can be represented with the simplified notation. For example:

(a . ()) is represented by (a)

(b . (a . ())) can be written as (b a)

(c . (b . (a . ()))) can be written as (c b a) [uiowa.edu]

However, the s-expression can be use to represent a more wider range of data structure and gives a graphical representation of the structure, which is an advantage over other notations.

5 Question 5

Write a brief report that compares functional programming versus imperative programming (procedural programming). Highlight what is similar in both approaches and what is different in each approach. Use any two languages to highlight the difference between them.

A function is a mathematical expression that **RETURNS** a value. In **functional programming** every expression is a function, so everything has to return a value. We have to clarify that when we write (in Java for example) a function that return VOID; this is not functional programming. [Simpson (2014)]

Let's see a code example that will help us to clarify better what is functional programming. Consider the following code snippet to calculate the area of a window: [Simpson (2014)]

```
length = lengthOf(window);
height = heightOf(window);
area = length * height;
```

The above example is how we would do it in Java for example. This is NOT functional programming, because we are storing the value returned by the functions in variables that get stored in memory. Store something in memory is a side effect that must be avoid in functional programming. [Simpson (2014)]

The following code snippet is how we would do it in functional programming: [Simpson (2014)]

```
( (lengthOf(window)) * (heightOf(window)) )
```

As explained by Louden and Lambert (2011): "in imperative programming languages, variables refer to memory location that store values. Assignment allows these memory locations to be reset with new values". On the other hand, in Functional programming, variables are not related to memory location. Therefore, in pure Functional programming there are not assignments and so the value of a variable can not be changed.

Another important feature of Functional programming is that it doesn't support some Control flow statements like Loops or If-Else, because these controls require Assignments so the memory location can be reset with new values. Instead, they use recursion and conditional expressions (function calls). [tutorialspoint.com (2016)]

In Table 1 we resume some of the most important defferences between Funtional and Imperative programming.

Imperative programming	**Functional programming**
Mutable data. [tutorialspoint.com (2016)]. The data or variables can not be reset to new values.	Inmutable data. [tutorialspoint.com (2016)]
Not efficient for parallel programming. Can also be used in paralle programming but is not efficient compare to Functional programming. [tutorialspoint.com (2016)]	Efficient parallel programming. Functional programming don't support state. Thus, because there are no state-change issues, it is easier to program functions to work parallely. [tutorialspoint.com (2016)]
Use loops to iterate. [tutorialspoint.com (2016)]	Use reursion to iterate. [tutorialspoint.com (2016)]
Imperative programming style: Focused on describe **HOW** the computer should accomplish the task. [Petricek and Skeet (2007)]	Declarative programming style: Focused on describe **WHAT** the computer should accomplish. [Petricek and Skeet (2007)]

Table 1: Imperative vs. Functional programming

Bibliography

James Bennett. *An introduction to Python bytecode.* opensource.com, 2018. URL https://opensource.com/article/18/4/introduction-python-bytecode. 3

geeksforgeeks.org. *Strings in C.* URL https://www.geeksforgeeks.org/strings-in-c-2/. 2

Kenneth Louden and Kenneth Lambert. *Programming Languages - Principles and Practice.* Third edition, 2011. 7

oracle.com. *Class String.* URL https://docs.oracle.com/javase/10/docs/api/java/lang/String.html. 2

oracle.com. *Primitive Data Types.* . URL https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html. 2

John Papiewski. *Advantages & Disadvantages of Typeless Programming Languages.* techwalla.com, 2016. URL https://www.techwalla.com/articles/advantages-disadvantages-of-typeless-programming-languages. 5

Tomas Petricek and Jon Skeet. *Functional Programming for the real world.* , 2007. 7

Richard Simpson. *Imperative vs Functional.* , 2014. URL https://www.youtube.com/watch?v=n40Nz2YwRls. 6, 7

study.com. *Java: String Data Type.* URL https://study.com/academy/lesson/java-string-data-type.html.

thecodeboss.dev. *Programming Concepts: Static vs. Dynamic Type Checking.* , 2015. URL https://thecodeboss.dev/2015/11/programming-concepts-static-vs-dynamic-type-checking/. 5

tutorialspoint.com. *Functional Programming.* , 2016. URL https://www.tutorialspoint.com/functional_programming/functional_programming_tutorial.pdf. 7

uiowa.edu. *FUNCTIONAL PROGRAMMINGWITH SCHEME.* . URL http://homepage.divms.uiowa.edu/~slonnegr/plf/Book/AppendixB.pdf. 6

wiki.c2.com. *Ess Expressions.* , 2013. URL https://wiki.c2.com/?EssExpressions. 6

wikipedia.org. *Binary number.* URL https://en.wikipedia.org/wiki/Binary_number. 1