Programming with Scala: Language Exploration

Bhim Upadhyaya

Contents

List of Figures

List of Tables

Chapter 1

Introduction to Computing

The Oxford English Dictionary (OED) defines computing as "the use of operation of computers"; similarly, computation is defined as "the action of mathematical calculation." In daily life, we often find these words being used interchangeably even though scientific community makes distinction. Let's first analyze computation as it came first in the human civilization, formally with the invention of numbers. But it is quite self-evident that humans performed computation before inventing numbers as there should be a though process before finding suitable symbols for that thought process. This kind of thought process is likely to be available in other *Mammalias* as well as in some other *Classes*, categorized using traditional biological taxonomy.

Let's take two examples to illustrate computation: $1+1=2$ and $13+29=42$. Now, let's ask ourselves these questions: *What percentage of world population can perform first addition? What percentage of the world population can perform second addition without using a calculating machine What percentage of world population can perform second addition using a calculating machine?* We should not be surprised if the answer to our first question is not 100%. The United Nations' data show that answers to our second and third questions are not 100% [UNL13].

Analyzing further in the same direction, there are many more questions to be asked including: *How long it took us to recognize real world objects? How long it took us to take instructions (both in the form of signs and spoken language) from elders and perform the addition task for the first time in our lives? How long it took us to recognize written alphabets and numerals? How long it took us to perform written addition? How long it took human kind to be in this state of mind, which allows one to instruct and another to follow instructions and perform actions?* These questions might look a bit overwhelming and unnecessary at first, but these and many other similar questions govern our learning life cycles.

Now, let's take slightly different example to set stage for our Scala lessons.

SN	Hiearchy	Human	Dog	Domestic Pigeon	Cat
1	*Kingdom*	Animalia	Animalia	Animalia	Animalia
2	*Phylum*	Chordata	Chordata	Chordata	Chordata
3	*Class*	Mammalia	Mammalia	Aves	Mammalia
4	*Order*	Primates	Carnivora	Columbiformes	Carnivora
5	*Family*	Hominidae	Canidae	Columbidae	Felidae
6	*Genus*	Homo	Canis	Columba	Felis
7	*Species*	H. Sapiens	C. lupus	C. livia	F. catus

Table 1.1: Sample Biological Taxonomy Data

This too might look counter intuitive initially but we will write a Scala program for this later in this chapter. Table 1 shows sample biological categorization of human, dog, domestic pigeon, and cat. Here are some of the questions: *Is it a computational problem? Do we have sufficient information to decide whether it is a computational problem?*

Let's say, we are asked to build a dictionary or an information base that can be referred to get information. Now, it is fairly convenient to decide whether it is a computational problem if we have computing background. But this might be still confusing if we do not have any idea about computing, because computation cannot be seen on the surface. Even Google search may not look like a computational problem on the surface as we can't see regular calculations. In fact, Google search is a complex computation.

Assuming we don't have any knowledge of computing as defined by OED. Probably it is fair to say that all the human beings search at least one item in their life. When we are searching something, our mind performs computation. We might need to locate, count, or categorize items. Locating something might involve counting. For example, if we have to locate a book in another room, then we have to cross at least one door, assuming these are regular rooms in a regular house. Since we have enormous practice going from one room to another room in our lives, we might be performing the computation even without realizing it. Now, let's think about what it takes to train an infant as he/she grows to perform the same task. Does the infant need to learn how to count in order to perform this task? Probably the answer is yes. And it might take years to train the infant. Learning computing is not much different from this infant's training. The major difference is age. And of course, infants too can start learning computing these days.

We know how hard it is to live our lives without using any tool. Even in stone age, our ancestors used some sort of tools: a stone, a stick, or a little more sophisticate tool. Now, all of us, we know why we need tools. Also we know that it is not the same tool that can be utilized to solve every problem in our lives. This is true in the case of computational tools as well. Since this book deals with a particular programming language, Scala, in details, let's be concrete and say that this is true for programming languages as well. Programming languages are parts of computational tools.

Now we have some ideas about computation. Let's ask another question: *can every computational problem be computed?*. Well, there are several university level courses dedicated to answer this question. For now, we focus on our two problems—addition and tiny information base for biological taxonomy. The first one can certainly be computed. We limited the scope of second and made it viable for computing. Please note that we did not go for genomics, which requires enormous computing power.

In the following section, we discuss the basics of computing tools, called computers.

1.1 Introduction to Computers

Computers are the tools that we can use to perform some computations and come in various shape and size. There are hundreds of companies around the world that manufacture varieties of computers and computer parts. Generally a computationally useful computer has two categories of components—hardware and software. Hardware consumes energy and performs computations whereas software contains the logic for operations. It is the software that instructs the hardware in order to achieve a computational goal. A computational goal can can be as primitive as inverting a digit, converting 0 to 1 and vice-versa. In this book, we will learn a programming language that helps us to instruct computers in order to achieve one or more computational goals. Clearly it is a software component that helps us to create other software components.

1.1.1 Basic Components

Digital computers have the following basic components:

- Memory Unit
- Processing Unit
- Storage Unit

- Input Device
- Output Device

Almost every digital computing machine has some sort of memory. For example, if we are performing addition mentioned earlier, $1 + 1 = 2$, using a digital calculator, it remembers at least three different items: digit 1 (first operand), operation + (operator), and digit 1 (second operand). A typical laptop, say a laptop from *One Laptop per Child*, has much more memory than a typical calculator [OLC17]. The reason is that a laptop has to hold much more information and has to perform much more sophisticated operations than a typical calculator. And things might be different if we are referring to a scientific calculator. For now, we stick with a simple calculator that performs addition, subtraction, multiplication, and division.

In case of our calculator, we need a unit that performs the addition operation. The unit that performs this kind of operations is called *Processing Unit*. In case of digital computers, every high level operation or computational goal like opening a file or googling a word is eventually represented with 1s and 0s, these are the only two signals a digital computer understands. Interestingly, this transformation is a complex process and is studied as a computer engineering degree in traditional universities. From this course's perspective, let's remember the fact that every program we write will be eventually processed by a processing unit. A common terminology used for such a processing unit is *Central Processing Unit* as there are other processing units in a typical computer. For example, a keyboard has a processor to process keyboard inputs.

1.1.2 Operation

(content here)

1.2 Operating Systems

(content here)

1.3 Programming Languages

(content here)

1.4 Introduction to Scala

(content here)

1.5 Program Attributes

(content here)

1.6 Conclusion

(content here)

1.7 Review Questions

(content here)

1.8 Problems

(content here)

1.9 Answers to Review Questions

(content here)

1.10 Solutions to Problems

(content here)

Chapter 2

Scala Fundamentals

(content here)

2.1 Literals

(content here)

2.2 Identifiers and Keywords

(content here)

2.3 Types

(content here)

2.4 Declarations and Definitions

(content here)

2.5 Expressions

(content here)

2.6 Conclusion

(content here)

2.7 Review Questions

(content here)

2.8 Problems

(content here)

2.9 Answers to Review Questions

(content here)

2.10 Solutions to Problems

(content here)

Chapter 3

Classes and Objects

(content here)

3.1 Class Members

(content here)

3.2 Class Definition

(content here)

3.3 Object Definitions

(content here)

3.4 Conclusion

(content here)

3.5 Review Questions

(content here)

3.6 Problems

(content here)

3.7 Answers to Review Questions

(content here)

3.8 Solutions to Problems

Chapter 4

Control Structures

(content here)

4.1 For Expressions

(content here)

4.2 While Loops

(content here)

4.3 If Expressions

(content here)

4.4 Exception Handling

(content here)

4.5 Conclusion

(content here)

4.6 Review Questions

(content here)

4.7 Problems

(content here)

4.8 Answers to Review Questions

(content here)

4.9 Solutions to Problems

Chapter 5

Operators

(content here)

5.1 Operators as Methods

(content here)

5.2 Arithmetic Operators

(content here)

5.3 Relational and Logical Operators

(content here)

5.4 Bitwise Operators

(content here)

5.5 Operator Precedence and Associativity

(content here)

5.6 Conclusion

(content here)

5.7 Review Questions

(content here)

5.8 Problems

(content here)

5.9 Answers to Review Questions

(content here)

5.10 Solutions to Problems

Chapter 6

Data Input and Output

(content here)

6.1 Single Character Input

(content here)

6.2 Single Character Output

(content here)

6.3 Reading From a File

(content here)

6.4 Writing to a File

(content here)

6.5 Navigating Directories

(content here)

6.6 Conclusion

(content here)

6.7 Review Questions

(content here)

6.8 Problems

(content here)

6.9 Answers to Review Questions

(content here)

6.10 Solutions to Problems

Chapter 7

Traits

(content here)

7.1 Traits as Interfaces

(content here)

7.2 Construction Order

(content here)

7.3 Trait Members

(content here)

7.4 Multiple Inheritance

(content here)

7.5 Traits with Implementations

(content here)

7.6 Conclusion

(content here)

7.7 Review Questions

(content here)

7.8 Problems

(content here)

7.9 Answers to Review Questions

(content here)

7.10 Solutions to Problems

Chapter 8

Functions

(content here)

8.1 Functions as Methods

(content here)

8.2 Anonymous Functions

(content here)

8.3 Functions as Values

(content here) sectionFunction Parameters (content here)

8.4 Higher-Order Functions

(content here)

8.5 Closures

(content here)

8.6 Currying

(content here)

8.7 Conclusion

(content here)

8.8 Review Questions

(content here)

8.9 Problems

(content here)

8.10 Answers to Review Questions

(content here)

8.11 Solutions to Problems

Chapter 9

Pattern Matching

(content here)

9.1 Case Classes

(content here)

9.2 Variable Patterns

(content here)

9.3 Typed Patterns

(content here)

9.4 Pattern Binders

(content here)

9.5 Literal Patterns

(content here)

9.6 Stable Identifier Patterns

(content here)

9.7 Constructor Patterns

(content here)

9.8 Tuple Patterns

(content here)

9.9 Extractor Patterns

(content here)

9.10 Sequence Patterns

(content here)

9.11 Infix Operation Patterns

(content here)

9.12 XML Patterns

(content here)

9.13 Regular Expression Patterns

(content here)

9.14 Irrefutable Patterns

(content here)

9.15 Type Patterns

(content here)

9.16 Conclusion

(content here)

9.17 Review Questions

(content here)

9.18 Problems

(content here)

9.19 Answers to Review Questions

(content here)

9.20 Solutions to Problems

Chapter 10

Inheritance and Composition

(content here)

10.1 Extending Classes

(content here)

10.2 Overriding Methods and Fields

(content here)

10.3 Abstract Classes

(content here)

10.4 Invoking Superclass Constructors

(content here)

10.5 Polymorphism and Dynamic Binding

(content here)

10.6 Composition

(content here)

10.7 Conclusion

(content here)

10.8 Review Questions

(content here)

10.9 Problems

(content here)

10.10 Answers to Review Questions

(content here)

10.11 Solutions to Problems

Chapter 11

List Processing

(content here)

11.1 List Construction

(contenthere)

11.2 Operations

(content here)

11.3 Patterns

(content here)

11.4 List Class

(content here)

11.5 List Ojbect

(content here)

11.6 Conclusion

(content here)

11.7 Review Questions

(content here)

11.8 Problems

(content here)

11.9 Answers to Review Questions

(content here)

11.10 Solutions to Problems

Chapter 12

The Scala Collections Framework

(content here)

12.1 Mutable versus Immutable Collections

(content here)

12.2 Sets

(content here)

12.3 Maps

(content here)

12.4 Sequences

(content here)

12.5 Tuples

(content here)

12.6 Conclusion

(content here)

12.7 Review Questions

(content here)

12.8 Problems

(content here)

12.9 Answers to Review Questions

(content here)

12.10 Solutions to Problems

Chapter 13

Actors

(content here)

13.1 The Components of Actors

(content here)

13.2 Creating Actors

(content here)

13.3 Sending and Receiving Messages

(content here)

13.4 Life Cycle

(content here)

13.5 Channels

(content here)

13.6 Linking

(content here)

13.7 Conclusion

(content here)

13.8 Review Questions

(content here)

13.9 Problems

(content here)

13.10 Answers to Review Questions

(content here)

13.11 Solutions to Problems

Chapter 14

XML Processing

(content here)

14.1 XML Literals

(content here)

14.2 Serialization and Deserializing

(content here)

14.3 Data Extraction

(content here)

14.4 Pattern Matching

(content here)

14.5 Loading and Saving

(content here)

14.6 Conclusion

(content here)

14.7 Review Questions

(content here)

14.8 Problems

(content here)

14.9 Answers to Review Questions

(content here)

14.10 Solutions to Problems

Chapter 15

Parsing

(content here)

15.1 Lexical Analysis and Parsing

(content here)

15.2 Running Parser

(cotent here)

15.3 Regular Expression Parser

(content here)

15.4 JSON Parser

(content here)

15.5 Error Handling

(content here)

15.6 Conclusion

(content here)

15.7 Review Questions

(content here)

15.8 Problems

(content here)

15.9 Answers to Review Questions

(content here)

15.10 Solutions to Problems

Chapter 16

GUI Programming

(content here)

16.1 Simple Application

(content here)

16.2 Events

(content here)

16.3 Panels

(content here)

16.4 Layouts

(content here)

16.5 Example Application

(content here)

16.6 Conclusion

(content here)

16.7 Review Questions

(content here)

16.8 Problems

(content here)

16.9 Answers to Review Questions

(content here)

16.10 Solutions to Problems

Chapter 17

Unit Testing

(content here)

17.1 Unit Testing in Scala

(content here)

17.2 ScalaTest

(content here)

17.3 ScalaCheck

(content here)

17.4 JUnit

(content here)

17.5 TestNG

(content here)

17.6 Tests as Specifications

(content here)

17.7 Conclusion

(content here)

17.8 Review Questions

(content here)

17.9 Problems

(content here)

17.10 Answers to Review Questions

(content here)

17.11 Solutions to Problems

Bibliography

[OLC17] One laptop per child. http://one.laptop.org/, 2017.

[UNL13] United nations adult literacy rate. http://data.un.org/Data.aspx?d=SOWC&f=inID