Programming I First Assignment
Grado en Ingeniería Informática

Luis Miguel Danielsson[1,2] and Clara Benac Earle[1]

[1] Universidad Politécnica de Madrid (UPM), Spain
[2] IMDEA Software Institute, Spain
{lm.danielsson}@alumnos.upm.es {cbenac}@fi.upm.es

1 Introduction

Encoding and decoding is a very common technique used in Computer Science. It is used to encode characters, text, images, audio and even video into binary formats (internal representations) that a computer can handle or the other way around, to encode binary data into other representations that a human can read. There are some character-text encodings such as ASCII [3]. Other formats such as JSON [1] or XML [7] are used to encode binary data into textual representation that a human can read. Moreover, they constitute a de facto standard of communicating data over systems that use different technologies and internal representations. These are widely used in Web Technologies. For images JPEG [6] [4] is widely used. Audio is widely encoded using MPEG-1(MP3) [2]. Video can be encoded with the F4V,FLV(Flash) [5]. Another type of 'encoding' is encryption [8]. In this case the encoding takes some clear data and a password producing the cypher-text. For decryption, the cypher-text and the password are needed to recover the clear data.

1.1 Assignment introduction

In this assignment we will focus in textual encoding-decoding. Each encoding algorithm has a pair of functions **code-decode**. The most straightforward coder is a function that takes a sequence of characters and converts it to another sequence of characters. This function takes as input a clearword *clrwrd*(you define what can be written as a clearword). Its output is the clearword codified according to your coder, we call this *ctext*. A decoder is also a function that takes a

sequence of characters, namely *ctext* and its purpose is to decode it producing the clearword *clrwrd*.

One of the most famous text encoding is the ASCII code [3]. In this encoding each character is associated with a decimal number. Note that in this case the coding function will take a sequence of characters as input and produce a sequence of numbers (int). And conversely, the decoding function will take a sequence of numbers and produce a sequence of characters. We propose to implement both a student-defined **code-decode** and an ASCII **codeASCII-decodeASCII** functions.

2 Design

In order to fulfill the requirements of this assignment we propose the following design. The functions **code** and **decode** will transform a sequence into another sequence and they will do so by invoking the functions **encodeChar** and **decodeChar**, respectively. The functions **codeASCII** and **decodeASCII** will transform a sequence into another sequence and they will do so by invoking the functions **encodeASCIIChar** and **decodeASCIIChar**, respectively. The functions that take a single character **encodeChar**, **decodeChar**, **encodeASCIIChar** and **decodeASCIIChar** must accept any character defined in the primitive char type.

2.1 Properties

The pair of functions **code-decode** and **codeASCII-decodeASCII** must comply with the following specification:

(P1) They must be inverse of the other. Thus, **decode**(**code**(*clrwrd*)) = *clrwrd* and **code**(**decode**(*ctext*)) = *ctext*.
(P2) They must change the input. Thus **code**(*clrwrd*) $\neq$ *clrwrd* and **decode**(*ctext*) $\neq$ *ctext*.

You must extend this specification with whatever design you consider for your **code-decode**. Also, you must implement the **codeASCII-decodeASCII** functions, with the actual mapping between characters and numbers defined in the ASCII table. These same properties apply for **codeASCII-decodeASCII**.

Example 1. [3] Remember that you **must** design your own coder
code$([c, l, a, r, a]) = [d, m, b, s, b]$ **decode**$([d, m, b, s, b]) = [c, l, a, r, a]$

Example 2. **code**$([c, l, a, r, a]) = [a, r, a, l, c]$ **decode**$([a, r, a, l, c]) = [c, l, a, r, a]$

Example 3. This is an example of the ASCII encoding:
codeASCII$([c, l, a, r, a]) = [99, 108, 97, 114, 97]$
decodeASCII$([99, 108, 97, 114, 97]) = [c, l, a, r, a]$

2.2 Method headers

The method headers for all the described functions are precisely the following:

- **encodeChar**$(c : char) : char$
- **decodeChar**$(c : char) : char$
- **code**$(clrwrd : []char) : []char$
- **decode**$(ctext : []char) : []char$
- **encodeASCIIChar**$(c : char) : int$
- **decodeASCIIChar**$(n : int) : char$
- **codeASCII**$(clrwrd : []char) : []int$
- **decodeASCII**$(ctext : []int) : []char$

3 Submission

The following guidelines are **mandatory** and failing to meet them will result in the submission not being accepted. If the submission is not accepted it will be marked with a 0.

(G1) Work in pairs.
(G2) The code is written in a single class **CoderDecoder.java** and the methods to implement are described in section 2.2.
(G3) Only one of the authors submits the file.
(G4) The code has to include the specification (PRE & POST) and some tests (≥ 2).
(G5) The code will be checked to detect **plagiarism**, if plagiarism is detected, the code will not be accepted.
(G6) The code will be submitted to DeliverIt (see section 3.3).

There will be 2 submissions described below:

[3] char quotes omitted for readability

3.1 First Submission

The first submission aims to make you design and implement the coding and decoding of a **single character** as a first step to progressively build the full functionality of this assignment. For the first submission you must implement the following methods and submit them in the class **CoderDecoder.java**:

- **encodeChar**($c : char$) : $char$
- **decodeChar**($c : char$) : $char$
- **encodeASCIIChar**($c : char$) : int
- **decodeASCIIChar**($n : int$) : $char$

This submission will account for 40% of the Project's mark. **Deadline: November 8th 2020**

3.2 Final Submission

The final submission aims to build the full functionality described in this document. It is a students' decision whether to build upon the methods implemented in the first submission or to implement an unrelated algorithm. For instance, example 2 just reorders the characters, but does not invoke an **encodeChar** method. For the second and final submission you must implement the following methods and submit them in the class **CoderDecoder.java**:

- **encodeChar**($c : char$) : $char$
- **decodeChar**($c : char$) : $char$
- **code**($clrwrd : []char$) : $[]char$
- **decode**($ctext : []char$) : $[]char$
- **encodeASCIIChar**($c : char$) : int
- **decodeASCIIChar**($n : int$) : $char$
- **codeASCII**($clrwrd : []char$) : $[]int$
- **decodeASCII**($ctext : []int$) : $[]char$

This submission will account for 60% of the Project's mark. **Deadline: December 6th 2020**

3.3 DeliverIt

In this section we will describe the process of interaction with the submission system DeliverIt. Once the assignment is active, you will receive an email with the credentials you need to access DeliverIt. Then go to http://vps142.cesvima.upm.es/and introduce them. At that point, you will see the subjects in which you are registered. Then, you need to register in this practical assignment: CoderDecoder. Then, you will be prompted with the selection of your group partner(s). In this case, you can only select one partner. After this step, you will receive another email containing the information of:

- Subject
- Assignment
- Group members

[4] At this point you can upload your code and receive feedback from the correction system.

4 Evaluation

In this section we will describe what is the criteria affecting the mark that you will get in your assignment. Please note that all members of a group will receive the same mark.

4.1 Automatic testing

The code that you submit will be checked automatically using JUnit. You will see this testing technique in Programming II. As described in section 3 you must submit your code to DeliverIt. You may submit multiple times in order to receive feedback from the correction system and improve your code. Multiple submission will not affect your mark. The mark that you receive in this part is obtained from the number of test that you pass over the number of existing tests.

[4] not necessarily in that order

4.2 Code Review

Apart from the automatic testing, the code will be reviewed by the faculty. Adherence to the good practices-coding conventions (available in Course material / Course Notes) will be evaluated as well as creativity for the **code-decode** algorithm.

References

1. ECMA International 2017. The json data interchange syntax. In Standard ECMA-404, 2017.
2. Fraunhofer Institute for Integrated Circuits IIS. mp3.
3. Sue Gee. Ieee milestone for ascii. In I Programmer, June 2016.
4. G. Hudson, A. Lger, B. Niss, and I. Sebestyn. Jpeg at 25: Still going strong. IEEE MultiMedia, 24(2):96–103, 2017.
5. Adobe Systems Incorporated. Adobe flash video file format specification version 10.1.
6. Joint Photographic Experts Group (JPEG). Overview of jpeg.
7. World Wide Web Consortium (W3C). Extensible markup language (xml) 1.0 (fifth edition). 2008.
8. Wikipedia. Encryption.