# Programming Concepts in the Silver Code Guide for Elders

Oana-Sorina Lupşe*
Ciprian-Bogdan Chirila**
Horia Ciocârlie**
*University Politehnica of Timişoara, Romania
Department of Automation and Applied Informatics
E-mail: oana.lupse@aut.upt.ro
**University Politehnica of Timişoara, Romania
Department of Computers and Information Technology
E-mail: chirila@cs.upt.ro; horia@cs.upt.ro

***Abstract*—In the context of our digital society the elders are using less the new technologies. In order to encourage their interaction with the electronic devices they need a better understanding of the underlying mechanisms. To facilitate the understanding of the digital world we propose the idea of creating a dedicated community named Silver Code community. The community will be built with people from seven European countries around a set of didactic materials on programming published on web site.**

## I. Introduction

The modern society we live in depends more and more on technology in general and more specifically on digital technology. In this context elders have a disadvantaged position because they did not grow up with gadgets and programs. Involving them in the digital world has several advantages among others:

- they can benefit from the e-administration facilities to access different services;
- they can use the electronic financial systems like e-banking for paying bills;
- they can better use medical assistance devices;
- they can interact better with young people (e.g. nephews) discussing about the new technologies and trends;
- they can spend quality time in a pleasant community.

In this sense we developed the Silver Code project having an Erasmus+ financial funding and including partners from seven European countries: Bulgaria, Italy, Austria, Portugal, Slovenia, Romania, Poland. The project is about creating a community of elders over 55 years old that should use blended learning tools in order to learn programming for a better integration in the digital world. It is considered that knowing how hardware and software systems work will increase the confidence and will give better skills to elders in using computers and other electronic gadgets. One of the main goals of the project is the creation of a learning material dedicated to elders for learning computer programming. In this sense we created the Silver Code programming guide.

In Figure 1 are depicted the main steps an elder has to make in order to understand programming and to be able to write short programs. These represent the main structure of the Silver Code project training materials [6]. We iterate briefly the content of each chapter of the programming guide. The first section is about learning the basic digital skills for using the computer, writing emails, chatting online, posting in social networks etc. The second section presents basic web programming examples in HTML [5], CSS [4] and JavaScript [9]. The third section describes the most important computational thinking patterns to be used in designing algorithms. The fourth section presents the basic elements of computer programming in JavaScript. The fifth section tutors the elders in creating two complex web applications the former dedicated to flower watering and the latter to pill reminding. In the

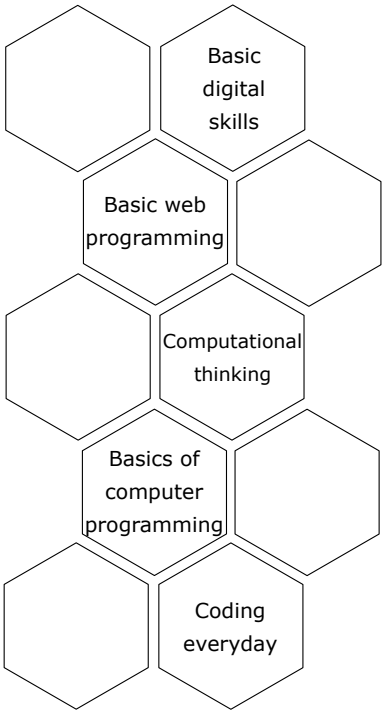


Fig. 1. Silver Code Programming Guide Structure

sixth section we have a glossary of terms used in the digital world. The paper is structured as follows. Section II presents related works in the field of computer programming guides and concepts. Section III presents briefly the Silver Code programming guide structure. Section IV presents one the final web applications the elders have to build by themselves. Section VI concludes and sets the future work.

## II. Related Works

In [1], [3], [2] is presented the auto-generative learning objects technology AGLO, which consists in reusable template models instantiated with random values having a specific learning objective. Their application is rich in the area of exact sciences: mathematics, computer science, medicine.

In [8] are presented ideas in the field of block chain cryptographic technology to be used in open education. Such approach is useful according to the Silver Code project proposal in the phase of issuing badges and certificates to the elders when they finish a unit or a module.

The state of the art contains dozens of learning programming materials (books, online tutorials, papers, videos, lectures in open academies) for different programming languages (Java, JavaScript, Python, Scratch etc.) and for different types of users: beginners (like hobbyists), intermediate (like students), experts (like IT companies developers, technicians), researchers (who develop new technologies). In [7] are researched some of the motivations, frustrations and design opportunities of older adults in learning computer programming.

## III. The Silver Code Programming Guide in a Nutshell

In this section we present briefly the content of our programming guide structured in modules and units.

### A. Basic Digital Skills

Module 1 is about presenting basic digital skills.

Unit 1.1 presents briefly the computer parts for elders not being familiarized at all with the digital technology: CPU, mother board, memory, HDD, modem, speakers, mouse, keyboard, video card, network card, printer, microphone, optical drive etc. Some details are given for memories like cache memory and virtual memory. Text file editors are explained in the context of UTF-8 and ASCII encoding together with basic text operations: find and replace, cut, copy, paste, undo, redo which work also in word processors. Finally, there are explained the basic steps in the creation of a PowerPoint presentation.

Unit 1.2 presents web browsers, email clients and social media tools like Skype and Facebook. In this unit we learn what an email message is, what are the basic operations with emails are and how to create an email account on Google website. Then, the sending of a new message is exercised paying attention to several details like: destination, carbon copy (CC), blind carbon copy (BCC), subject and message. Group mailing lists, attachments, automatic replies are also exercised.

Next, a few browsers are presented like:

- Internet Explorer which is an outdated browser but still available on older computers;
- Mozilla Firefox where the private browsing concept is explained, which is available in all other browsers;
- Google Chrome as a browser which enables the use of Google services on all your computers;
- Apple Safari that comes with the MacOS operating system.

The emphasize is put also on mobile web browsers that run on mobile phones and tablets:

- Android Browser;
- Chrome;
- Opera Mini;
- Internet Explorer;
- Safari.

The Skype tool is explained together with voice over IP, Internet Protocol (IP) and addresses concepts. The steps for installing Skype and setting up a call are exercised. Finally, there is presented the Facebook social network with its facilities.

Unit 1.3 presents algorithms, logical block diagrams and control structures. One section is dedicated to variables explained in the context of JavaScript programming language. Another section is about constants. Next section is about data types:

- `number`;
- `string`;
- `boolean`;
- `object` together with the special `null` value;
- `undefined` a special type with its own unique value.

Once all ingredients are defined the next section presents operators:

- additive +, −;
- multiplicative ⋆, /, ...;
- incremental ++, −−, ...;
- comparison and relational ==, !=, <=, ...;
- assignment =, +=, ...;
- logical &&, ||, !.

Another section is dedicated to simple algorithms or reading two values and performing one formula written in natural language, pseudocode, logical block diagrams. A special section is dedicated to the block diagrams where the basic blocks are presented:

- start block;
- stop block;
- input block;
- output block;
- assignment block;
- decision block.

A very simple algorithm is then exemplified using JavaScript code. Basic control structures are explained:

- sequence;
- selection;

- looping.

Unit 1.4 presents array algorithms for uni-dimensional arrays or vectors. The introductory section motivates the need for the arrays with convincing examples and explains the indexing mechanism. Then, specific array algorithms are described:

- reading an array from the input;
- writing an array to the output;
- linear searching of an element;
- searching for the minimum / maximum element;
- computing the sum of all elements.

Unit 1.5 presents two-dimensional arrays or matrix algorithms. A blood pressure matrix is built on two dimensions hourly measurements and daily series in order to exemplify one of matrices uses. Reading and writing algorithms for matrix elements are explained using two nested loops.

Unit 1.6 presents subroutines or functions. The introductory section explains the concept. The routine definition section explains how to declare a function in the context of JavaScript.

Unit 1.7 presents basic string operations. In the first section a graphical explanation, hardware related representation is given in an example. String constructions are exemplified in the context of JavaScript. Small executable examples in HTML embedded JavaScript code are also given. Details about ASCII characters encoding are given in this context. Next, string methods are described and exemplified as follows:

- concatenation or joining two strings;
- `length` of a string;
- `indexOf` a string;
- case conversion `toLowerCase`, `toUpperCase`;
- `replace` for replacing a substring in a string;
- `slice` for slicing a string into multiple ones based on a separator;
- `trim` for eliminating leading and trailing white spaces;
- `substring` for extracting a substring from a string;
- `match` to check if a substring is part of a string.

### *B. Basic Web Programming*

Module 2 presents basic elements of web programming.

Unit 2.1 presents the steps for building a simple web page. The first section starts with a basic presentation for HTML:

- tags and attributes;
- head and body;
- images.

A valuable reference is set towards the `www.w3schools.com` web site where all these aspects are explained in details. Unfortunately, the accessibility of the material is restricted to English speakers only.

Unit 2.2 describes basic concepts of Cascading Style Sheets (CSS). A first section motivates the need for CSS. Next section explains how selectors work:

- type selectors (e.g. `h1{...}`)
- universal selectors (e.g. `*{...}`);
- class selectors (e.g. `.black{...}`);
- ID selectors (e.g. `#panel{...}`).

The next section is about measurement units:

- relative percentages (%);
- centimeters (cm);
- typographical em units (em);
- points (pt);
- pixels (px).

Colors can be expressed as:

- hex code (e.g. #FFAABB);
- short hex code (e.g. `#8A9`);
- RGB percentage (e.g. `rgb(50%, 50%, 50%)`);
- RGB absolute values (e.g. `rgb(0,0,255)`).

The next sections are about setting styles for:

- backgrounds using properties like `background-color`, `background-image`;
- texts using properties like `color` and alignments `text-align`;
- images using properties like border, width and height;
- tables using properties for border colors, padding and margins.

Unit 2.3 presents the creation of a simple web application in JavaScript having a more elaborated user interface than the previous examples. The first section presents the syntactical elements of the language, reminds the elders the variables, the operators, the if-else instruction, the functions and the browser related events. In this unit elders have all the elements to write a small application that displays a bulb and reacts to the clicks on that bulb by turning it on and off alternatively. For this they will use two images: the former of an "on" bulb and the latter of an "off" bulb that will be displayed interchangeably. In this unit some JavaScript elements are overlapping the ones from Unit 1.3.

### *C. Computational Thinking*

Module 3 is more abstract and teaches computational thinking concepts that are usually used in code writing by programmers.

Unit 3.1 presents decomposition, pattern recognition, abstraction and algorithm design. Decomposition is breaking a problem into parts. While breaking a problem into parts parallelization may be applied to solve each of the part problems. The opposite operation to decomposition is synthesis. Once the part problems were solved their solutions can be assembled back to form the solution of the initial problem. An example is given in this sense with the help of a bicycle. The elders must break down the bicycle into components, learn how each of the components function independently and then putting them together as a whole. Thus, they can understand how other complex machines work.

In the next section patterns are emphasized from different sources: nature, pictures, poems etc. Patterns are expected to appear in the process of decomposition. Patterns imply the repetition of operations so they imply efficient problem solving. In this context patterns and patterns recognition are very important elements of computational thinking.

Abstraction is another important dimension of computational thinking. A set of informal definitions are given followed by some examples:

- a picture of Pablo Picasso where three musicians are abstracted as geometric shapes and colors;
- the learning of a foreign language by sentences having the structure: subject, action and object;
- universal orientation symbols like: H letter for hospitals, airplane symbol for airports, anchor symbol for ports etc.

The algorithm design is the next section where algorithms definitions are given: step by step instructions, recipes, storyboards. The algorithm design is the transformation phase from modeling to operational. Next, code is explained as a computer language that are translated from human language for computers to understand.

Unit 3.2 contains only exercises for the previously four presented concepts.

### D. Basics of Computer Programming

Module 4 presents the basics of computer programming. Unit 4.1 is an introductory to programming languages in general. A definition of programming languages is given. In the next section there are given several code samples in languages like: C, C++, Java, C#, PHP, Ruby, Python, SQL, Visual Basic and LaTeX. Then there are presented samples written in each of these languages about how to write on the screen "I like Silver Code" text message. Finally, a table is compiled with the most used websites and their implementation languages. On the front-end most web sites use JavaScript, while on the back-end the choices are divers.

Unit 4.2 focuses on JavaScript object-oriented programming elements. First in order to define object-oriented programming the other programming paradigm is explained, namely procedural programming. The basic elements of object technology are explained: classes, objects, state, behavior. Then several concepts are presented like: encapsulation, aggregation, inheritance for ordering the abstractions, polymorphism explained using an animal sounds example.

The next section focuses on object properties (attributes) and methods. A runnable example in embedded in HTML is given. Again the language primitive types are iterated and the concept of object reference is explained. The built in types are reviewed and object creation is explained using the new operator and an example is given in the context of book objects having attributes like author and title.

Unit 4.3 presents how to install Notepad++ text editor and to create and edit a JavaScript project. These competences will be used intensively from now on.

Unit 4.4 presents some captivating gaming applications in JavaScript like: hello, telepathy, guessing game. Finally, a library application is developed around the previously defined book class.

### E. Coding Everyday

Module 5 takes programming at the creativity level where elders will implement and customize their own applications. Unit 5.1 explains the code of a flower watering web application. The application itself has its own modules in HTML, CSS and JS.

Unit 5.2 will be explained in detail in section IV where the design of a pill reminder application is presented.

Unit 5.3 presents some practical examples of how coding helps people in their everyday life in the context of four applications: Uber, Airbnb, food ordering and payments.

For Uber is presented the code sample from the front-end in order to prove to elders that its code is very similar to what they have previously learned.

For the Airbnb hospitality service marketplace some HTML code is presented having the same purpose.

For food ordering an example is given from the United Kingdom, namely the `www.hungryhouse.co.uk` website.

For electronic payments the selected example is the one from the International Banking of Poland with some code extracted from their website.

Unit 5.4 presents augmented reality examples:

- head-up displays present in airplanes and recent cars;
- augmented reality mobile phone applications;
- medical e-learning applications of the human body structuring.

### F. Glossary of Technical Terms

Module 6 is a glossary of technical terms heavily used in the digital universe.

## IV. The Pill Reminder Application

The Pill reminder application is a single web page application that records the pills that an elder has to take. In Figure

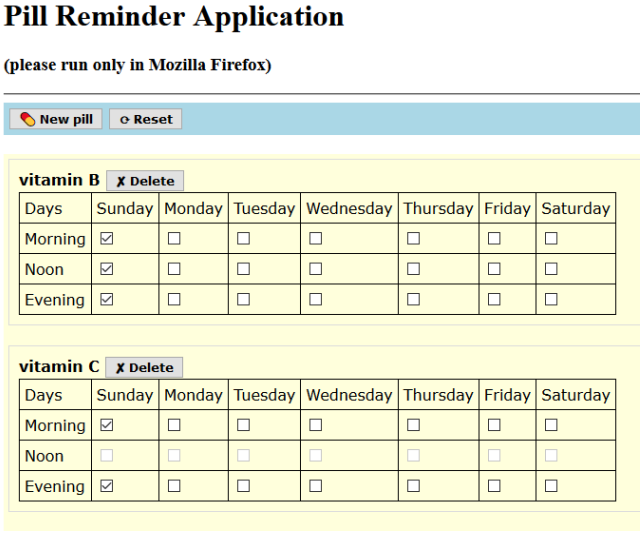


Fig. 2. The pill reminder application

2 we have an example of a use case for the Pill Reminder application. There are defined two medicines which happen to be vitamins. The application allows pills recordings for each day of the week and for the most important three moments of the day: morning, noon and evening. When defining a pill, the user can enable its consumption record on a certain day and on a certain moment of the day in a week as the physician wrote the prescription. The modules of the application are:

- `index.html` module contains the application HTML layout;
- `index.css` module contains the formatting tags for the HTML elements;
- `index.js` module contains the application entry point and the instantiation of the main classes;
- `pills.js` module contains the class definition for the pills collection;
- `pill.js` module contains the class definition for the pill;
- `week.js` module contains the class definition for the week schedule element;
- `day.js` module contains the class definition for the day schedule element.

The JavaScript modeled classes are:

- `Pills` class is the collection class managing all the pills in the application;
- `Pill` class models a medical pill the patient has to take;
- `Week` class models a collection of 7 days when the pills have to be taken;
- `Day` class models a day with its the three important moments for the pills to be taken.

Each class is set in a separate file module.

### A. The Pills class

The Pills class manages the collection of Pill objects. The persistent content is stored into a browser cookie. Next, we will iterate the methods elders will have to implement. The `init()` method performs the initialization tasks. If there is no already saved cookie a data initialization method is called, otherwise the data from the cookie is read for the pills list instantiation.

The `initData()` method creates a sample hierarchy of objects to be displayed to the users. The structure has three layers:

- the pill objects that contains week objects;
- the week object that contains 7 day objects;
- the day objects, that contain primitive values like:
  - day id;
  - three integers as booleans (0,1) for the obligation to take the pill in the three moments of the day (morning, noon and evening);
  - three integers as booleans (0,1) for the status of the taken pill for each of the moments of the day.

The `findById()` function is defined to be used when clicks are performed on the web page in different elements and their corresponding objects must be retrieved from the collection to perform operations on them.

The `load()` method reads the JSON representation of the cookie and creates the list of pills. The week object with 7 days objects are embedded in each pill.

The `store()` method calls JSON library functions to serialize the objects into JSON format and then to store them into the cookie.

The `create()` method enables the appearance of the new pill HTML panel where the user may add its details and hides the pill creation button.

The `doCreate()` method performs the followings:

- computes the new id for the pill;
- gets the data from the HTML controls;
- instantiates the days, week, pill objects;
- pushes the newly created pill into the collection;
- hides the user interface creation panel;
- calls the storing method of the collection.

The `doCancel()` method hides the pill panel and restores the pill creation button.

The `show()` method iterates all pill objects from the collection, generates all the HTML representations and adds these representations to a pills HTML div element.

The `delete(id)` method performs three operations:

- iterates the collection to find the pill;
- deletes it from the collection and removes the HTML representation from the user interface;
- calls the storing method of the collection because its content was changed.

The `take()` method marks one pill as taken in one moment of the day.

### B. The Pill class

The `Pill` constructor builds a pill object out of an integer id, name and week object references.

The `toHTML()` method generates an identifiable HTML div element containing an inner representation generated by the next method.

The `getInnerHTML()` method generates an HTML table with four rows (one for the day names and one for each important moment of the day) and seven columns (for each day of the week). The table cells contain radio buttons which are checked or not depending on the user which took or not the pill. Radio buttons are grayed (marked as inactive) if the pills are not mandatory in that moment of the day.

### C. The Week class

The `Week` constructor builds a week instance as an Array collection of day objects.

The `addDay(day)` method adds a day object to the collection.

The `getDay(id)` method retrieves a day object from the indexed collection.

### D. The Day class

The `Day` constructor creates a day object from seven parameters:

- id codes the identifier of the day from 0 to 6, 0 is Monday and 6 is Sunday;
- integer as boolean (0,1) `neededM` the pill is needed in the morning;
- integer as boolean (0,1) `neededN` the pill is needed at noon;

- integer as boolean (0,1) `neededE` the pill is needed in the evening;
- integer as boolean (0,1) `takenM` the pill was taken in the morning;
- integer as boolean (0,1) `takenN` the pill was taken at noon;
- integer as boolean (0,1) `takenE` the pill was taken in the evening.

The `getId()` method returns the identifier of the day (0-6).

The `getName()` method returns the literal name of the day (e.g. Monday).

The two projects implementation require only a text editor like Notepad++ and a web browser like Mozilla for debugging and running the applications.

## V. Discussion

In this section we analyze the concepts that we defined throughout the material and how are they used in the web applications to be designed by elders.

At the end of Unit 1 the elders should know how to use all the tools in order to interact with the online Silver Code community. One open point for improvement would be the presentation of an online forum and its philosophy (users, threads, posts) to facilitate its use in the context of the Silver Code project. In module 1 we explained all the basic programming concepts, notations and algorithms and they were compiled in a compact material with links to auxiliary bibliography to enable further documenting.

After attending the modules of Unit 2 the elders have the first hands on programming by implementing the bulb web application.

Module 3 somehow separates the concepts definition from their testing in two separate units. The solution would be to split the four concepts into two concepts per unit together with their exercises.

In module 4 some concepts are redundant in the sense that they are already presented in other modules. It is the case for the programming languages definition and JavaScript types. On the other hand the Notepad++ editor setup module should be placed earlier in the material probably a the beginning of Module 2.

Module 5 uses the jQuery JavaScript library to access the elements from the applications web pages. The library was not presented to elders in our materials but it uses object-oriented instances and methods for the accesses. Regarding persistence, presenting a relational database to elders would be too difficult so we used the JSON format for representation and the cookie mechanism from browsers for persistence storing. Cookies can store a limited size quantity of information, a few tens of kilobytes, so the textual representations has to be very compact. Booleans were represented as integers because ”true” and ”false” constants take up 4, respectively 5 bytes while 0 and 1 take only 1 byte in the JSON representation.

## VI. Conclusions and Future Work

We conclude that the Silver Code project team created a programming guide covering all concepts necessary to understand programming. Using the guide the elders will be able to write small size runnable programs. There are included all the necessary programming concepts at a decent level for the elders.

The next step of the project is to embed the learning materials in the Ilias [10] LMS platform on the `silvercodeproject.eu` website. As immediate future work we intend to pilot test with 30 elders in face to face meetings and then to get the feedback in order to improve the programming guide materials.


## Acknowledgments

We would like to thank the Silver Code team members for the ambition and effort put into this project.


## References


[1] Ciprian-Bogdan Chirila. Auto-generative learning objects for it disciplines. In *Proceedings of the International Conference on Virtual Learning 2015*, pages 1–6, Bucharest, Romania, October 2015. University of Bucharest, Romania.

[2] Ciprian-Bogdan Chirila. Auto-generative learning objects in online assessment of data structures disciplines. *BRAIN - Broad Research in Artificial Intelligence and Neuroscience*, 8(1):24–34, April 2017.

[3] Ciprian-Bogdan Chirila, Horia Ciocarlie, and Lacramioara Stoicu-Tivadar. Generative learning objects instantiated with random numbers based expressions. *BRAIN - Broad Research in Artificial Intelligence and Neuroscience*, 6(1-2):1–16, October 2015.

[4] Mozilla Corporation. Cascading style sheet (css). https://developer.mozilla.org/en-US/docs/Web/CSS, 2018.

[5] Mozilla Corporation. Web technology for developers, html, 2018.

[6] Valentina Georgieva, Maria-Elena Antunes, Elisa Chiesa, Karina Sirk, Erol Koc, Alexander Kobylarek, and Ciprian-Bogdan Chirila. Silvercode training material, 2018.

[7] Phillip J. Guo. Older adults computer programming: Motivations, frustrations, and design opportunities. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 7070–7083, Denver, Colorado, USA, May 06-11 2017.

[8] Carmen Holotescu. Openeduchain: Design and applications of a blockchain for open education. In *In Proceedings of the 14-th International Scientific Conference eLearning and Software for Education (ELSE)*, Bucharest, Romania, April 2018.

[9] ECMA International. Standard ecma-262 ecmascript 2016 language specification. http://www.ecma-international.org/publications/standards/Ecma-262.htm, 2016.

[10] Ilias Team. Ilias e-learning. http://www.ilias.de, 2018.