University of Zurich[UZH]

Department of Informatics

Software Construction (HS17)

Course Project

TAs: Raphael Matile, Livio Sgier
Head Assistant: Jürgen Cito
Instructor: Prof. Bertrand Meyer

This term you will carry out a project with the prime goal of putting to practice the main principles and techniques that you learn in the Software Construction course.

Project Description

HTML Generator

Many applications generate output, such as testing statistics or simulation results, which can be reported to the user in static HTML format. However, generating such static websites is often messy, especially when HTML is handled as strings in code. Hours can be spent on debugging wrong links, forgotten closing tags, and hard-to-read indentation.

The goal of this project is to relieve the programmer from this burden. A library of Eiffel classes will handle URL generation and linking, correct tag naming and indentation, writing well-formed HTML, and so on. The library's API should support creating a website with multiple linked pages with images, bullet lists, tables, and so on. It should also be possible to include existing HTML snippets in a document, for instance, an image map already stored in a file or in a string object.

The architecture of the library is, of course, left to your creative imagination and pragmatic instincts. But, make sure that it is *extensible for other output formats*. Some design patterns you might consider include the visitor, composite, builder, and decorator. As you will see during the course, different techniques and choices can impact the code's maintainability, flexibility to accommodate certain features and the ease with which others can use it.

Output to Markdown (optional):
Many systems (for instance, GitHub documents) are formatted in a simplified markup language called Markdown[1]. In addition to generating HTML documents, you could extend your library to also support generating documents in Markdown format.

Repeating from above, to signify the importance of good software architecture: Make sure to design your library in a way that is extensible for other output formats. Having a concrete other format in mind (such as Markdown) might help to design proper abstractions that allow for extensibility and maintainability.

[1] https://en.wikipedia.org/wiki/Markdown

Initial Setup

The project is carried out in **groups of four** students. You have to form the groups by Sunday, September 24th. Once you have a group, send one email per group to Jürgen Cito (cito@ifi.uzh.ch) with the names of the group members and their GitHub usernames (register on github.com if you have no account yet). Also specify if your group has a preference for one of the exercise session days (Tuesday or Wednesday). If you don't find a group, also send us an email with your name, GitHub username, and preference for the day of the exercise session, so we can put you together with other students.

We will use Git for version control. All submissions (documents and source code) will be delivered through your repository that will be created for you on GitHub. It will be a private repository within our GitHub organization (https://github.com/sealuzh) and will be available only for the respective team members under the name *sc-hs17-groupX* where *X* is your group number.

Grading and Project Structure

The project will be graded and contributes 50% of the final course grade. You need to have at least a 4.0 in both the project and the exam to pass the course!

To help you plan your workload during the semester, we have adopted a project structure with 5 milestones: Requirements Specification, API Design, Test Plan, Implementation, and Testing and Revision. Each phase has a strict deadline and deliverables that need to be handed in. Additionally, for every phase you will need to give a short presentation in one the indicated exercise sessions. The presentation should summarize what you have done in the past project phase. The purpose of the presentation is to receive feedback from the TAs as to how the presented solution should be adapted to serve as a better basis for future phases.

Milestones

1 Requirements Specification (20 points)

Duration: Monday, September 25th – Sunday, October 15th (3 weeks)
Deliverable: SRS document (10-20 pages)
Presentation: Exercise session October 17th/18th
Template: on course website
Submission: *sc-hs17-groupX*/documents/requirements.pdf [SRS document]

The ultimate purpose of requirements specification is to describe what qualities a system must exhibit and what goals it must reach. It does not describe how these qualities and goals are achieved. Your task is to develop an SRS document[2] for your project, which will serve to clarify the scope and features of your project and to prioritize the implementation of functionality. We will provide you with a template and example documents from previous years. When you write your SRS document, make sure your document adheres to the 15 quality goals discussed in the lecture.

The presentation (5-7 minutes, 4-6 slides) should give an overview of your SRS and explain the most important points. If there were any issues during the SRS writing phase that created discussion among the team members, they could make an interesting presentation topic.

[2] https://en.wikipedia.org/wiki/Software_requirements_specification

2 API Design (20 Points)

Duration: Monday, October 16th – Sunday, October 29th (2 weeks)
Deliverables: Design document, Eiffel classes for public API
Presentation: Exercise session October 31st / November 1st
Template: on course website
Submission: *sc-hs17-groupX*/documents/design.pdf [Design document]
sc-hs17-groupX/src [Public API classes]

Write a set of Eiffel classes that will be the public API of your library. The set of features has to satisfy the requirements described in the SRS. The Eiffel classes have to be equipped with class and feature comments as well as contracts. The routines don't have to be implemented yet. In the design document, add a class diagram of the library and describe which design patterns you have used. For each design patterns specify which classes are part of the pattern and why you used this particular pattern.

In the presentation (5 minutes, 4-6 slides) you should show an overview of your design. Explain important design decisions and describe which patterns you used.

3 Test Plan (15 points) - UPDATED

Duration: Monday, October 30th – Monday, December 4th
(5 weeks – in parallel with implementation)
Deliverables: Test plan
No Presentation
Submission: *sc-hs17-groupX*/documents/design.pdf [updated design document]
sc-hs17-groupX/tests/ [your test suite]

In this project step, you develop a **test plan** for **your** project. This means that you will take the role of an external test engineering team. Write tests for each **functional requirement** of the SRS. Annotate each test with the requirement that it exercises. In addition, write tests for each **public API routine**. Again, annotate each test with the routine under test.

4 Implementation (30 points) - UPDATED

Duration: Monday, October 30th – Monday, December 4th
(5 weeks – in parallel with testing)
Deliverables: Revised design documents, implemented classes
No presentation
Submission: *sc-hs17-groupX*/documents/design.pdf [Revised design document]
sc-hs17-groupX/src/ [Implemented classes]

Use the prioritized functional requirements to decide on what functionality should be implemented first. Make sure that your code satisfies both the functional and the nonfunctional requirements. If you have to change the design or API, revise the design document and add a chapter where you explain what changes were made in the design and why.

There will be no presentation for this milestone. However, there will be an open exercise session 2 weeks before the deadline to discuss unclear points and receive feedback for the planned implementation.

5 Testing and Revision (15 points)

Duration: Until Monday, December 18th
Deliverables: implemented classes
Presentation: December 20th / 21st
Submission: *sc-hs17-groupX*/src/ [Implemented classes]

Given the implementation developed in the previous step and the test plan you have devised, write automated tests for the project. Run the test suite that you have created in the test plan phase. If there were changes in the API, you might have to adapt your tests. Document all bugs that you have uncovered using the issue tracker of your GitHub project.

There is no one answer for devising software projects. Devise an (informal) document were you document and justify all your design decisions (this includes design patterns you used, you can gladly take over text from your design documents).

In the final presentation (10-12 minutes, 8-10 slides) you should describe how you worked on the project, how you distributed the work and handled communication in the group. You can also show a demo or code snippets to show the usability of your library. As a conclusion of the course, give your personal impression of the project: are you happy with the result? Would you do some part differently now?