Software Construction HS 2021

Instructor: Prof. Dr. Alberto Bacchelli — Assignment 4

Teaching Assistants: Enrico Fregnan, Robin Baettig, Jan Bauer, Deborah Jakobi, Matthias Mylaeus, Joel Ruettiman, Lukas Zehnder — Week 11

To correctly complete this assignment you **must**:

- Carry out the assignment with your team only (unless otherwise stated). You are allowed to discuss solutions with other teams, but each team should come up its own personal solution. A strict plagiarism policy is going to be applied to all the artifacts submitted for evaluation.
- Provide solutions to the exercises. Each solution will consist of source code and its changes and/or explanations (*e.g.*, of decisions taken):
  - The explanations must be written in a PDF file with the name: *Group[id on OLAT]-a[AssignmentNumber].pdf*[1]
  - Source code and its changes, as well as explanations must be pushed to the master branch of your GitHub repository by **Dec 21, 2021 @ 16:00**[2] and the tutor who follows your group has to be tagged in the commit they need to consider.

Please note that **this assignment has a reduced workload** compared to the previous ones. Previous assignments were supposed to be have a three-week workload, whereas this assignment has a two-week workload. Therefore it is possible to complete and submit this assignment at least one week before the official deadline. If you decide to submit earlier, inform the tutor who follows your group so that they can start the grading earlier, if they have capacity.

[1] *e.g.*, a correct name would be: *Group1-a4.pdf*.

[2] Solutions sent within the first 24 hours after the deadline will be given 50% of the points they would normally get. Solutions sent after 24 hours from the deadline will not be graded.

Exercise 1 - A Blackjack Game - Design and Implementation (60 pts)

This exercise asks you to *design and implement* a working Blackjack game (please see Appendix A, at the end of this assignment, to find the rules to be used for the game). The game's requirements are the following:

Blackjack's Game Requirements

In this game, one user plays Blackjack via the terminal against the computer who is the Dealer.

At the start of the game, the program shows the amount of money that the Player has (the starting value is 100 CHF). Then, the Dealer asks the Player whether they want to make a bet or go away with the money. If the Player decides to go away with the money, the game finishes. If the Player wants to play, the Dealer asks for the amount to bet in this round. The Player inputs a bet whose amount must be lower or equal than the available money. Then the round starts.

Each round follows the Blackjack rules, also in terms of which cards are visible to the Player and when (e.g., at the start of each round, the Player sees the values of their card, as well as the visible card from the Dealer). The Player has to input their decisions (e.g., 'hit' or 'stay') on the console. When the round is finished, if the money left to the Player is higher than 0, the Dealer asks again whether to play another round or not; otherwise, the game is finished.

The requirements leave freedom to decide how to display information, interact with the console, etc. so that the group can use their creativity. The game must be fully playable.

Use what you learned throughout this course that is relevant to the design and implementation of this game. Document your design process and decisions as you see fit in a PDF document. This document has to be shared together with your implementation.

The exercise will be evaluated on design and implementation quality, based on what was taught in class (e.g., responsibilities, use/misuse of design patterns, design quality, testing). Please see Appendix B, at the end of this assignment, to find the rubrics used for the evaluation of this exercise.

Appendix A: Blackjack' Rules

The following rules have been adapted from the Bicycle Cards:

a. The Player make their bets.

b. The Player is dealt two cards.

c. Dealer is dealt two cards where the second card is hidden from the Player.

d. The objective of the game is to have a higher point total than the dealer (but no more than 21, anything over 21 is an automatic loss called a bust) — if the Player beat the dealer in this way, they win from the casino what they bet (they also win if the dealer busts). Aces can be worth either 1 or 11; every other card is worth its face amount (face cards are worth 10).

e. An initial two card hand composed of an ace and a face card is called a blackjack and is the best possible hand.

f. After the first round of dealing, the Player has the option to hit (receive more cards) or stay (no more cards). If hitting results in the Player busting (total going over 21), then their bet is lost.

g. After the Player is done hitting/staying, the Dealer flips over their hidden card. If the Dealer's total is less than 17, then they need to hit (receive a new card). This process repeats until the Dealer's hand either totals to 17 or more or busts (goes above 21).

h. After the Dealer is done, the final results are decided — if the Dealer busts, then the player who did not bust earlier wins their bet. If the Dealer does not bust, then the Dealer's total is compared to the Player's. If the Player's total is greater than the Dealer's, the Player wins money (in the amount that was bet). If the Player's total is less than the Dealer's, the Player loses money. No money is exchanged in the event of a tie.

Appendix B: Evaluation Rubrics

		Exemplary			Competent			Developing				Weight
		10	9	8	7	6	5	4	3	2	1	
Completeness		All the main requirements are met and additional optional features are added			Almost all the main requirements are met			A few of the main requirements are met				**35%**
Source code quality		Design patterns are well placed and implemented. Methods are of a proper size. No design flaws are present. Other software engineering principles are also used appropriately and correctly.			Design patterns are well placed, though not implemented completely. No major design flaws are present. Other software engineering principles are also used appropriately yet sometimes incorrectly.			Inappropriate use is made of design patterns or anti-patterns are used. Design flaws are spread throughout the code. Other software engineering principles are also used inappropriately or not all.				**40%**
Testing		The system has a >75% of meaningful test coverage (line coverage).			The system has a >45% of meaningful test coverage.			The system has a <20% of meaningful test coverage.				**15%**
Code readability	**Formatting**	Code is well formatted and follows the language's conventions.			Code is well formatted yet diverges from the language's conventions with little motivation.			Code is poorly formatted and does not follow the language's conventions.				**2%**
	Naming	Appropriate and clear names are used for variables, methods, classes, etc.			Most names are clear and appropriate; some names are ambiguous or vague.			Names are vague, ambiguous, or unrelated to their contexts.				**5%**
	Comments	Appropriate use is made of comments to document the code correctly.			Most comments are appropriately used yet some lack purpose or are unnecessary.			Little to no comments of value; code is poorly documented via comments.				**3%**