Rollins College

Rollins Game Review Forum Report

Desmond Nieves - ArrayList & Back-End Implementation

Stephan Zambrano - Front end Implementation of GUI

PJ González - Archiving & Back-End Assistance

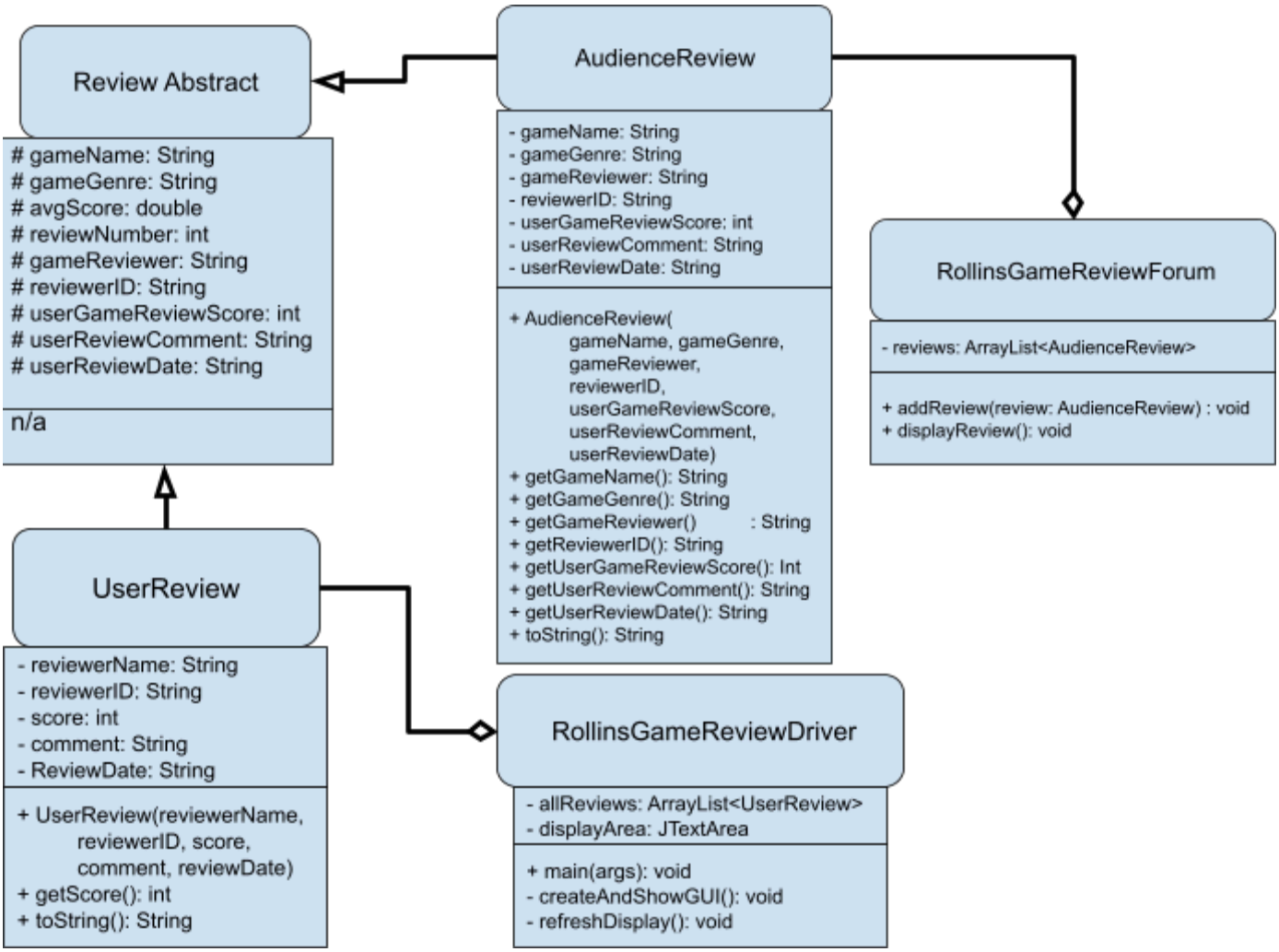
CMS 270: Object Oriented Design and Development

Doctor Sirazum Tisha

The 30th of November of 2025

The *Rollins Game Review Forum* is a forum for students to create and view video games reviews. Just like any other review forum, Rollins Game Review Forum provides ease of access for Rollins Students to communicate with one another over prior, new, and upcoming video games. Students would, hypothetically, be able to leave a review for any video game that's been released. However, to prevent malpractice and misuse from non-Rollins students, there is the requirement that in order to leave a review, a Student ID is required. Emphasizing the necessity for ID, a student or user would only need to offer their Rollins student ID if they want to make and post a review onto the forum, and similarly to most forums, users can still view and read reviews or posts made by other users freely without their Student ID (An ID is only necessary to post a review, not to view).

While there are a handful of websites and resources available for every student to use at their leisure, there were none whatsoever for video games. As such, given that we, as a team, really like video games, over the course of an hourlong discussion, the idea of making an application for Rollins students to talk or write about video games was mutually agreed upon to be a novel concept. Especially given that a majority of people play video games regularly nowadays. Consequently, it was agreed that making a Forum for video game reviews would be the most relevant & interesting angle for our project.



Throughout the project, we had to ensure that we correctly utilized Object-Oriented Programming (OOP) principles. As a result of that necessity, when making our original UML diagram, we wanted to immediately establish how the project's classes would relate and inherit properties from one another. Consequently, the idea of an abstract class came into mind when defining audienceReview and userReview classes. Afterall, both classes shared a handful of elements and variables to such an extent that it's safe to refer to them as identical.

Therefore, reviewAbstract was made to contain all of the possible elements and values within a review, while preventing code repetition. Then, the two aforementioned classes, audienceReview and userReview, relate to reviewAbstract by extending it, and using the values it defines for complex methods and more importantly, to define the values within a given review. In other words, the audience and user review classes inherit the protected fields from the abstract class. Furthermore, because the fields within the abstract are protected, they can be accessed not only within the abstract file, but also within any related subclasses.

Related, protected access modifiers are not only useful for abstract classes, but they're effective at encapsulating the values within protected fields. Within audience and user review classes, both utilize the protected (abstracted) fields, resulting in successful protection of internal data. Given that the fields are protected, getter methods were implemented in order to avoid code misuse (also because getters are just simply good to have).

Within the RollinsGameReviewForum class, internal logic for storing reviews was defined, however amongst its methods, only two are public: addReview() and displayReview(). Aforementioned interior logic is encapsulated to prevent unauthorized use (from anyone who does not have access to the github repository) while simultaneously obscuring the methods and manner by which the data gets stored in the ArrayList.

Getting a bit visual, encapsulation is also present in the manner by which the Graphical User Interface (GUI) is defined within RollinsGameReviewDriver. Said driver contains a private text field and handles the user's input internally. Upon running, should the user want to add a new review to the forum, the GUI needs to have some form of text boxes for the user to fill in with their review (i.e. their ID, name, review comment, etc.). As such, the GUI, though it does load and display prior data from the ArrayList, also creates new instances of UserReviews after the user's input is properly validated alongside their Student ID (for safety reasons).

Consequently, due to the manner by which methods are defined, alongside the implementation of abstracted fields, certain inherited methods are overridden as appropriate. Effectively, each class has its own respective definition of any given method, and because the program interacts with reviews via a shared interface, new reviews and review types could be defined with minimal changes to existing code structure. The result is that all review objects within the ArrayList are swiftly displayed within the GUI's panels.

However, for all that we accomplished with the physical program we still encountered struggle and difficulties as a group and individually. Though we struggled trying to iron out a proper way to execute a program of this caliber, our greatest fault was a mixture of communication and time crunch.

To state briefly, all three of us have relatively busy schedules that hindered that amount of time we could dedicate to work together on the program. In effect, we were meeting once or twice a week if possible, however for a complex program like this, once or twice a week, even if for a few hours, does not always guarantee a generous base line to make good progress. As a direct consequence, a majority of our work had to be accomplished remotely, with our primary method of communication being over emails and text messages. Though such methods are not necessarily bad, they can potentially harm the group's cohesion and by extension the project's cohesion.

Two separate instances of messy communication resulted in the loss of hours of work and progress. However, despite our communication shortcomings, we still came together and discussed as a group how we could avoid repeating these issues by spamming the group chat and outlook email whenever we'd begun to work and finished working on the program.

In terms of contribution, Desmond took charge of half of the back end. Particularly, he was the one to suggest the implementation of an arrayList for the forum and ended up volunteering to create a mock arrayList to help showcase what the Rollins Game Review Forum would look like in its prime.

Likewise, Stephan defined the userReview and GameReviewDriver classes. Over the course of thanksgiving break, he went out of his way to practice GUI implementation with the intent of applying what he learnt to the program. As a result, he retroactively redefined the classes he had previously worked on to make them compatible with the GUI we wanted to implement.

Lastly, PJ provided assistance towards the group by defining the audience review and GameReviewForum and Driver classes as necessary while simultaneously focusing on archiving and code compatibility. Aside from archiving and documenting the group's efforts, PJ helped lay the foundation of the program's classes and methods at the beginning of the project and added notes elaborating on what each class did and how they'd interact with one another.