Erik Roganský
Wednesday 18.00 - 19.40

Open Door's Day at Hogwarts

1. Project Objective

The goal of this project is to design and develop an interactive application that allows users to plan and take a virtual tour of Hogwarts, the famous school of witchcraft and wizardry in the Harry Potter universe. The application is targeted towards fans of the Harry Potter series who wish to explore the castle and learn more about the different subjects taught in the school.

The application will begin with a character creation process, where the user will choose their name, gender, and preferred house from the four houses of Hogwarts: Gryffindor, Hufflepuff, Ravenclaw, and Slytherin. Additionally, the user will select their interests from a list of options, such as potions, charms, divination, and care of magical creatures.

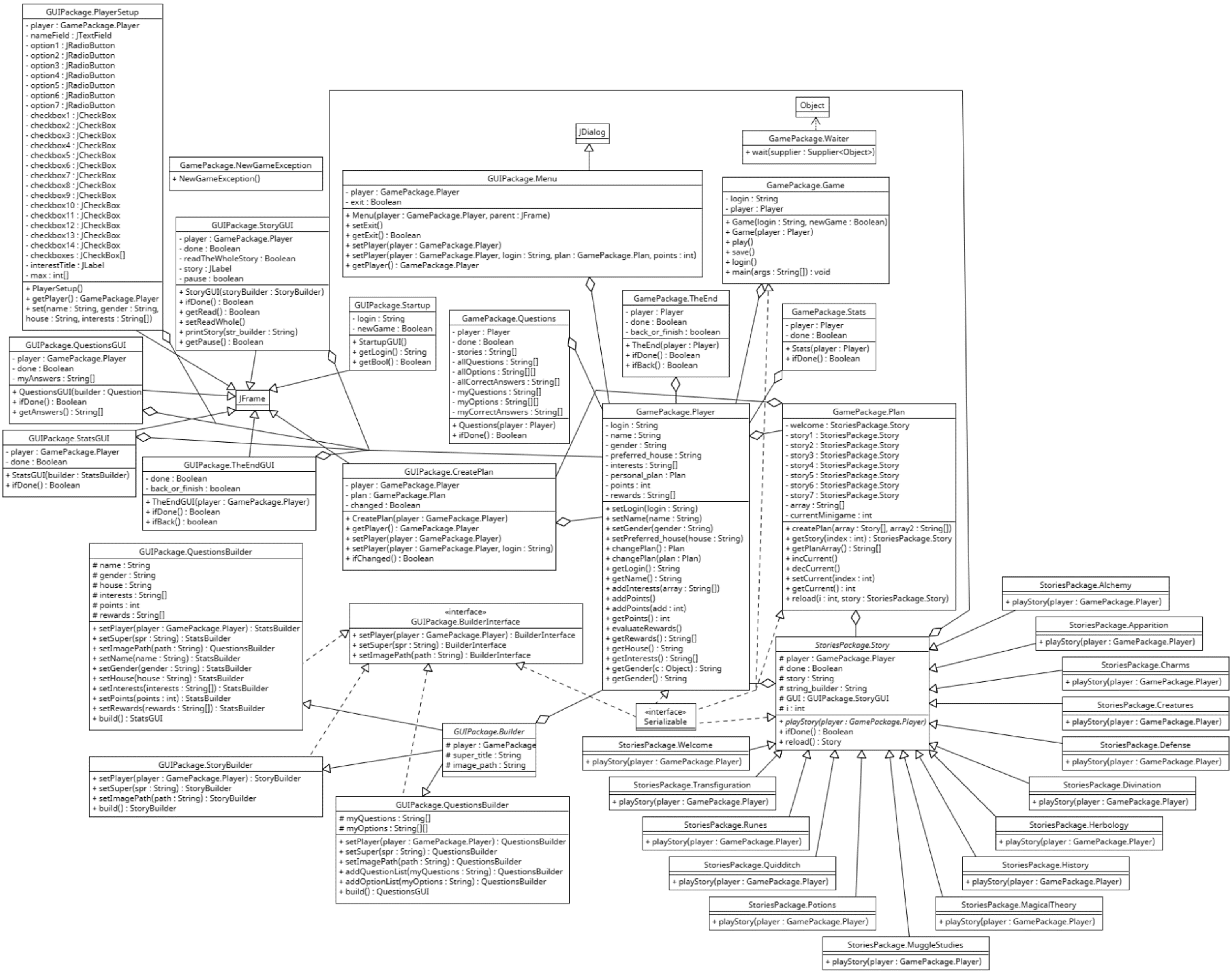
Based on the user's choices, the application will generate a personalized itinerary for the tour. The itinerary will be adjustable, and users can modify it to suit their preferences. Each place on the tour will be accompanied by a brief description and explanation of the subjects taught in that location.

Once the user has completed their tour, they will be presented with a quiz that is personalized based on the places they visited. The quiz will test the user's knowledge of the subjects they encountered during the tour and will provide feedback on their performance. The user's statistics and performance will be then saved and displayed.

To make the experience more engaging, users will be able to earn rewards for completing the tour and performing well on the quiz.

Overall, this application will provide an immersive and educational experience for fans of the Harry Potter series, allowing them to explore Hogwarts and learn more about the magic and lore of the series.

2. Class Diagram


GUIPackage.PlayerSetup
- player : GamePackage.Player
- nameField : JTextField
- option1 : JRadioButton
- option2 : JRadioButton
- option3 : JRadioButton
- option4 : JRadioButton
- option5 : JRadioButton
- option6 : JRadioButton
- option7 : JRadioButton
- checkbox1 : JCheckBox
- checkbox2 : JCheckBox
- checkbox3 : JCheckBox
- checkbox4 : JCheckBox
- checkbox5 : JCheckBox
- checkbox6 : JCheckBox
- checkbox7 : JCheckBox
- checkbox8 : JCheckBox
- checkbox9 : JCheckBox
- checkbox10 : JCheckBox
- checkbox11 : JCheckBox
- checkbox12 : JCheckBox
- checkbox13 : JCheckBox
- checkbox14 : JCheckBox
- checkboxes : JCheckBox[]
- interestTitle : JLabel
- max : int[]
+ PlayerSetup()
+ getPlayer() : GamePackage.Player
+ set(name : String, gender : String, house : String, interests : String[])
GamePackage.NewGameException
+ NewGameException()
GUIPackage.StoryGUI
- player : GamePackage.Player
- done : Boolean
- readTheWholeStory : Boolean
- story : JLabel
- pause : boolean
+ StoryGUI(storyBuilder : StoryBuilder)
+ ifDone() : Boolean
+ getRead() : Boolean
+ setReadWhole()
+ printStory(str_builder : String)
+ getPause() : Boolean
JDialog
Object
GamePackage.Waiter
+ wait(supplier : Supplier<Object>)
GUIPackage.Menu
- player : GamePackage.Player
- exit : Boolean
+ Menu(player : GamePackage.Player, parent : JFrame)
+ setExit()
+ getExit() : Boolean
+ setPlayer(player : GamePackage.Player)
+ setPlayer(player : GamePackage.Player, login : String, plan : GamePackage.Plan, points : int)
+ getPlayer() : GamePackage.Player
GamePackage.Game
- login : String
- player : Player
+ Game(login : String, newGame : Boolean)
+ Game(player : Player)
+ play()
+ save()
+ login()
+ main(args : String[]) : void
GUIPackage.QuestionsGUI
- player : GamePackage.Player
- done : Boolean
- myAnswers : String[]
+ QuestionsGUI(builder : Question
+ ifDone() : Boolean
+ getAnswers() : String[]
JFrame
GUIPackage.Startup
- login : String
- newGame : Boolean
+ StartupGUI()
+ getLogin() : String
+ getBool() : Boolean
GamePackage.Questions
- player : Player
- done : Boolean
- stories : String[]
- allQuestions : String[]
- allOptions : String[][]
- allCorrectAnswers : String[]
- myQuestions : String[]
- myOptions : String[][]
- myCorrectAnswers : String[]
+ Questions(player : Player)
+ ifDone() : Boolean
GamePackage.TheEnd
- player : Player
- done : Boolean
- back_or_finish : boolean
+ TheEnd(player : Player)
+ ifDone() : Boolean
+ ifBack() : Boolean
GamePackage.Stats
- player : Player
- done : Boolean
+ Stats(player : Player)
+ ifDone() : Boolean
GUIPackage.StatsGUI
- player : GamePackage.Player
- done : Boolean
+ StatsGUI(builder : StatsBuilder)
+ ifDone() : Boolean
GUIPackage.TheEndGUI
- done : Boolean
- back_or_finish : boolean
+ TheEndGUI(player : GamePackage.Player)
+ ifDone() : Boolean
+ ifBack() : boolean
GUIPackage.CreatePlan
- player : GamePackage.Player
- plan : GamePackage.Plan
- changed : Boolean
+ CreatePlan(player : GamePackage.Player)
+ getPlayer() : GamePackage.Player
+ setPlayer(player : GamePackage.Player)
+ setPlayer(player : GamePackage.Player, login : String)
+ ifChanged() : Boolean
GamePackage.Player
- login : String
- name : String
- gender : String
- preferred_house : String
- interests : String[]
- personal_plan : Plan
- points : int
- rewards : String[]
+ setLogin(login : String)
+ setName(name : String)
+ setGender(gender : String)
+ setPreferred_house(house : String)
+ changePlan() : Plan
+ changePlan(plan : Plan)
+ getLogin() : String
+ getName() : String
+ addInterests(array : String[])
+ addPoints()
+ addPoints(add : int)
+ getPoints() : int
+ evaluateRewards()
+ getRewards() : String[]
+ getHouse() : String
+ getInterests() : String[]
+ getGender(c : Object) : String
+ getGender() : String
GamePackage.Plan
- welcome : StoriesPackage.Story
- story1 : StoriesPackage.Story
- story2 : StoriesPackage.Story
- story3 : StoriesPackage.Story
- story4 : StoriesPackage.Story
- story5 : StoriesPackage.Story
- story6 : StoriesPackage.Story
- story7 : StoriesPackage.Story
- array : String[]
- currentMinigame : int
+ createPlan(array : Story[], array2 : String[])
+ getStory(index : int) : StoriesPackage.Story
+ getPlanArray() : String[]
+ incCurrent()
+ decCurrent()
+ setCurrent(index : int)
+ getCurrent() : int
+ reload(i : int, story : StoriesPackage.Story)
GUIPackage.QuestionsBuilder
# name : String
# gender : String
# house : String
# interests : String[]
# points : int
# rewards : String[]
+ setPlayer(player : GamePackage.Player) : StatsBuilder
+ setSuper(spr : String) : StatsBuilder
+ setImagePath(path : String) : QuestionsBuilder
+ setName(name : String) : StatsBuilder
+ setGender(gender : String) : StatsBuilder
+ setHouse(house : String) : StatsBuilder
+ setInterests(interests : String[]) : StatsBuilder
+ setPoints(points : int) : StatsBuilder
+ setRewards(rewards : String[]) : StatsBuilder
+ build() : StatsGUI
«interface»
GUIPackage.BuilderInterface
+ setPlayer(player : GamePackage.Player) : BuilderInterface
+ setSuper(spr : String) : BuilderInterface
+ setImagePath(path : String) : BuilderInterface
«interface»
Serializable
GUIPackage.Builder
# player : GamePackage
# super_title : String
# image_path : String
GUIPackage.StoryBuilder
+ setPlayer(player : GamePackage.Player) : StoryBuilder
+ setSuper(spr : String) : StoryBuilder
+ setImagePath(path : String) : StoryBuilder
+ build() : StoryBuilder
GUIPackage.QuestionsBuilder
# myQuestions : String[]
# myOptions : String[][]
+ setPlayer(player : GamePackage.Player) : QuestionsBuilder
+ setSuper(spr : String) : QuestionsBuilder
+ setImagePath(path : String) : QuestionsBuilder
+ addQuestionList(myQuestions : String) : QuestionsBuilder
+ addOptionList(myOptions : String) : QuestionsBuilder
+ build() : QuestionsGUI
StoriesPackage.Story
# player : GamePackage.Player
# done : Boolean
# story : String
# string_builder : String
# GUI : GUIPackage.StoryGUI
# i : int
+ playStory(player : GamePackage.Player)
+ ifDone() : Boolean
+ reload() : Story
StoriesPackage.Alchemy
+ playStory(player : GamePackage.Player)
StoriesPackage.Apparition
+ playStory(player : GamePackage.Player)
StoriesPackage.Charms
+ playStory(player : GamePackage.Player)
StoriesPackage.Creatures
+ playStory(player : GamePackage.Player)
StoriesPackage.Defense
+ playStory(player : GamePackage.Player)
StoriesPackage.Welcome
+ playStory(player : GamePackage.Player)
StoriesPackage.Transfiguration
+ playStory(player : GamePackage.Player)
StoriesPackage.Divination
+ playStory(player : GamePackage.Player)
StoriesPackage.Runes
+ playStory(player : GamePackage.Player)
StoriesPackage.Herbology
+ playStory(player : GamePackage.Player)
StoriesPackage.Quidditch
+ playStory(player : GamePackage.Player)
StoriesPackage.History
+ playStory(player : GamePackage.Player)
StoriesPackage.Potions
+ playStory(player : GamePackage.Player)
StoriesPackage.MagicalTheory
+ playStory(player : GamePackage.Player)
StoriesPackage.MuggleStudies
+ playStory(player : GamePackage.Player)

3. Evaluation criteria

3.1. Main

As the author of this project, I am confident that I have successfully fulfilled the main criteria set out for the project. Firstly, the project meets the given topic and my objective, with some modifications to ensure that it is correctly done.

One of the key features of my project is the implementation of inheritance in two separate hierarchies. The first hierarchy extends from the abstract 'Story' class, while the second hierarchy comprises builders extended from the abstract 'Builder' class. In the case of the builders, I have also utilized an interface. This approach has allowed me to write more efficient and reusable code and has helped to ensure that the project is easier to maintain and modify in the future.

Another important aspect of my project is the use of polymorphism in various classes. For example, the 'Player' class includes the methods 'addPoints()' and 'addPoints(int add)', which demonstrate how polymorphism can be used to support different operations based on the context. In the first case, the points are incremented by a default value, while in the second case, a specific amount is added. This approach throughout my whole code not only improves the readability of the code but also allows for greater flexibility when making modifications.

I have also taken care to ensure that every class where necessary is encapsulated correctly. This means that each class is self-contained and cannot be affected by changes made outside of its scope. This approach helps to reduce errors and make the code more robust.

In terms of relationships between classes, I have extensively used aggregation throughout virtually the whole project, as can be seen in the diagram provided. Aggregation is a key principle in object-oriented programming, and its use here has helped to simplify the design and ensure that the code is more modular and easier to maintain.

Finally, I have taken great care to ensure that my code is correctly organized, and thoroughly commented. I have also provided detailed documentation that covers all aspects of the project, from the architecture and design patterns used to the specific implementation details.

In summary, I am confident that my project meets the main criteria set out for it and that it represents a well-designed, efficient, and maintainable implementation of the objectives outlined.

3.2. Secondary

I believe I have met 7 of the secondary requirements, from which GUI deducts 2, making a total of 8 secondary requirements met:

1. Implemented the Builder design pattern in the 'GUIPackage' module, using classes such as 'BuilderInterface', 'Builder', 'StoryBuilder', 'QuestionsBuilder', and 'StatsBuilder'. This pattern was utilized for example in 'StoriesPackage.Charms' to build various game elements in a flexible and maintainable way.

2. Implemented a Try/Catch exception mechanism in the 'GamePackage' module using the 'NewGameException' class, which was utilized in the 'Game' class to handle new game scenarios.

3. Developed a Graphical User Interface (GUI) in the 'GUIPackage' module that includes a range of features, such as interactive menus, story printing, and quiz. This module comprises 2 separate criteria.

4. Employed Multithreading for instance in the 'StoriesPackage' module in every class that extends the 'Story' class. This was accomplished by creating a new thread that could run simultaneously with the main thread, allowing for more efficient processing of gameplay elements.

5. Utilized Runtime Type Identification (RTTI) in the 'StoriesPackage' module, specifically within the 'Story' class. This feature enabled the program to identify the specific type of object at runtime, allowing the class it was called from to create a new instance of it.

6. Employed Lambda expressions in every 'ifDone()' method and more throughout the program. This feature was used to call the ‘wait()’ method in ‘Waiter’ class that uses a ‘Supplier’.

7. Implemented Serialization in the 'Game', ‘BuilderInterface’, ‘Player’, ‘Plan’ and ‘Story’ classes, allowing game data to be saved and loaded easily. Serialization was employed to convert game data to a stream of bytes that could be stored and retrieved efficiently.

4. Commits

Commits on May 11, 2023

1. Final Version
2. Stories finished and some debugging

Commits on May 10, 2023

1. Optimized and diagram repaired
2. Repaired stats
3. Added diagram
4. Javadoc done
5. Game Javadoc comment done
6. All methods and constructors commented with Javadoc

Commits on May 9, 2023

1. GUI javadoc start
2. Building stories slightly changed
3. Story package javadoc done
4. Package game probably done (no debugging yet)
5. README update
6. Waiter optimized and new Javadoc added
7. Builder pattern changed
8. Builder 2 added and Javadoc beginning

Commits on May 8, 2023

1. Builder pattern added
2. Stats changed and reload methods added
3. Pause story printing
4. new pc test?

Commits on May 7, 2023

1. small changes

Commits on May 6, 2023

1. Menu, change player and similar changes

Commits on May 5, 2023

1. Stats look changed
2. End added
3. Added rewards to stats

Commits on May 4, 2023

1. Added stats
2. Added quiz
3. Welcome changed to work like stories

Commits on May 3, 2023

1. Apparition

Commits on May 1, 2023

1. private
2. Hagrid
3. A little change in creating a new game
4. Potions
5. Herbology
6. Little changes to the jlabel sizes and renamed to story context
7. Moved story printing to backend and DADA done

Commits on Apr 19, 2023

1. DADA - beginning
2. Added interest counting
3. Added Back buttons in setup and planning

Commits on Apr 18, 2023

1. KeyListeners in setup and plan done

Commits on Apr 16, 2023

1. KeyListeners in player setup almost done
2. KeyListeners in player setup pt1
3. Quidditch story
4. Added some keyListeners
5. Charms and transfiguration stories
6. A little changes
7. Add points after reading the whole story again
8. Add points after reading the whole story

Commits on Apr 12, 2023

1. MiniGames -> Stories

Commits on Apr 10, 2023

1. README add

Commits on Apr 6, 2023

1. Final check
2. check
3. Finish initial upload
4. Upload

5. Conclusion

After thoroughly reviewing the project requirements and dedicating significant time and effort to the task, I am confident that I have met all the necessary conditions. It is possible that small bugs may exist in the game's edge cases that I have not considered. However, it is worth noting that new bugs can surface in any game, even years after its release.

I believe that the game is functioning as intended, but I recognize that there is always room for improvement. If any issues arise, I am prepared to address them promptly and efficiently. Overall, I am content with the results of the project and believe that I have met the requirements set forth in the project specifications.