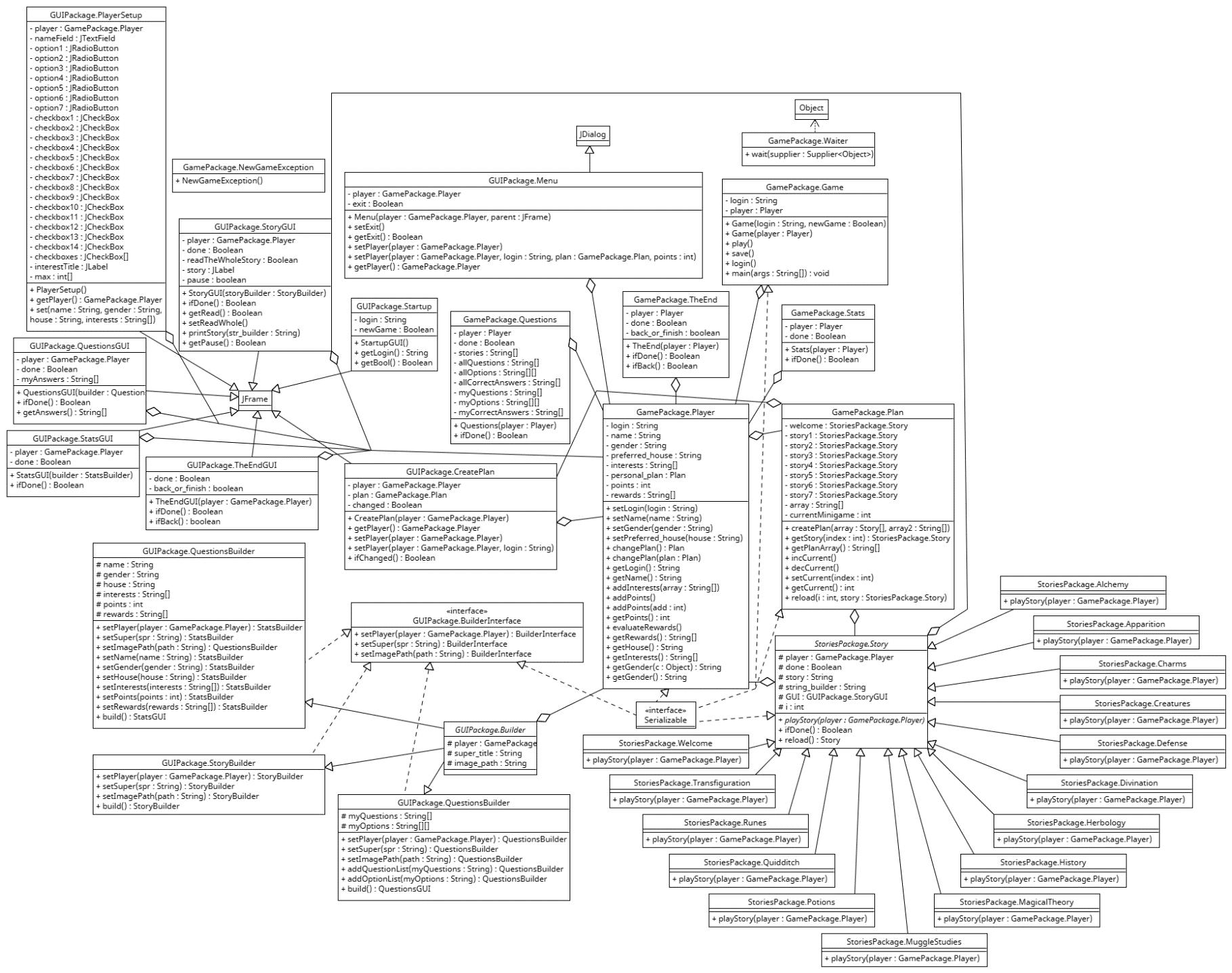
**Open Door’s Day at Hogwarts**

1. **Project Objective**

The assignment was to create an application that would somehow implement planning of a trip. As for my application, I’ve chosen to create a game, set in the Harry Potter universe, concretely in Hogwarts. The story is that a teenager comes to Open Door’s Day to Hogwarts to gain their first impression. However, the castle is big, and they won’t be able to go everywhere, therefore they need to plan the trip. After the planning, the player visits the places they’ve chosen where the teachers explain the subjects of their specialization. The player can, however, decide to change the plan after the visiting places had started, and they can freely do that, however, no visitor can visit the same place more than once. After the trip, there is a short quiz, personalized according to the places visited. In the end the player gets to know how they did, so they’ll receive statistics. The player can be customized.

1. **Class Diagram**
2. **Evaluation criteria**
   1. **Main**

In my opinion, I fulfilled the main criteria of the project. My project meets the given topic and, for the most part, with some changes, meets my objective.

The project also contains inheritance in two separate hierarchies. The first ones are extended from the abstract ‘Story’ class and the second ones are builders extended from the ‘Builder’ class. When it comes to the builders, I also used an interface there. I used polymorphism in many classes, for instance in the ‘Player’ class ‘addPoints()’ and ‘addPoints(int add)’ where in the first case the points are simply incremented and in the second case there is a specific amount added. Every class where it is necessary is encapsulated correctly. The aggregation is used throughout virtually the whole project as can be seen in the diagram above.

I believe that my code is correctly organized, and I believe that the comments I wrote are more than enough.

* 1. **Secondary**

I believe that I fulfilled 8 of the secondary criteria:

1. Builder design pattern in ‘GUIPackage’ – classes ‘BuilderInterface’, ‘Builder’, ‘StoryBuilder’, ‘QuestionsBuilder’ and ‘StatsBuilder’ – used in e.g. ‘StoriesPackage.Charms’
2. Try/catch exception in ‘GamePackage’ – class ‘NewGameException’ – used in ‘Game’
3. GUI – which counts as 2 criteria - ‘GUIPackage’
4. Multithreading in ‘StoriesPackage’ in every class that extends ‘Story’
5. RTTI in ‘StoriesPackage’ – class ‘Story’
6. Lambda expressions in every ifDone() method and more
7. Serialization e.g. in ‘Game’ class
8. **GitHub**