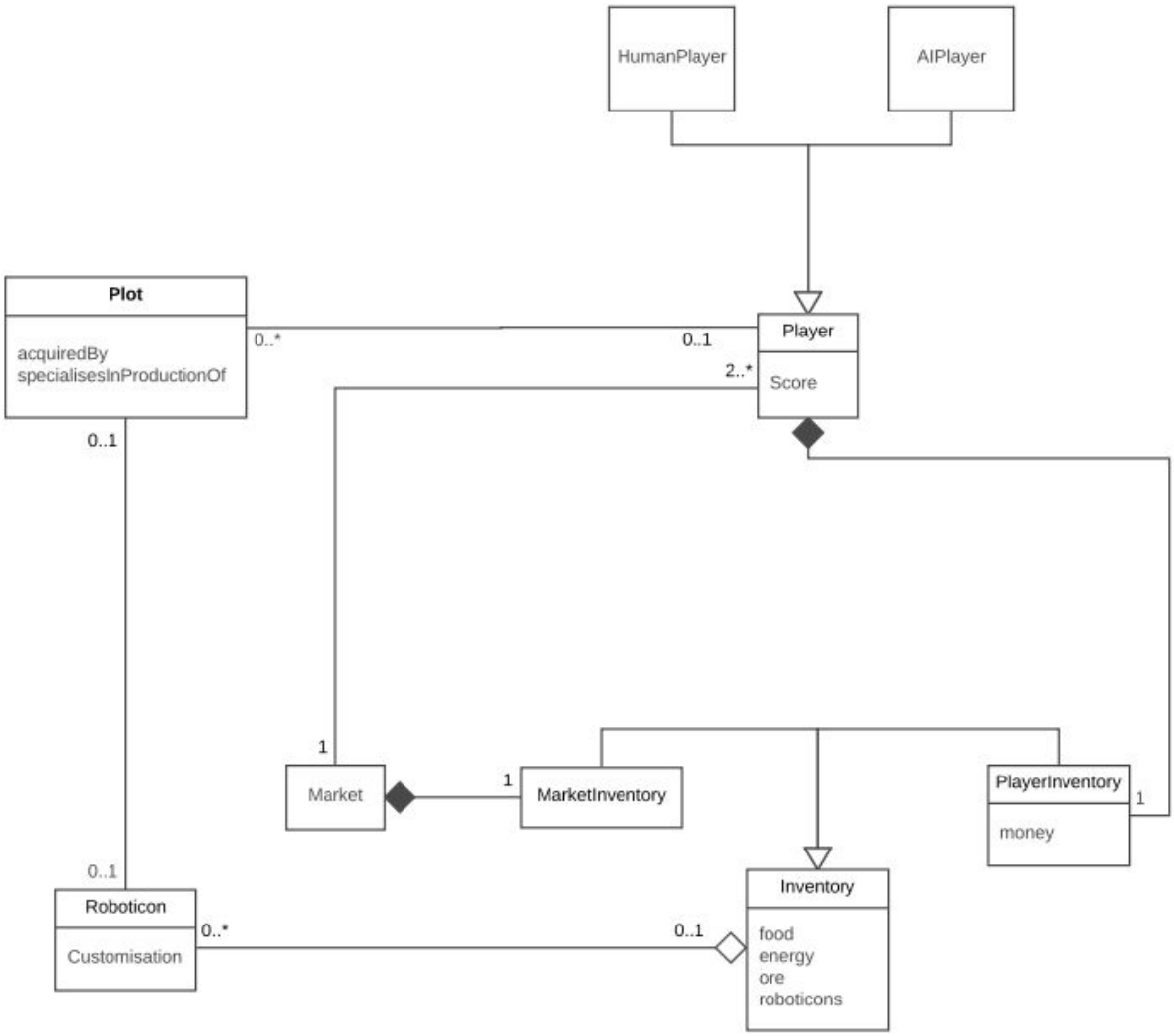
Top Right Corner

Conceptual architecture

We created a UML class diagram for the conceptual architecture of our system. We used the UML 2 class diagram notation and used Lucidchart to create it.



Justification

Use cases, collaboration diagrams, sequence diagram

We created a number of use cases describing the steps a user might take to perform various tasks and then created UML 2 collaboration diagrams and a UML 2 sequence diagram (using Lucidchart) to visualize the behaviour of the system and the interaction of its components according to these use cases (use cases and diagrams are available here: https://sepr-topright.github.io/SEPR/documentation/assessment1/extras/UMLclassdiagramjustificationextras.pdf). The use cases implemented many of the system requirements and helped to show the correctness of the model as it supports the necessary communication links between components. We decided not to include these diagrams in this report as we didn't want to stick with the method names and exact processes they define, they simply acted to help show that the architecture was valid and provided the necessary communication links.

How the classes and associations satisfy (or facilitate) the requirements and justification of the class structure

The PlayerInventory class satisfies requirements 1.5 as it stores all the information needed to keep track of the resources, roboticons and money that the players and market may be in possession of in its attributes. It also facilitates requirement 1.6 as when money is first allocated to a player it may be stored in the money attribute of this class. As well as facilitating requirements 1.12, 1.13, 1.18, 1.19, 1.20 and 1.24 as all of these requirements involve the use or modification of information concerning the players money, resources or roboticons.

The MarketInventory class satisfies requirement 1.7 as the necessary information is stored in this class' attributes to keep track of the resources and robiticons that the market is in possession of. It also facilities requirement 1.8 as when resources and roboticons are first allocated to the market the information can again be stored in the attributes of this class. It also facilitates system requirements 1.12, 1.18, 1.19, 1.20 as all these requirements require information about the resources or roboticons that the market is in possession of.

It is shown that the PlayerInventory and MarketInventory classes are related to the Player and Market classes respectively by way of composition as the inventories only have meaning when associated with either a market or player object (as they contain information relating to the resources and money that belong to a specific player or market object). These associations also have multiplicity of 1 on the inventory end of the association as each player (and the market) has a single unique inventory. The player and market inventories are represented by separate classes as the player's inventory has to store money, whilst the markets inventory does not (see requirements 1.5 and 1.7). However due to their similarities it makes sense that both are specialisations of an inventory class capable storing all elements that both classes must store (food, ore, roboticons and energy). The inventory class is associated with the Roboticon class as it must store roboticons and the association is an aggregation as inventories may contain roboticons. The multiplicities are 0..1 and 0..* as each roboticon may belong to at most one inventory (once it has been placed on a plot it belongs to none) but an inventory could hold any number of roboticons.

The Plot class helps satisfy requirement 1.3 as it shows that the system can support a number plots. It satisfies requirement 1.4 as its attribute acquiredBy is there to determine whether or not it has been acquired, and if so which player has acquired it. The acquiredBy attribute is also necessary to facilitate requirement 1.10 as once all plots have been acquired the game must stop. Its attribute specialisesInProductionOf helps satisfy requirement 1.9 by recording which resource the plot is best at producing. Its association with the Roboticon class exists with the multiplicities of 0..1 as it must be possible for a plot of land to have a single roboticon placed upon it to satisfy requirement 1.15 (but a roboticon does not have to be placed on any plot and a plot does not have to have a roboticon placed on it). Its association with the Roboticon class also facilitates requirement 1.16 (cause each plot with a roboticon on it to produce resources). The Plot class' association with Player exists in order to satisfy requirements 1.4 (as each plot may be acquired by a certain player), 1.15 (as each player must be able to place a roboticon on plots they own) and 1.16 (as players must receive the resources produced by each plot that they own). The multiplicities of the association between the Player and Plot class are 0..1 and 0..* in order to satisfy requirement 1.26 as each plot can only be acquired by one player (but is initially unacquired) and each player can acquire many plots (but initially has none).

The Player class exists as many game elements such as roboticons, player inventories and plots of land must be associated with or interact a unique player at some point during the game. Hence why the associations between the Player class and the Roboticon, PlayerInventory and Plot classes exist to ensure that requirements 1.4, 1.5 and 1.11 are met.

The Player class has a score attribute as the system must calculate each player's score in order to satisfy requirements 1.22 (calculating each player's final score) and 1.23 (reporting which player is the winner based on their finals scores).

The Market class is associated with the Player class as it must be possible for players to interact with the market in order to:

- buy from and sell to it
- enter into an auction with other players and it
- to play a game of chance via the market

(i.e. in order to implement requirements 1.12, 1.17, 1.18, 1.19, 1.20 and 1.24). The multiplicities of the association are given as 2..* and 1 as the market must be able to interact with at least 2 players (as the system should support at least 2 player to satisfy requirement 1.1) and each player should only be able to interact with one market (THE market).

The HumanPlayer and AIPlayer specialisations of the Player class exist in order to satisfy requirement 1.1. In order for the system to allow at least one human player to play the game against at least one computer controlled player it must support both types of player (and they must both have all the associations that the Player class has).

The Roboticon class has the attribute customisation in order to satisfy requirement 1.13 (it must be possible to customise roboticons) and to facilitate requirement 1.15 (as it must only be possible to place roboticons that have been customised).