DESIGN PATTERNS IN C++ ASSIGNMENT 1

DESIGN DOCUMENT

Matthew Freak
Fremk005 - 110348401

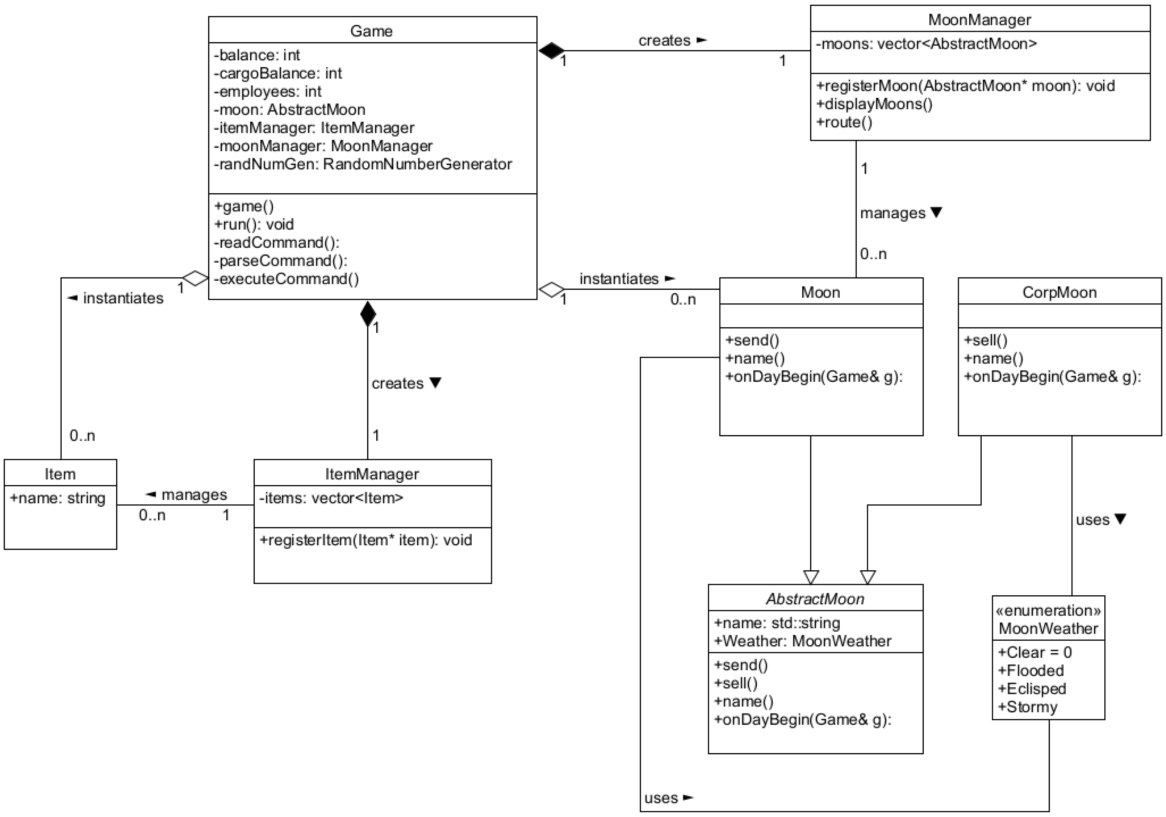

Figure 1: Deadly Corp UML version 1

Notable design decisions

- I have decided to create a class called Item which will act as a data structure to hold all information relevant to item stats etc.
- I have created two classes inheriting from abstract moon one for the corporation moon and one for the rest of the moons.
- I have chosen to use vectors to store items and moons as they are flexible, easy to use, fast, and ordered.

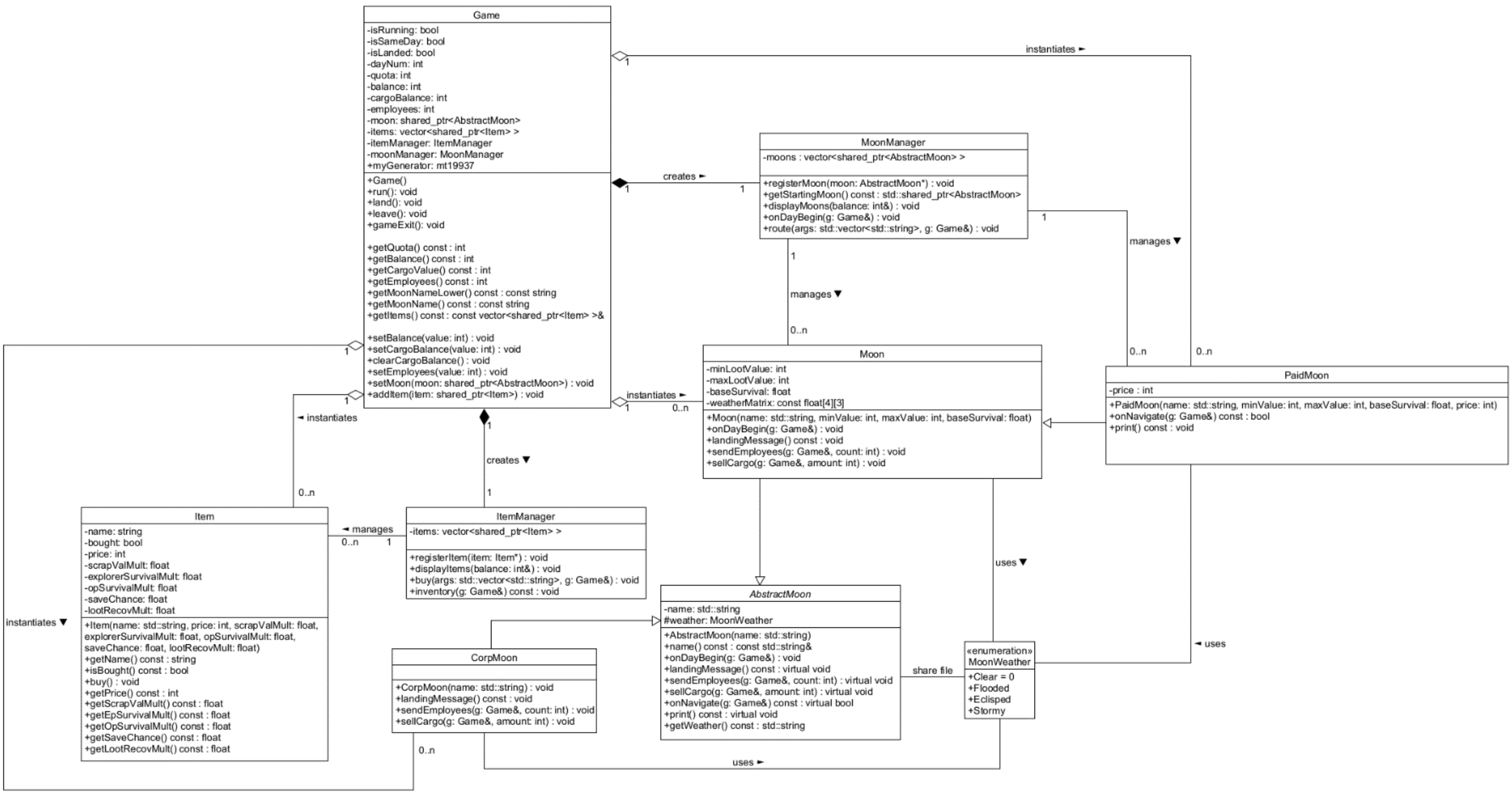
Game
-isRunning: bool
-isSameDay: bool
-isLanded: bool
-dayNum: int
-quota: int
-balance: int
-cargoBalance: int
-employees: int
-moon: shared_ptr<AbstractMoon>
-items: vector<shared_ptr<Item> >
-itemManager: ItemManager
-moonManager: MoonManager
+myGenerator: mt19937
+Game()
+run(): void
+land(): void
+leave(): void
+gameExit(): void
+getQuota() const : int
+getBalance() const : int
+getCargoValue() const : int
+getEmployees() const : int
+getMoonNameLower() const : const string
+getMoonName() const : const string
+getItems() const : const vector<shared_ptr<Item> >&
+setBalance(value: int) : void
+setCargoBalance(value: int) : void
+clearCargoBalance() : void
+setEmployees(value: int) : void
+setMoon(moon: shared_ptr<AbstractMoon>) : void
+addItem(item: shared_ptr<Item>) : void
instantiates ►
1
0..n
creates ►
1
1
MoonManager
-moons : vector<shared_ptr<AbstractMoon> >
+registerMoon(moon: AbstractMoon*) : void
+getStartingMoon() const : std::shared_ptr<AbstractMoon>
+displayMoons(balance: int&) : void
+onDayBegin(g: Game&) : void
+route(args: std::vector<std::string>, g: Game&) : void
1
manages ▼
0..n
manages ▼
1
0..n
0..n
Moon
-minLootValue: int
-maxLootValue: int
-baseSurvival: float
-weatherMatrix: const float[4][3]
+Moon(name: std::string, minValue: int, maxValue: int, baseSurvival: float)
+onDayBegin(g: Game&) : void
+landingMessage() const : void
+sendEmployees(g: Game&, count: int) : void
+sellCargo(g: Game&, amount: int) : void
PaidMoon
-price : int
+PaidMoon(name: std::string, minValue: int, maxValue: int, baseSurvival: float, price: int)
+onNavigate(g: Game&) const : bool
+print() const : void
instantiates ►
1
0..n
◄ instantiates
1
0..n
creates ▼
1
1
Item
-name: string
-bought: bool
-price: int
-scrapValMult: float
-explorerSurvivalMult: float
-opSurvivalMult: float
-saveChance: float
-lootRecovMult: float
+Item(name: std::string, price: int, scrapValMult: float, explorerSurvivalMult: float, opSurvivalMult: float, saveChance: float, lootRecovMult: float)
+getName() const : string
+isBought() const : bool
+buy() : void
+getPrice() const : int
+getScrapValMult() const : float
+getEpSurvivalMult() const : float
+getOpSurvivalMult() const : float
+getSaveChance() const : float
+getLootRecovMult() const : float
◄ manages
0..n
1
ItemManager
-items: vector<shared_ptr<Item> >
+registerItem(item: Item*) : void
+displayItems(balance: int&) : void
+buy(args: std::vector<std::string>, g: Game&) : void
+inventory(g: Game&) const : void
instantiates ▼
1
0..n
uses ▼
AbstractMoon
-name: std::string
#weather: MoonWeather
+AbstractMoon(name: std::string)
+name() const : const std::string&
+onDayBegin(g: Game&) : void
+landingMessage() const : virtual void
+sendEmployees(g: Game&, count: int) : virtual void
+sellCargo(g: Game&, amount: int) : virtual void
+onNavigate(g: Game&) const : virtual bool
+print() const : virtual void
+getWeather() const : std::string
CorpMoon
+CorpMoon(name: std::string) : void
+landingMessage() const : void
+sendEmployees(g: Game&, count: int) : void
+sellCargo(g: Game&, amount: int) : void
share file
«enumeration»
MoonWeather
+Clear = 0
+Flooded
+Eclisped
+Stormy
◄ uses
uses ►

Figure 2: Deadly Corp UML version 2

Notable design decisions and changes

- My final implementation uses the same system structure as the original version. I did however add one extra class for the paid moons in the bonus task which inherits from Moon.
- While the structure is the same I have added many functions and variables and extra details from the original. The core functions are essentially the same. Most of the added functions are getter or setter or functions difficult to foresee until they are needed.
- I did use vectors to store items and moons like planned however I made them vectors of shared pointers which pointed to the items or moons. I used shared pointers as the Game class need to also point to the moon currently being orbited or landed on. The same goes for items as the Game class points to items that are in the player's inventory.
- MoonManager does also manage CorpMoon but I couldn't get an arrow to it.
- I decided to make Item a struct to reflect its data structure nature. While it did end up with a lot of functions, they are all getter or setters.