Class and Package Diagram (Interact with Character)

Group 6

Blekinge Institute of Technology

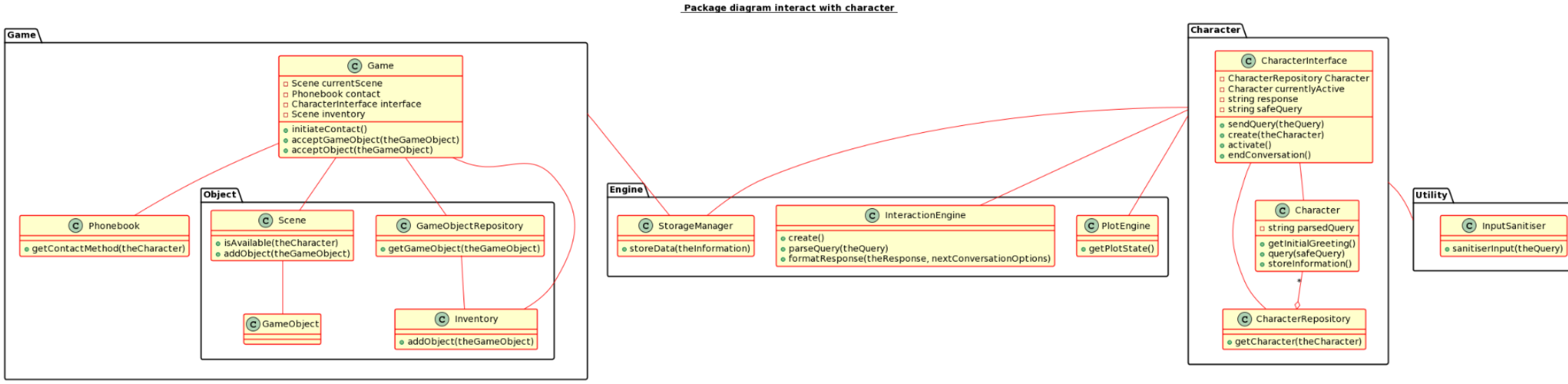
PA1472

17/02/2023

Oliver Bölin, Farhad Asadi, Samuel Täng, Michael Törnvall, Kim Budischewski

We did two different Class and packages

Diagram 1



Game
The game package is designed to encapsulate all the core game logic and functionality, making it easier to manage and maintain the codebase. It allows other packages to easily access and modify the game state.

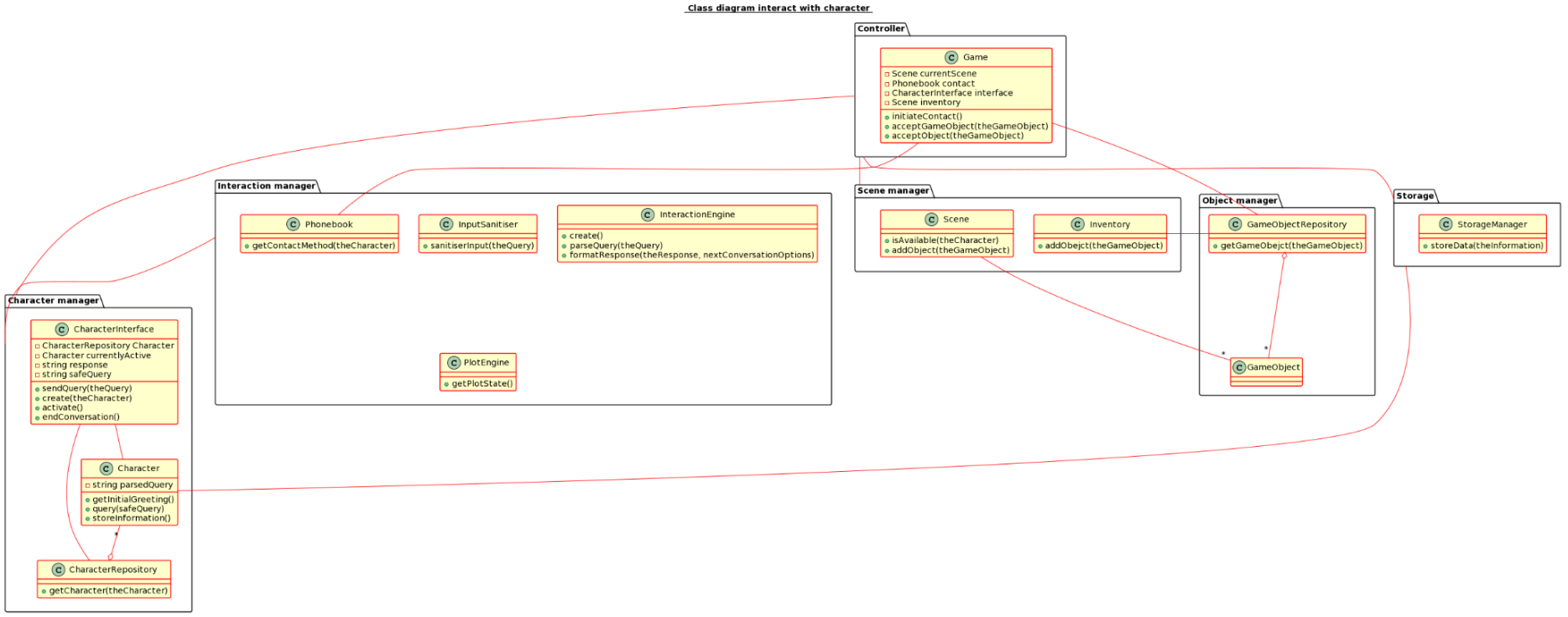
Object
The Object package is responsible for managing and manipulating objects that are used in the game, such as scenes, inventories, and game objects. The package provides a way to store, retrieve, and manipulate objects as needed, it also includes repositories for objects and inventory items.

Engine
The engine package provides the low-level functionality that the game relies on to function, such as managing user input, updating the game world, and triggering events based on the game's plot.

Character
The character package contains all character related classes. Providing a functionality for the character classes to get updated from the Engine package and using the Utility for input sanitation.

Utility
The utility package only contains InputSanitiser, providing utility for the character package.

Diagram 2



Interaction Manager
This package contains all interaction related classes. By putting these in the same package we allow interactions to modify or update other classes such as the character.

Object Manager
The package is responsible for managing the creation, storage, and retrieval of all game objects within the game. The package acts as an interface between the game and the individual game objects, providing a centralised repository for all objects and allowing the game to easily access and manipulate them.

Scene Manager
This package contains all scene related classes. This package is responsible for the current active scene and showing the currently available objects.

Controller
This package contains the Game class, which has the overall view of all of the other packages and classes and is the one who initiates conversations, and has control over trading objects between characters.

Storage

This package contains the StorageManager class, which would connect to a permanent storage, allowing the game to interface with storage. Also enabling modifying and usage of storage containers by other classes.

Character Manager
All classes related to characters, making for one package

UML code - Diagram 1

```
@startuml
title
<u> Package diagram interact with character </u>
end title
skinparam handwritten true
skinparam class {
padding 100
BackgroundColor Business
BorderColor red
BorderThickness 1.5
Shadowing true
ArrowColor red


SequenceLifeLineBorderColor red
}
skinparam package {
padding 50
BorderThickness 2
BackgroundColor #FFFFFF
BorderColor Black
}

package "Game" {
package "Object" {
class Scene {
+isAvailable(theCharacter)
+addObject(theGameObject)
}
class GameObjectRepository {
+getGameObject(theGameObject)
}
class Inventory {
+addObject(theGameObject)
}
Scene -- GameObject

}
class Game {
-Scene currentScene
-Phonebook contact
-CharacterInterface interface
-Scene inventory
+initiateContact()
+acceptGameObject(theGameObject)
```

```
+acceptObject(theGameObject)
}

class Phonebook {
+getContactMethod(theCharacter)
}



Game -- Scene
GameObjectRepository -- Inventory
Game -- GameObjectRepository
Game -- Inventory
Game -- Phonebook
}

package "Character" {
class CharacterInterface {
-CharacterRepository Character
-Character currentlyActive
-string response
-string safeQuery
+sendQuery(theQuery)
+create(theCharacter)
+activate()
+endConversation()
}

class CharacterRepository {
+getCharacter(theCharacter)
}

class Character {
-string parsedQuery
+getInitialGreeting()
+query(safeQuery)
+storeInformation()
}

CharacterInterface -- CharacterRepository
CharacterInterface -- Character
Character "*"--o CharacterRepository
}

package "Utility" {
class InputSanitiser {
+sanitiserInput(theQuery)
}
```

```
}

package "Engine" {
class InteractionEngine {
+create()
+parseQuery(theQuery)
+formatResponse(theResponse, nextConversationOptions)
}

class StorageManager {
+storeData(theInformation)
}

class PlotEngine {
+getPlotState()
}
}
Character -- Utility
Character -- InteractionEngine
Game -- StorageManager
StorageManager -- Character
Character -- PlotEngine
@enduml
```

UML code - Diagram 2

```
@startuml
title
<u> Class diagram interact with character </u>
end title

skinparam class {
padding 100
BackgroundColor Business
BorderColor red
BorderThickness 1.5
Shadowing true
ArrowColor red
SequenceLifeLineBorderColor red
}

package "Interaction manager"{
class InputSanitiser
class InteractionEngine
class PlotEngine
class Phonebook
}
package "Character manager"{
```

```
class CharacterInterface
class Character
class CharacterRepository
}
package "Object manager"{
class GameObjectRepository
class GameObject
}
package "Scene manager"{
class Scene
class Inventory
}
package Storage{
class StorageManager
}
package Controller{
class Game
}

"Interaction manager" -- "Character manager"

Game : -Scene currentScene
Game : -Phonebook contact
Game : -CharacterInterface interface
Game : -Scene inventory
Game : +initiateContact()
Game : +acceptGameObject(theGameObject)
Game : +acceptObject(theGameObject)

Scene : +isAvailable(theCharacter)
Scene : +addObject(theGameObject)

Game -- GameObjectRepository

GameObjectRepository : +getGameObejct(theGameObject)

GameObjectRepository -- Inventory

Inventory : +addObejct(theGameObject)

Scene --"*" GameObject
GameObjectRepository o--"*" GameObject

Game -- Phonebook

Phonebook : +getContactMethod(theCharacter)

Controller -- "Character manager"
```

```
Controller -- "Scene manager"

CharacterInterface : -CharacterRepository Character
CharacterInterface : -Character currentlyActive
CharacterInterface : -string response
CharacterInterface : -string safeQuery
CharacterInterface : +sendQuery(theQuery)
CharacterInterface : +create(theCharacter)
CharacterInterface : +activate()
CharacterInterface : +endConversation()

CharacterInterface -- CharacterRepository

CharacterRepository : +getCharacter(theCharacter)

CharacterInterface -- Character

Character : -string parsedQuery
Character : +getInitialGreeting()
Character : +query(safeQuery)
Character : +storeInformation()

Character "*"--o CharacterRepository

InputSanitiser : +sanitiserInput(theQuery)

InteractionEngine : +create()
InteractionEngine : +parseQuery(theQuery)
InteractionEngine : +formatResponse(theResponse, nextConversationOptions)

Controller -- Storage
Storage -- Character
StorageManager : +storeData(theInformation)

PlotEngine : +getPlotState()
@enduml
```