Property *(abstract)*

```
-propertyID: int
-propertyName: String
-propertyCost: double
-propertyRent: double
-propertyOwned = false: boolean
-propertyOwner: Player
+properties = new PropertyJsonReader(): PropertyJsonReader
```
```
+Property(): ctor
```

PropertyJsonReader

```
-lands = new HashMap<Integer, String[]>(): HashMap<Integer, String[]>
-railRoads = new HashMap<Integer, String[]>(): HashMap<Integer, String[]>
-companies = new HashMap<Integer, String[]>(): HashMap<Integer, String[]>
```
```
+PropertyJsonReader(): ctor
```

Land

```
-landProperties = new HashMap<Integer, Land>(): HashMap<Integer, Land>
```
```
+Land(): ctor
+createLandProperties(): HashMap<Integer, Land>
```

RailRoad

```
-railRoadProperties = new HashMap<Integer, RailRoad>(): HashMap<Integer, RailRoad>
```
```
+RailRoad(): ctor
+createRailRoadProperties(): HashMap<Integer, RailRoad>
```

Company

```
-companyProperties = new HashMap<Integer, Company>(): HashMap<Integer, Company>
```
```
+Company(): ctor
+createCompanyProperties(): HashMap<Integer, Company>
```

User *(abstract)*

```
-name: String
-money: double
```
```
+User(): ctor
+setMoney(): void
```

Card *(abstract)*

```
-players = Game.getPlayers(): HashMap<String, Player>
-cards = new ListJsonReader(): ListJsonReader
```

ListJsonReader

```
-chanceArrayList = new ArrayList<String>(): ArrayList<String>
-communityChestArrayList = new ArrayList<String>(): ArrayList<String>
```
```
+ListJsonReader(): ctor
```

Player

```
-position = 1: int
-ownedPropertyList = new ArrayList<Property>(): ArrayList<Property>
-ownedRailRoadNumber: int
-players = new HashMap<String, Player>(): HashMap<String, Player>
-jailCount: int
-inJail = false: boolean
```
```
+Player(): ctor
+createPlayers(): HashMap<String, Player>
+changePosition(): void
+addOwnedRailRoadNumber(): void
+addJailCount(): void
+addParkingCount(): void
```

Banker

```
+Banker(): ctor
```

CommunityChest

```
-communityChestIndex = 0: int
```
```
+drawCommunityChestCard(): void
+addCommunityChestIndex(): void
```

Chance

```
-chanceIndex = 0: int
```
```
+drawChanceCard(): void
```

Game

```
-commands = new ArrayList<String[]>(): ArrayList<String[]>
-landIds = new ArrayList<Integer>(): ArrayList<Integer>
-railRoadIds = new ArrayList<Integer>(): ArrayList<Integer>
-companyIds = new ArrayList<Integer>(): ArrayList<Integer>
-communityChestIds = new ArrayList<Integer>(): ArrayList<Integer>
-chanceIds = new ArrayList<Integer>(): ArrayList<Integer>
-maxRounds: int
-roundsPlayed: int
-bankruptcy = false: boolean
-banker = new Banker(): Banker
-players = Player.createPlayers(): HashMap<String, Player>
```
```
+readCommandFile(): void
+playTurns(): void
+show(): void
```

Method parameters, getters and setters are removed from the UML diagram.

In this assignment, I read the .json files in JsonReader classes and used them in Property and Card classes so the attributes and methods are initialized. Property and Card classes are abstract because they can't be instantiated. I created Land, RailRoad and Company classes to create objects of all properties. I also created CommunityChest and Chance classes to behave different for every card. I created an abstract User class and extended Player and Banker classes from it. As you can see, inheritance is used for property, cards and users. Both players are instantiated in Player class. Banker is instantiated in Banker class. All actions are done in Game class. Polymorphism is used in Game class. It mostly consists of static methods and variables. This class reads the command.txt file and does actions according to the dice. It then makes the player play their turn. When someone bankrupts, their money reaches 0 or command file ends, game ends.