1. Gereksinimler

* Programlama Dili : JAVA
* Kullanılan API’ler : Slick, LWJGL, JDBC
* DBMS : MySQL

# Kullanılan API’ler

## LWJGL

Lightweight Java Game Library, bir takım native library ler sayesinde Java’dan OpenGL kütüphanesine ulaşılabilmesini sağlayan bir API’dir. Bu API’lerin sadece programda kullandığım kısımlarını açıkça anlatacağım. Programda bu API’ye direk erişimimiz yok.

## Slick

Slick, LWJGL’in sağladıklarını kullanarak yazılmış tam bir Java 2D oyun kütüphanesi. Bizim geniş olarak kullandığımız özelliği StateBasedGame sınıfı. Main sınıfımız olan RPG372.java bu sınıfı extend etmektedir. Ayrıca aynı sınıfın main methodunda AppContainer sınıfının bir nesnesi yaratılmıştır. Bu sınıf oyunun render edileceği pencereyi açmaktadır. StateBasedGame in bize sağladığı yarar, BasicGameState sınıfını extend eden sınıfların nesnelerini bu sınıfa ekleyip aralarında istediğimiz gibi geçiş yapmamızı sağlayan bir Finite State Machine framework ü sağlaması. Bu BasicGameState ler ise çok önemli 3 metot içermektedir. Bunlar init(), render() ve update() dir. Bunlar basit bir oyun döngüsünün aşamalarıdır. Render metodunda o state aktifken ekrana çizme işlemleri yapılır, update metodunda ise input alınıp işlenir, init ise sadece bir kere başta çağırılır, genelde resource loading işlemlerinin yapıldığı kısımdır. Statelerin ayrıntılarına daha sonraki kısımlarda girilecektir. Burada sadece Slick in ne işe yaradığı anlatılmaktadır.

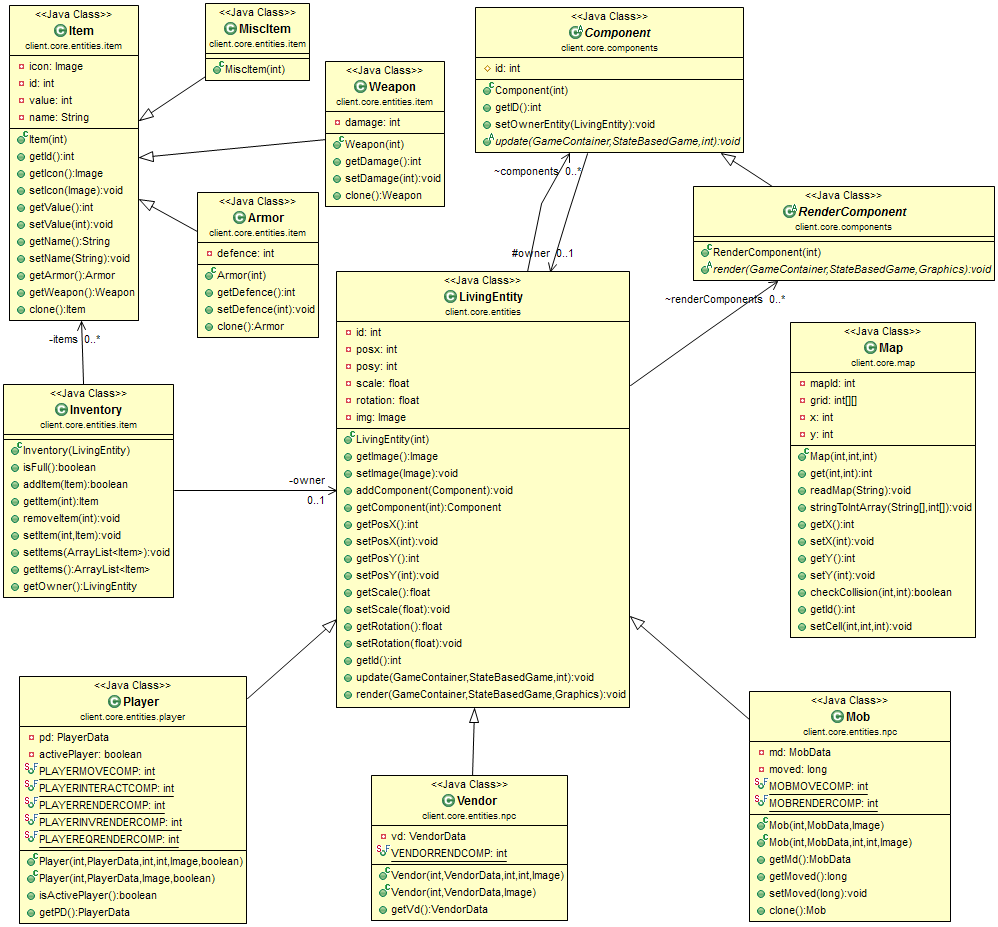
## JDBC

Java Database Connectivity, Java’dan MySQL e bağlanmamız için bize ara yüz sağlayan kütüphanedir.

1. RPG372

Oyunun genel yapısının anlaşılması için UML ve bir takım açıklamalar olacaktır. Daha sonra ilk önce bahsettiğimiz game stateleri ve aralarındaki geçişler, sonra da her state in kendi içindeki döngüsü anlatılacaktır.

# UML Diyagramı

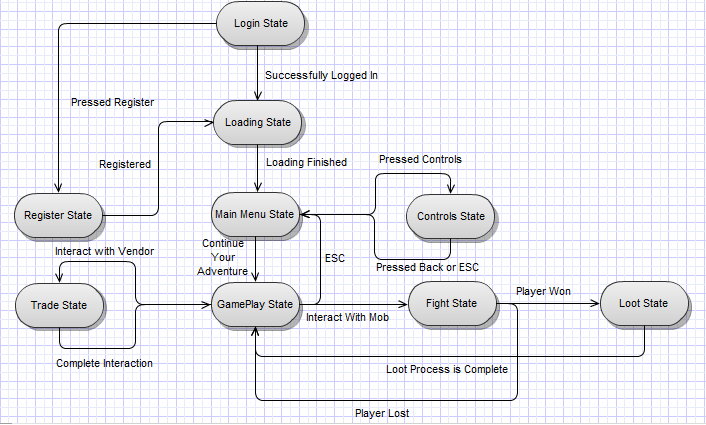


UML de dikkat edilmesi gereken en önemli nokta Entity-Component yapısıdır. Bu yapı bir örnekle daha iyi açıklanabilir. Player LivingEntity’i extend eden bir sınıftır. PlayerMoveComponent Player a addComponent metodu ile eklenebilir. Bu componentin update metodu klavye inputunu okuyup basılan yön tuşlarına göre Player sınıfının setPostX() ve setPosY() metotlarını çağırmaktadır. Böylelikle herhangi bir yerde Player sınıfının update metodunu çağırmak, kendi componentlerinin update metodunu çağırdığı için Player ın hareket etmesini sağlayacaktır. RenderComponent ise component in özel bir alt sınıfı olup update metodunun yanında render metoduna sahiptir. Player sınıfının render metodu çağırıldığında Playerın pozisyon bilgisine göre ekrana çizdirilmesini sağlayan PlayerRenderComponent sınıfı mevcuttur.

LivingEntity i extend eden üç sınıfında kendi Data sınıfları mevcuttur. Bu PlayerData, MobData ve VendorData sınıfları içerilerinde Inventory sınıfını extend eden PlayerInventory, MobInventory ve VendorInventory i bulundurur. Bu sınıflar diyagramı karmaşıklaştırdığından dolayı gösterilmemektedir.

Ayrıca diyagramda game stateleri bulunmamaktadır. Bu sınıflar sonradan açıklanacaktır.

# Game States ve Diyagramları



Stateler arasındaki geçişler diyagramda gösterilmiştir. Şimdi bu statelerde tek tek her döngüde neler olduğuna bakalım.

## Login State

## Register State

## Loading State

## Main Menu State

## Controls State

## GamePlay State

## Trade State

## Fight State

## Loot State