ATEŞ VE SU OYUNU RAPORU

ADİL KAYGIN
221401003

Öncelikle kodun çalışma prensibini anlatacağım sonra hangi dizayn tercihlerini neden yaptığımı anlatacağım. Oyunumuzu çalıştırmak için Main.java dosyasını compile etmemiz ve çalıştırmamız gerekiyor. Main sınıfında oyunu kayıtlı dosyadan mı yoksa yeni oyundan mı başlatacağımı belirliyorum. Yeni oyundan başlayacağımız durumda öncelikle JFrame i extend eden GameFrame sınıfından bir obje oluşturuyoruz. GameFrame sınıfının constructor ı JPanel sınıfını extend eden bir GamePanel objesi oluşturuyor. GamePanel objesi gerekli diğer bütün Componentları oluşturuyor ve 2 tane inner sınıfa sahip bunlar, KeyAdapter sınıfını extend eden MyKeyAdapter ve Runnable interface'ini implement eden GameLoop sınıfları. Bu sınıflar adlarından anlaşılacağı üzere klavye girdisi almak ve oyun döngüsünü sağlamakla görevliler.

GamePanel'in kullandığı çoğu Component'ın atası JLabel yani Player, Monster, Door, Dot, PowerUp, Block sınıfları JLabel'ı extend ediyor Trap sınıfı Block sınıfını extend ediyor.

Her sınıfı tek tek incelemeden önce kullandığım interfacelerden bahsedeceğim. Collidable ve Gravitational adında iki tane interface var Collidable interface i collision detection için üretilmiştir. Collidable olan Player,Monster, Door, Dot, PowerUp, Block sınıfları Collidable interface'ini implement ediyor. Trap sınıfı Block sınıfını extend ediyor dolayısıyla Collidable ı implement ediyor. Gravitational interface'i ise yerçekiminden etkilenen sınıflar için kodlandı. Yani Player ve Monster sınıfları Gravitational interface'ini implement ediyor.

Şimdi dizayn tercihlerimden ve neden tercih ettiğimden bahsedeceğim.

MyKeyAdapter ve GameLoop sınıflarını inner sınıf olarak kullandım ve ilgili methodları inner sınıfların içine yazdım. Böylece her method olması gerektiği yerde ve kullanması ve anlaması çok daha kolay.

Çarpışma denetlemesi kolay ve her obje için standart olsun diye Collidable interface'ini oluşturdum.

Mermiler için Player sınıfına Bullet inner sınıf ekledim, JLabel sınıfını extend ettim ve Collidable interface'ini implement ettim. Bu sayede mermi ile ilgili bütün işlemleri Player içinde hallettim.

Olabildiğince encapsulation uygulamaya çalıştım. Yani işlemlerin çoğu sınıf içinde gerçekleşiyor ve bazı metotlar ile sınıflar arasındaki ilişkiyi sağlıyorum.

Gravitational interface'i ile yerçekiminin uygulanmasını da standart hale getirdim.

Bakımı ve güncellemesi kolay olsun diye kullanılan çoğu parametreyi değişken olarak kullandım.

Zorlandığım kısımlara gelirsek en çok çarpışma denetlemesi ve yer çekiminde zorlandım.

Çarpışmalar için çözümüm çarpışma maskesi kullanarak Rectangle sınıfının intersect metodunu çağırmak oldu.

Yerçekiminde zorlanma sebebim akıcı görünmesi için ivmeli yerçekimi kullanmam ve dikey hızı GameLoop içinde güncellememden kaynaklı Çarpışmaları kontrol etmekte sorun yaşamam oldu. Bunun için limit hız getirdim.

Ayrıca oyun içinde sürekli elemanları ArrayList lerden alıp güncellemelerini yaptığım için ConcurrentModificationException hatası aldım bunu düzeltmek için Local olarak parametreleri kopyalayarak işlemleri yaptım. Senkronizasyon kullanmak işe yaramadı nedenini bulamadım.

Son olarak GamePanel'den işi biten Componentları çıkarırken SwingUtilities içerisindeki invokeAndWait sınıfını kullandım.

WildCard kısmında canavarları öldürünce PowerUp düşürmelerini tasarladım bu PowerUp'lar Pembe , CamGöbeği, Kırmızı, Mavi ve Magenta renklerinde. Pembe PowerUp Alındığında o anki skor ile oyunu kazanmanızı sağlar. CamGöbeği rengi PacMan dan esinlenilerek implement edildi aldığınız zaman süresi dolana kadar canavarlara çarptığınızda siz değil canavarlar ölür. Kırmızı renk aktif olduğu sürece her renk karakter ile kırmızı tuzaklardan geçebilirsiniz. Mavi renk aynı şekilde mavi tuzaklara görünmez yapar. Magenta rengi de aktif olduğu sürece Karakterin her renkte Magenta oyuncu kadar zıplamasını sağlar. Ayrıca Skor barının yanına hangi PowerUp için ne kadar süremiz kaldığını gösteren bir Etiket koydum. Oyun biraz zor olduğu için PowerUpları birikimli yaptım ama çok kolay bir şekilde birikimsiz yapılabilir.

Canavar her öldüğünde bulunduğu konuma bir PowerUp bırakır PowerUp ın tipi constructor içinde rastgele belirlenir. GamePanel içinde powerups ArrayListi tutulur. Player ile PowerUp'ların çarpışması kontrol edilir ve çarpıştıklarında Player'ın setPowerUp methodu çağırılır Player içinde gerekli modifikasyonlar yapılır. PowerUp sayaçları için Timer sınıfından yararlanılmıştır.

save kısmında threadler ve Timerler düzgün kaydedilemediği için save yüklenirken yeniden initialize edilir.