BMT-209 Güz Dönemi MiKurī Projesi

Rapor Yazarları ve Öğrenci Numaraları  
Mohammad Murad Chamaa: 23181616063  
Juan Diego Ron Molina: 22181616070

# Özet

Java dilinde kırmızı ve mavi takımın olduğu, canavarlarla saldırılar düzenleyerek kazanmaya çalıştığımız bir oyun hazırlandı.

# İçindekiler

**Giriş Bölümü:** 1-2 sf.

**Oyun Projesi Bölümü:** 2-8 sf.

**Sonuç Bölümü:**

**Ekler Bölümü:**

**Kaynakça Bölümü:**

# Giriş

Java günümüzde de hala birçok sektörde aktif olarak kullanılan köklü bir programlama dilidir. Kod yazılması diğer dillere kıyasla zor olsa da dilde barındırdığı birçok özelliğin gücü ve yeni yeni kullanılan programlama dillerine göre programların daha hızlı çalışması günümüze kadar hala popüler olarak kullanılması sağlanmıştır.

Java dilinin yaygın olarak kullanıldığı alanlar mobil uygulamalar, web tabanlı uygulamalar, girişimci uygulamalar gibi birçok alanda kullanılmaktadır. Örneğin Spotify, Netflix, Amazon, Barclays, MATLAB ve daha birçok uygulama Java dilinden yararlanılarak yazılmıştır. Fakat oyun sektöründe genellikle tercih edilmeyen bir dildir (Minecraft hariç). Çünkü oyun sektöründe geliştirilen oyunlar artık herhangi bir oyun motoru üzerinden geliştirilerek yapılıyor.

Unturned, The Long Dark, Traffic Racer oyunları Unity Engine; Valorant, Callisto Protocol, Sea of Thieves oyunları Unreal Engine; Brotato, Swords and Sandals Immortals oyunları Godot Engine ile yapılmıştır.

Bu oyun motorları oyun yapımcıları için her ne kadar kolaylık sağlasa da sağladığı rahatlık büyük ve karmaşık oyunlar için geçerlidir. Bu yüzden genellikle basit bir oyunu bir derleyici yardımıyla kodlamak, oyun motoru üzerinde yazmaktan daha basit ve rahattır. Fakat her bir programlama dili üzerinden her tür oyunun yapılması tavsiye edilmez. Örneğin C üzerinden genellikle oyun yapılması tavsiye edilmez çünkü C’nin GUI kısmı (grafiksel kullanıcı arayüzü) için yeterince destek yoktur.

Bir de oyunlarda platform, oyuncu, düşman, silah gibi birçok nesne olduğundan kullanılacak programlama dilinin nesne yönelimli programlamayı da desteklemesi şarttır.

Bu yüzden oyunumuz için hem basit bir GUI oluşturabilecek hem de nesne yönelimli programlama özelliğini barındıran bir programlama dili seçilmelidir. Java programlama dili bu şartları sağladığı için Java dili üzerinden oyun geliştirmeye karar verdik.

Bunun için Minecraft oyununda da kullanılana JavaFX kütüphanesini de kullanacağız. JavaFX sadece GUI kodlamak için değil, oyunlarda bulunması gereken birçok özelliği basit bir biçimde eklememizi sağlar. Bir de GUI’ı kodlamak yerine GUI işlemini FXML dosyası ile GUI üzerinden yapmamızı sağlayan SceneBuilder uygulaması sayesinde GUI tasarımı basit bir biçimde kısa bir zamanda yapılabilir.

Bir de oyun içerisinde müzik ve resimlerin de kullanılması gerekmektedir.

# Oyun Projesi

## Oyunu Belirleme

Java dili her ne kadar eskiden oyun sektöründe popüler olsa da günümüzde oyun geliştirmek için kullanılmıyor. Bu yüzden Java’daki en önemli özelliklerden olan OOP’yi (nesne yönelimli programlamaya) avantajımıza kullanacağımız bir biçimde bir oyun türü seçilmeli. Bu yüzden Pokemon tarzında bir oyun yapmaya karar verdik.

Kırmızı ve mavi taraf olmak üzere taraflarda bulunan mini canavarların birbiriyle savaşarak kazanmak oyunun amacıdır. İlk hangi tarafın canavarları yok edilirse o taraf kaybeder. Bu yüzden canavarlarınızı stratejik bir biçimde kullanarak düşmanı yenmeniz gerekmektedir.

Oyunun adını MiKurī koymaya karar verdik. Çünkü açılımı Mini Kurīchā (mini canavar) olduğundandır.

## Oyunun Temel Mimarisi

UML

## Oyunun Kodları

### MikuriApplication Sınıfı

İlk olarak oyunun açılışını sağlayacak bir MikuriApplication sınıfı tanımlandı. Bu sınıf ana metodunu çalıştırarak start metodunu çalıştırır ve oyunun ilk ekranını açar. Sonra da FXML yüklemek için bir nesne oluşturup o nesneyi bölmeye atadı. Bu sayede FXML dosyasını root nesnesine yüklendi. Burada start metodu override edilmiş çünkü start metodu Application sınıfından mirastır.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Sonrasında yüklediğimiz dosyayla sahnemiz açılarak ayarları yapıldı ve sahne ekrana yansıtıldı. Bundan sonra FXML dosyasına atanan StartController sınıfı açılır.

### MikuriMenu Arayüzü

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu arayüzü implements eden (kendi sınıfına uygulayan) bütün sınıflar buradaki bütün metotları içerisinde barındırması zorunludur.

### TransitionController Sınıfı

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu sınıf soyut olarak tanımlandı çünkü bu sınıftan nesne oluşturulması istenmiyor. Bu sınıfın içerisindeki menuyeDegisme ve automaticIncrease metotları soyut olarak tanımlandığından bu sınıfı miras alan tüm alt sınıfların bu metotlar içermesi zorunludur.

### StartController Sınıfı

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Burada StartController sınıfı TransitionController sınıfını miras alır ve Initializable arayüzünü kendi sınıfına uygular.

initialize metodunda StartController sınıfının ilk çalıştırması gereken ögeleri çalıştırır. Bu metot çalıştırıldığında automaticIncrease metodunun çalışıp çalışmadığını dener. Eğer hata verirse hata yakalanır ve o hatayı RunTimeException ile beklenmeyen bir hata olduğunu gösterir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

automaticIncrease metodunda InterruptedException sınıfındaki istisnaları atıp yükleme ekranının çalıştırılmasını sağlar. Metot FXML dosyasını içerdiğinden @FXML ile belirtildi. Bu metot TransitionController sınıfından geldiği için bu metodu override eder. Yükleme ekranının oluşmasını sağlar, burada yükleme barını zaman içinde doldurarak oyunu yükler. Bar dolduğunda ise menuyeDegisme metodunu dener.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

menuyeDegisme metodu TransitionController sınıfından override edilmiş olup yükleme ekranından ana menü ekranına geçişi sağlar. Sonrasında progressBar ögesinin bulunduğu sahneyi window nesnesine atayarak sonradan yeni sahneye geçiş yapmamızı sağlar.

### MikuriController Sınıfı

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu sınıf hem Initializable hem de MikuriMenu arayüzünü kendi sınıfına uygular. menuyeDegisme metodundaki FXML dosyası bu sınıfa bağlı olduğu için sınıf çalışır. Buradaki nesnelerin hepsi private olarak tanımlanmış ve bir tanesi hariç hepsi FXML dosyası ile bağlantılıdır.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu metot çalıştığında ilk önce ses dosyasının uygulamada çalışıp çalışmadığını dener. İlk başta ses dosyasının var olup olmadığı kontrol edilir, sonra ses dosyası ses nesnesine tanımlanıp sesi oynatır. Ses oynatıldıktan sonra onSlider metodu çalışır.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

onSlider metodu volumeSlider kaydırıcısını ayarladıktan sonra ses ile senkronize edilir. Override edilmiş çünkü MikuriMenu sınıfından uygulanmış bir metottur.

hide metodu gözükmemesi gereken ögeleri gizleyip show metodu gözükmesi gereken ögeleri gösterir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu metot çalıştırıldığında butonları gizleyerek oyuna geçiş metodu olan gameDegisme metodunu çalıştırır. Override edilmiş çünkü MikuriMenu sınıfından uygulanmış bir metottur.

metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

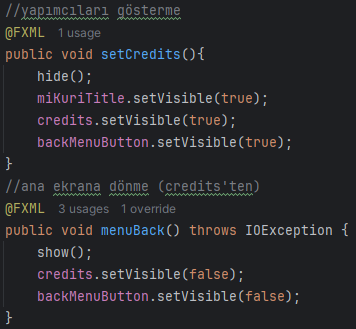
Açıklama otomatik olarak oluşturuldu

Bu metot çalıştırıldığında window nesnesini newGameButton düğmesinin bulunduğu sahneden menuToGame.fxml dosyasındaki sahneye çeker.

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu metot çıkış butonuna basıldığında çalışır. Override edilmiş çünkü MikuriMenu sınıfından uygulanmış bir metottur. İlk başta (ögeleri gizleyip) “Emin misiniz?” mesajını göstererek evet ve hayır adında iki seçenek sunar. Evet butonuna basılırsa “Çıkış yapılıyor…” yazısını yazıp 1 saniye sonra oyundan çıkar. Hayır butonuna basılırsa (ögeleri gösterip) ana menüye döner.



setCredits metodu oyunu yapan kişilerin adlarını ekrana yansıtır. Bunu yaparken diğer ögeleri gizler (miKuriTitle hariç).

menuBack metodu ise credits butonuna basıldıktan sonra ana menüye dönmek için koyulan backMenuButton düğmesine basıldığında çalışır. Bunu yaparken credits ve backMenuButton ögelerini gizleyip ana menü ögelerini gösterir.

### MenuToBarController Sınıfı

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu sınıf StartController sınıfını miras alır ve sadece menuyeDegisme metodunu overload eder. StartController sınıfındaki bütün işleri kendisi de yapar fakat menuyeDegisme metodunda yükleme ekranından oyun içi ekrana geçiş yapacağı için o aradaki fark yüzünden overload (tekrar yükleme) yapılır.

### InGameController Sınıfı

metin, ekran görüntüsü, yazılım içeren bir resim

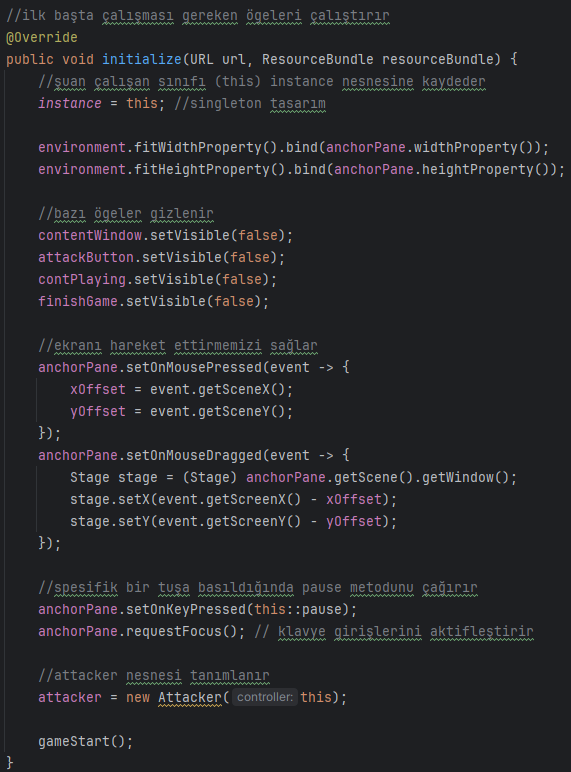
Açıklama otomatik olarak oluşturuldu

Bu sınıf MikuriController sınıfını miras alırken Initializable arayüzünü kendine uygular. Görüleceği üzere birden fazla global değişken bu sınıfta tanımlanmıştır.

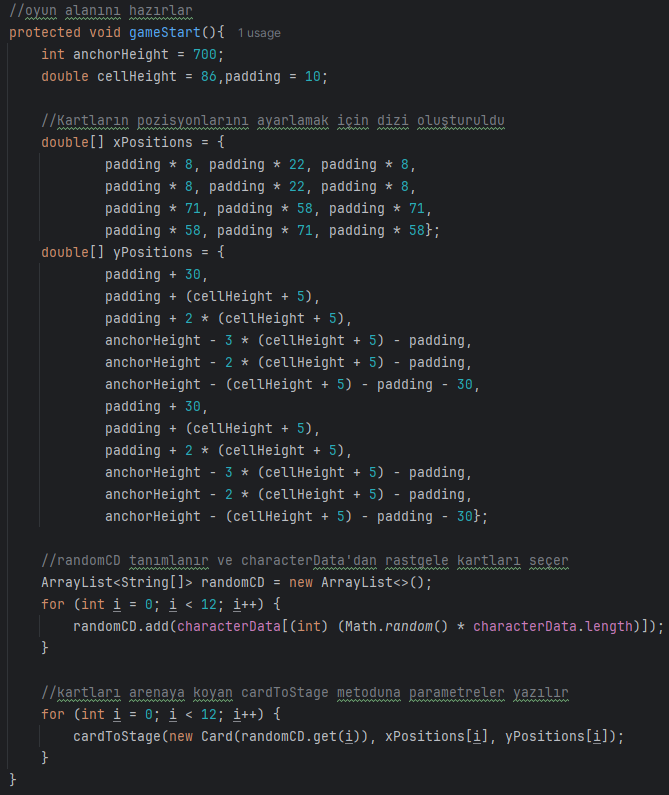
metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Burada tanımlanan global değişkenler ilki kart (canavar) bilgilerini characterData dizisine sırasıyla atar. Burada characterImages için HashMap kullanılmasının sebebi ImageView ile Card nesnesi arasında bağlantı kurulmasını sağlayarak ileride kart resimlerini ekrana yazdırılması sağlanabilecektir. Fotoğrafları listelere tanımlamak için ArrayList kullanıldı. Bu değişkenler private tanımlandığından ve bu değişkenlerden bazılarını alıp yazdırmak istediğimizden bu getter metotlarını oluşturduk.



Bu metot ilk başta çalışması gereken ögeleri çalıştırır. İlk başta instance nesnesine şu anki sınıf atanır. Sonra ekran boyutları ayarlanıp ekranın hareketi sağlanır. Sonrasında pause metodunun bir klavye tuşuyla aktifleştirilmesi sağlanır. Sonda atak yapmamızı sağlayacak nesneyi tanımlayıp gameStart metodu çalıştırılır.



Bu metot oyunda belirmesi gereken canavarları (kartları) oyunda belirmesini sağlar. Tanımlanması gereken değişkenler tanımlanır, dizilere kartların arenada uygun pozisyonda durması için uygun değerler tanımlanır, rastgele kartlar seçilip cardToStage metodu yardımıyla kartlar arenaya dizilir.

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

İlk başta resimleri character nesnesine atarak onları takımlara göre kaydeder. Eğer kart üst tarafta kalıyorsa kırmızı takımda altta kalıyorsa mavi takımda yer almaktadır. Mavi takımı siz, kırmızı takımı ise bilgisayar (CPU) oynamaktadır; bu yüzden sadece kartların mavi kısmında bulunanlar ile atak yapabilirsiniz. En sonda kartlar ile character arasındaki bağlantıyı sağlayıp kartları sahneye ekletir.

### Attacker Sınıfı

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

### Card Sınıfı

### CardFunctionality Sınıfı

### SaveInFile Sınıfı

## Oyunda Kullanılan Özelliklerin Özeti

### Sınıflar ve Nesne Yapısı

Kalıtım (Inheritance)

Çok Biçimlilik (Polymorphism)

Kapsülleme (Encapsulation)

Soyutlama (Abstraction)

Hata Yönetimi (Exception Handling)

Tarih ve Zaman İşlemleri

Kullanıcı Arayüzü

Java Temel Yapıları

Ek Gereksinimler

# Sonuç

# Ekler

**Oyunun Ekran Görüntüleri:**

Ana Menü:

dağ, bulut, metin, gökyüzü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Credits:

bulut, metin, gökyüzü, dağ içeren bir resim

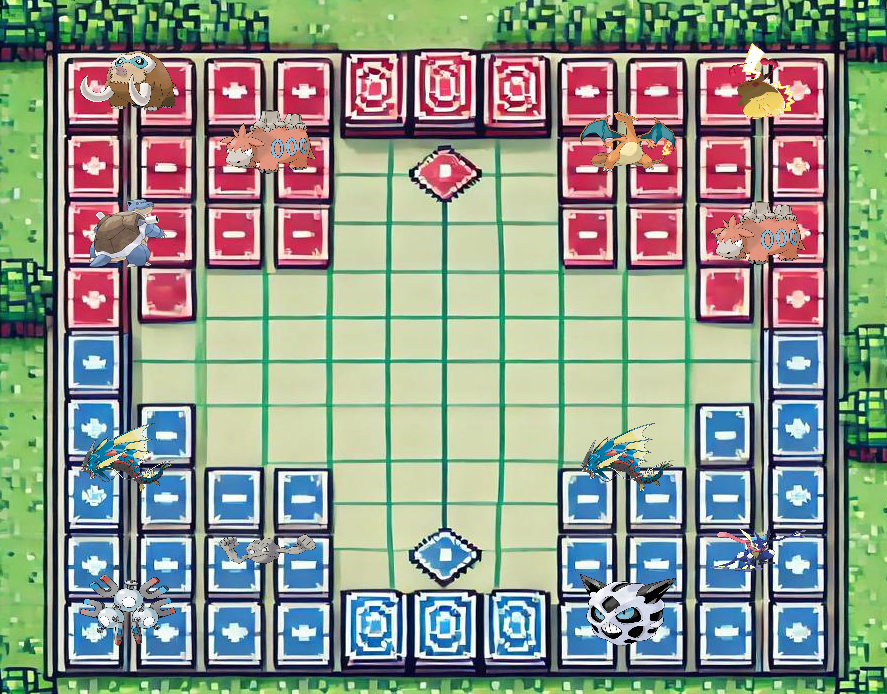
Açıklama otomatik olarak oluşturuldu

Exit Game:

bulut, dağ, gökyüzü, resim içeren bir resim

Açıklama otomatik olarak oluşturuldu

Oyun İçi Ekranı:

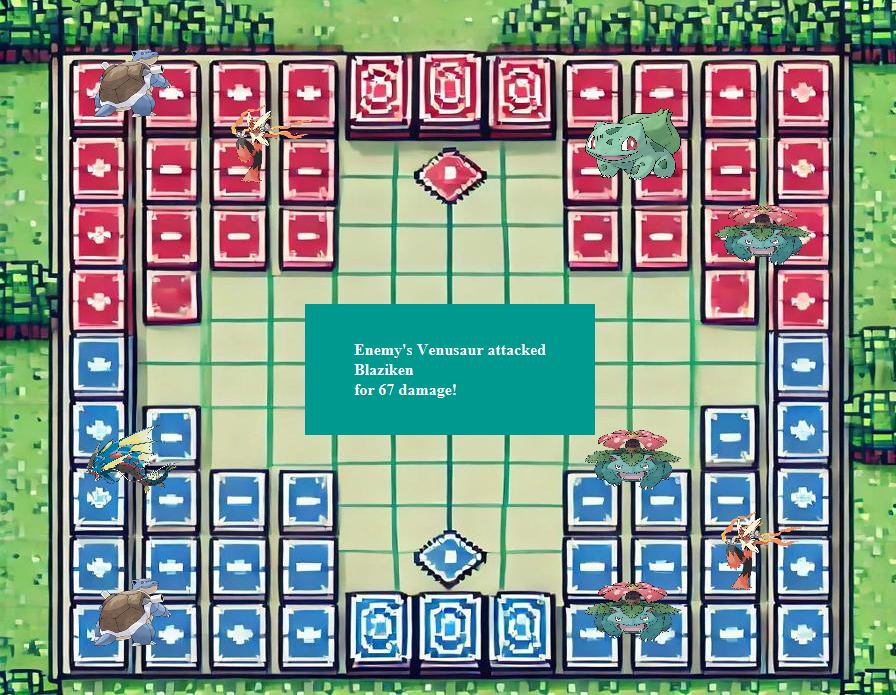


Duraklama Menüsü:

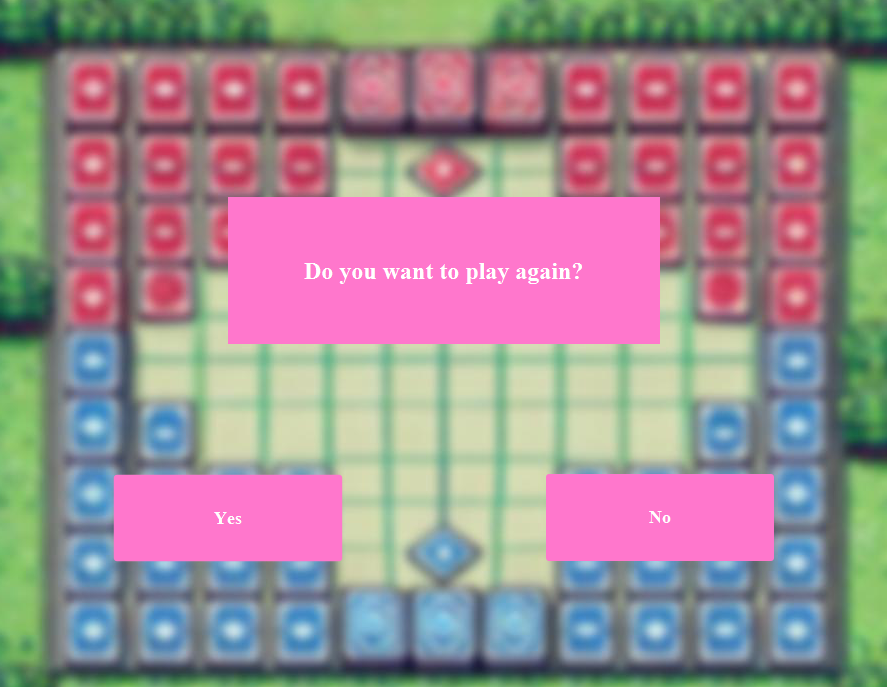
ekran görüntüsü, dikdörtgen, kare, renklilik içeren bir resim

Açıklama otomatik olarak oluşturuldu

Oyunun İlerleyişi:



Oyun Bitişi Sonrası Menü:



# Kaynakça

|  |  |
| --- | --- |
| [1] | Wikipedia contributors, “Java (programming language),” Wikipedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Java\_(programming\_language). [Accessed 22 12 2024]. |
| [2] | contributors, Spectrum, “Top Programming Languages 2024,” IEEE Spectrum, 2024. [Online]. Available: https://spectrum.ieee.org/top-programming-languages-2024. [Accessed 22 12 2024]. |