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

I. 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.

II. Oyun Projesi

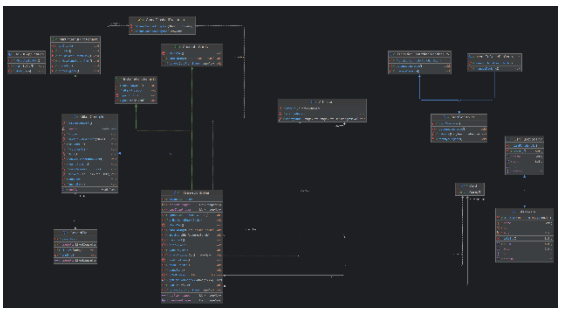
A. Oyunu Belirleme

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.

B. Oyunun Temel Mimarisi



Oyunun temel mimarisi bu resimde gösterilmiştir. Toplamda 9 sınıf, 2 soyut sınıf, 3 arayüz ve 1 istisna kullanılmıştır.

C. Oyunun Kod Kısmı

1. MikuriApplication Sınıfı

```
package com.pokemonlikegame.mikuri_project;

import ...

public class MikuriApplication extends Application {

    //uygulamayı başlatma
    public static void main(String[] args) { launch(); }

    @Override
    public void start(Stage primaryStage) throws IOException {
        //FXML dosyasını root'a atama
        FXMLLoader fxmlLoader = new FXMLLoader
                (MikuriApplication.class.getResource( name: "start.fxml"));
        AnchorPane root = fxmlLoader.load();

        //yeni bir sahne nesnesi oluşturup sahneyi gösterme
        Scene scene = new Scene(root);
        primaryStage.initStyle(StageStyle.UNDECORATED);
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

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.

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

2. MikuriMenuFunctionality Arayüzü

```
package Functionality;

import java.io.IOException;

public interface MikuriMenuFunctionality {  2 usages  2 implementations
    void onSlider();  1 usage  1 implementation

    void onNewGameButtonClick();  1 usage  1 implementation

    void onExitGameButtonClick();  2 usages  2 implementations

    void menuBack() throws IOException;  3 usages  2 implementations

    void setCredits();  1 usage  1 implementation

    void gameDegisme() throws IOException;  2 usages  2 implementations
}
```

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

3. TransitionControllerFunctionality Sınıfı

```
package Functionality;

import java.io.IOException;

public abstract class TransitionControllerFunctionality {  2 usages  2 inheritors
    //alt sınıflara bu metotların uygulanması
    public abstract void menuyeDegisme() throws IOException;  1 usage  2 implementations

    public abstract void automaticIncrease() throws InterruptedException;  1 usage  1 implementation
}
```

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.

4. StartController Sınıfı

```
package com.pokemonlikegame.mikuri_project;

import ...

public class StartController extends TransitionControllerFunctionality implements Initializable {

    @FXML
    private Label count;
    @FXML
    private ProgressBar progressBar;
    private double progress;  4 usages

    //ilk çalıştırılacak ögeleri çalıştırır
    @Override
    public void initialize(URL arg0, ResourceBundle arg1){
        try {
            automaticIncrease();
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }

    //yükleme ekranının çalıştırmasını sağlar
    @FXML  1 usage
    @Override
    public void automaticIncrease() throws InterruptedException{
        progressBar.setStyle("-fx-accent:  #ff77cc");
        Timeline timeline = new Timeline(new KeyFrame(
                Duration.millis( v: 4),
                _ -> {
                    if (progress < 1.0){
                        progress += 0.002;
                        count.setText((int) (progress * 100) + "%");
                        progressBar.setProgress(progress);
                    } else {
                        //bar dolduğunda çalışır
                        try {
                            menuyeDegisme();
                            ///((Timeline) event.getSource()).stop();
                        } catch (IOException e) {
                            throw new RuntimeException(e);
                        }
                    }
                }));
        timeline.setCycleCount(Timeline.INDEFINITE);
        timeline.play();
    }

    //yükleme ekranından ana menüye geçişini sağlar
    @FXML  1 usage  1 override
    @Override
    public void menuyeDegisme() throws IOException {
        if(progressBar.getScene() != null){
            //yükleme ekranını window nesnesine atar
            Stage window = (Stage) progressBar.getScene().getWindow();
            if(window != null){
                //window nesnesindeki sahneyi yeni sahneye atama yapılır
                Parent root = FXMLLoader.load(Objects.requireNonNull
                        (getClass().getResource( name: "mikuri-StaLoaScreen.fxml")));
                window.setScene(new Scene(root,  v: 600,  v1: 450));

                //ayarlama yapma
                Rectangle2D screenLayout = Screen.getPrimary().getVisualBounds();
                window.setX((screenLayout.getWidth() - 600) / 2);
                window.setY((screenLayout.getHeight() - 450) / 2);
            }
```

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.

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.

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.

5. MikuriController Sınıfı

```java
package com.pokemonlikegame.mikuri_project;

import ...

public class MikuriController implements Initializable, MikuriMenuFunctionality {
    @FXML
    private Label txt;
    @FXML
    private TextArea victoriesText;
    @FXML
    private Button yesButton;
    @FXML
    private Button noButton;
    @FXML
    private Button newGameButton;
    @FXML
    private Button exitGameButton;
    @FXML
    private Button backMenuButton;
    @FXML
    private Button creditsButton;
    @FXML
    private Button victoriesButton;
    @FXML
    private Slider volumeSlider;
    @FXML
    private ImageView miKuriTitle;
    @FXML
    private ImageView soundIcon;
    @FXML
    private AnchorPane credits;
    @FXML
    private AnchorPane victoryWindow;

    private static MediaPlayer media; 9 usages

    public static MediaPlayer getMedia(){ return media; } 1 usage
```

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.

```java
@Override 1 override
public void initialize(URL arg0, ResourceBundle arg1){
    credits.setVisible(false);
    backMenuButton.setVisible(false);
    victoryWindow.setVisible(false);

    //ses dosyasının çalışmasını dener
    try {
        URL resource = getClass().getResource( name: "/music/startMusic.mp3");
        assert resource != null;
        Media ses = new Media(resource.toString());
        media = new MediaPlayer(ses);
        media.setCycleCount(MediaPlayer.INDEFINITE);
        media.setAutoPlay(true);
    } catch (Exception e) {
        System.err.println("Error initializing media: " + e.getMessage());
    }

    if (media != null) {
        onSlider();
    }

    SaveInFile.setController(this);
    SaveInFile.printInfo();
}
```

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. Sonrasında onSlider metodunu çalıştırılır, en sonda kazananların kaydedildiği dosya yüklenir.

```
//ses kaydırıcısını ses ile senkronize eder
@FXML  2 usages
@Override
public void onSlider() {
    volumeSlider.setMin(0);
    volumeSlider.setMax(1);
    volumeSlider.setValue(media.getVolume());
    volumeSlider.valueProperty().addListener(( _, _, newValue) -> {
        media.setVolume(newValue.doubleValue());
    });
}
//ögeleri gizler
public void hide(){  3 usages
    newGameButton.setVisible(false);
    exitGameButton.setVisible(false);
    creditsButton.setVisible(false);
    volumeSlider.setVisible(false);
    miKuriTitle.setVisible(false);
    soundIcon.setVisible(false);
    victoriesButton.setVisible(false);
}
//ögeleri gösterir
public void show(){  2 usages
    soundIcon.setVisible(true);
    newGameButton.setVisible(true);
    exitGameButton.setVisible(true);
    creditsButton.setVisible(true);
    volumeSlider.setVisible(true);
    miKuriTitle.setVisible(true);
    victoriesButton.setVisible(true);
}
```

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.

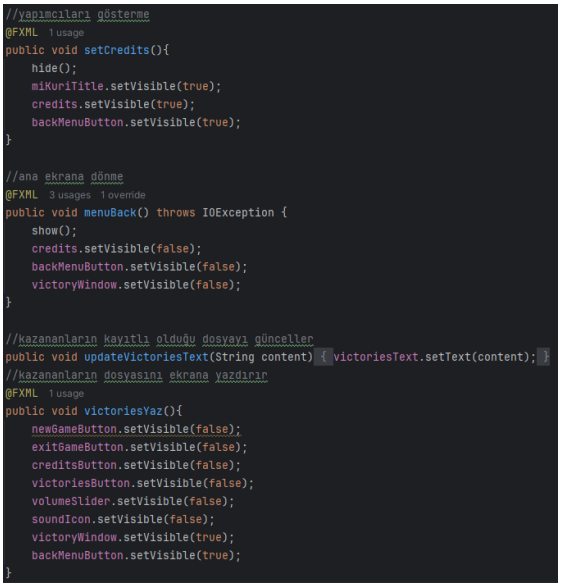
```
//yeni oyun başlatır
@FXML  1 usage
@Override
public void onNewGameButtonClick() {
    hide();
    try {
        gameDegisme();
    } catch (IOException ex) {
        throw new GameCrashedException(ex);
    }
}

//ana menüden oyuna geçmemizi sağlar
public void gameDegisme() throws IOException {  2 usages  1 override
    if(yesButton.getScene() != null){
        Stage window = (Stage) newGameButton.getScene().getWindow();
        if(window != null){
            Parent root = FXMLLoader.load(Objects.requireNonNull
                    (getClass().getResource( name: "menuToGame.fxml")));
            window.setScene(new Scene(root,  v: 600,  v1: 450));

            Rectangle2D screenLayout = Screen.getPrimary().getVisualBounds();
            window.setX((screenLayout.getWidth() - 600) / 2);
            window.setY((screenLayout.getHeight() - 450) / 2);
        }
    }
}
```

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

gameDegisme metodu çalıştırıldığında window nesnesini newGameButton düğmesinin bulunduğu sahneden menuToGame.fxml dosyasındaki sahneye çeker.

```
@FXML  2 usages  1 override
@Override
public void onExitGameButtonClick() {
    hide();

    //EMIN MISIN? (evet/hayir)
    txt.setVisible(true);
    txt.setText("Are you sure?");

    yesButton.setVisible(true);
    noButton.setVisible(true);

    //oyundan çıkar
    yesButton.setOnAction(event -> {
        yesButton.setDisable(true);
        noButton.setDisable(true);

        txt.setText("Exiting...");

        PauseTransition pause = new PauseTransition(Duration.seconds( v: 1));
        pause.setOnFinished(e -> {
            yesButton.setVisible(false);
            noButton.setVisible(false);
            txt.setVisible(false);
            System.exit( status: 0);
        }); pause.play();
    });

    //ana menüye döner
    noButton.setOnAction( _ -> {
        txt.setText("");

        yesButton.setVisible(false);
        noButton.setVisible(false);
        txt.setVisible(false);

        show();
    });
}
```

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.



```
//yapımcıları gösterme
@FXML  1 usage
public void setCredits(){
    hide();
    miKuriTitle.setVisible(true);
    credits.setVisible(true);
    backMenuButton.setVisible(true);
}

//ana ekrana dönme
@FXML  3 usages  1 override
public void menuBack() throws IOException {
    show();
    credits.setVisible(false);
    backMenuButton.setVisible(false);
    victoryWindow.setVisible(false);
}

//kazananların kayıtlı olduğu dosyayı günceller
public void updateVictoriesText(String content) { victoriesText.setText(content); }
//kazananların dosyasını ekrana yazdırır
@FXML  1 usage
public void victoriesYaz(){
    newGameButton.setVisible(false);
    exitGameButton.setVisible(false);
    creditsButton.setVisible(false);
    victoriesButton.setVisible(false);
    volumeSlider.setVisible(false);
    soundIcon.setVisible(false);
    victoryWindow.setVisible(true);
    backMenuButton.setVisible(true);
}
```

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.

updateVictoriesText metodu kazananların kayıtlı olduğu txt dosyasını her oyun sonucu geldiği zaman günceller.

victoriesYaz metodu ise gereksiz ögeleri gizleyerek hangi oyunun ne zamanda ve kimin kazandığını gösteren dosyayı ekrana yükler.

6. MenuToBarController Sınıfı

```
package com.pokemonlikegame.mikuri_project;

import ...

public class MenuToGameBarController extends StartController implements Initializable {
    @FXML
    private ProgressBar progressBar;

    //oyun yükleme ekranından oyuna geçişi sağlar
    @FXML 1 usage
    public void menuyeDegisme() throws IOException {
        if(progressBar.getScene() != null){
            Stage window = (Stage) progressBar.getScene().getWindow();
            if(window != null){
                Parent root = FXMLLoader.load(Objects.requireNonNull
                        (getClass().getResource( name: "mikuri-InGameScreen.fxml")));
                window.setScene(new Scene(root, v: 900, v1: 700));

                Rectangle2D screenLayout = Screen.getPrimary().getVisualBounds();
                window.setX((screenLayout.getWidth() - 900) / 2);
                window.setY((screenLayout.getHeight() - 700) / 2);
            }
        }
    }
}
```

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.

7. InGameController Sınıfı

```
package com.pokemonlikegame.mikuri_project;

import ...

public class InGameController extends MikuriController implements Initializable, InGameFunctionality, CharacterCards {
```

```
@FXML
private ImageView
        environment;
@FXML
private Button
        inGameToMenu,
        continueGame,
        exitFromInGame,
        evetButton,
        hayirButton,
        attackButton,
        contPlaying,
        finishGame;
@FXML
private Label pauseMenu;
@FXML
private Label contentWindow;
@FXML
private AnchorPane anchorPane;

private static InGameController instance;
private double xOffset = 0, yOffset = 0;

private String whoWon;  4 usages
private Attacker attacker;  1 usage
private boolean attackMode = false;  4 usag
private Card selectedCard = null;  8 usages
```

Bu sınıf MikuriController sınıfını miras alırken Initializable arayüzünü kendine uygular. Görüleceği üzere alanlar bu sınıfta tanımlanmıştır.

```java
//dizi içerisinde kart bilgileri tanımlanmıştır.
//{"İsim", "Can Değeri", "Element Türü", "Atak Gücü"}
private final String [][] characterData = {  2 usages
        {"Charizard", "120", "Fire", "90"},
        {"Blaziken", "117", "Fire", "82"},
        {"Camerupt", "143", "Fire", "84"},
        {"Blastoise","173", "Water", "83"},
        {"Greninja", "122", "Water", "87"},
        {"Swampert", "106", "Water", "78"},
        {"Lapras", "156", "Ice", "88"},
        {"Mamoswine", "133","Ice", "77"},
        {"Glalie", "125", "Ice", "74"},
        {"Pikachu", "60", "Electric", "110"},
        {"Jolteon", "101", "Electric", "82"},
        {"Magneton", "150", "Electric", "68"},
        {"Venusaur", "150", "Eco", "67"},
        {"Geodude", "120", "Eco", "80"},
        {"Bulbazaur", "150", "Eco", "70"},
        {"Scyther", "90", "Fly", "90"},
        {"Pidgey", "70", "Fly", "80"},
        {"Gyarados", "110", "Fly", "85"},
};

private static final ArrayList<ImageView> characterImages = new ArrayList<>();  8 usages
private static final List<ImageView> blueTeamImages = new ArrayList<>();  6 usages
private static final List<ImageView> redTeamImages = new ArrayList<>();  7 usages
private static final Map<ImageView, Card> cardImageMap = new HashMap<>();  7 usages

//getter metotlarını oluşturma
public static List<ImageView> getBlueTeamImages(){ return blueTeamImages; }  1 usage
public static List<ImageView> getRedTeamImages(){ return redTeamImages; }  1 usage
public static Card getCardForImageView(ImageView imageView) {  3 usages
    return cardImageMap.get(imageView);
}
```

Burada tanımlanan alanlar 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.

```java
//ilk başta çalışması gereken ögeleri çalıştırır
@Override
public void initialize(URL url, ResourceBundle resourceBundle) {
    //şuan çalışan sınıfı (this) instance nesnesine kaydeder
    instance = this; //singleton tasarım

    environment.fitWidthProperty().bind(anchorPane.widthProperty());
    environment.fitHeightProperty().bind(anchorPane.heightProperty());

    //bazı ögeler gizlenir
    contentWindow.setVisible(false);
    attackButton.setVisible(false);
    contPlaying.setVisible(false);
    finishGame.setVisible(false);

    //ekranı hareket ettirmemizi sağlar
    anchorPane.setOnMousePressed(event -> {
        xOffset = event.getSceneX();
        yOffset = event.getSceneY();
    });
    anchorPane.setOnMouseDragged(event -> {
        Stage stage = (Stage) anchorPane.getScene().getWindow();
        stage.setX(event.getScreenX() - xOffset);
        stage.setY(event.getScreenY() - yOffset);
    });

    //spesifik bir tuşa basıldığında pause metodunu çağırır
    anchorPane.setOnKeyPressed(this::pause);
    anchorPane.requestFocus(); //spesifik bir tuşa basıldığında pause metodunu çağırır

    //attacker nesnesi tanımlanır
    attacker = new Attacker( controller: this);

    gameStart();
}
```

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.

```java
public void gameStart(){
    blueTeamImages.clear();
    redTeamImages.clear();
    characterImages.clear();
    cardImageMap.clear();
    attackMode = false;
    selectedCard = null;

    int anchorHeight = 700;
    double cellWidth = 106, cellHeight = 86;
    double padding = 10;

    //Kartların pozisyonlarını ayarlamak için dizi oluşturuldu
    double[] xPositions = {
            padding * 8, padding * 22, padding * 8,
            padding * 8, padding * 22, padding * 8,
            padding * 71, padding * 58, padding * 71,
            padding * 71, padding * 58, padding * 71};
    double[] yPositions = {
            padding + 30,
            padding + (cellHeight + 5),
            padding + 2 * (cellHeight + 5),
            anchorHeight - 3 * (cellHeight + 5) - padding,
            anchorHeight - 2 * (cellHeight + 5) - padding,
            anchorHeight - (cellHeight + 5) - padding - 30,
            padding + 30,
            padding + (cellHeight + 5),
            padding + 2 * (cellHeight + 5),
            anchorHeight - 3 * (cellHeight + 5) - padding,
            anchorHeight - 2 * (cellHeight + 5) - padding,
            anchorHeight - (cellHeight + 5) - padding - 30};

    //randomCD tanımlanır ve characterData'dan rastgele kartları seçer
    ArrayList<String[]> randomCD = new ArrayList<>();
    for (int i = 0; i < 12; i++) {
        randomCD.add(characterData[(int) (Math.random() * characterData.length)]);
    }
    //kartları arenaya koyan cardToStage metoduna parametreler yazılır
    for (int i = 0; i < 12; i++) {
        cardToStage(new Card(randomCD.get(i)), xPositions[i], yPositions[i]);
    }
```

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.

```
//kartları arenaya taşımayı dener
@Override  1 usage
public void cardToStage(Card card, double x, double y){
    try{
        //resimler character nesnesine aktarılır ve düzenlenir
        String path = "/types/" + card.getName().toLowerCase() +".png";
        ImageView character = new ImageView();
        character.setImage(new Image(Objects.requireNonNull
                (getClass().getResourceAsStream(path))));
        character.setFitWidth(100);
        character.setFitHeight(80);
        character.setPreserveRatio(false);

        character.setLayoutX(x);
        character.setLayoutY(y);

        //mavi tarafta olan kartlar varsa mavi takıma aktarır
        if (isOnBluePart(y)){...}
        //kırmızı tarafta olan kartlar kırmızı takıma aktarılır
        else{...}

        //kart ile character arasındaki bağlantıyı sağlayıp kartları ekler
        cardImageMap.put(character, card);
        anchorPane.getChildren().add(character);
        characterImages.add(character);
    }catch (NullPointerException e){
        throw new GameCrashedException("Image " + card.getName() + " not found!!!!", e);
    }
}
private boolean isOnBluePart(double y) { return (y >= 400); }
```

```
//mavi tarafta olan kartlar varsa mavi takıma aktarır
if (isOnBluePart(y)){
    blueTeamImages.add(character);
    //mavi tarafta kalan kart seçildiği zaman bilgileri gösterir
    character.setOnMousePressed(event -> {
        if(event.isPrimaryButtonDown()){
            selectedCard = card;
            contentWindow.setVisible(true);
            attackButton.setVisible(true);

            contentWindow.setText(card.getName() + "\nHP: " + card.getHp() +
                    "\nType: " + card.getType() + "\nAttacks: " + card.getAttacks());
        }
    });
}
//kırmızı tarafta olan kartlar kırmızı takıma aktarılır
else{
    redTeamImages.add(character);
    //kırmızı tarafta kalan kart seçildiği zaman bilgileri gösterir
    //birde saldırı ekranını yükletir
    character.setOnMousePressed(event -> {
        contentWindow.setVisible(true);
        attackButton.setVisible(false);
        contentWindow.setText(card.getName() + "\nHP: " + card.getHp() +
                "\nType: " + card.getType() + "\nAttacks: " + card.getAttacks());

        //attack butonuna basıldığında kart atak yapmaya hazırlanır
        if(event.isPrimaryButtonDown() && attackMode && selectedCard != null){
            attackPressed(card, character);
            contentWindow.setVisible(true);

            attackMode = false;
            selectedCard = null;
            attackButton.setVisible(false);
        }
    });
}
```

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.

```
private void attackPressed(Card target, ImageView targetImage){  1 usage
    if (selectedCard == null){
        System.out.println("No attacker selected");
        return;
    }

    int damage = selectedCard.getAttacks();
    if(cardImageMap.containsKey(targetImage)){
        Card targetCard = cardImageMap.get(targetImage);
        targetCard.setHp(targetCard.getHp() - damage);
    }

    if(target.getHp() <= 0){
        target.setHp(0);
        contentWindow.setText(target.getName() + " has fainted!!!");

        removeCardFromStage(targetImage);
        checkWin();
    } else {
        contentWindow.setText(selectedCard.getName() + " attacked\n" + target.getName()
                + "\nfor " + damage + " of damage!");
    }

    if(!redTeamImages.isEmpty()){
        PauseTransition transition = new PauseTransition(Duration.seconds( v: 3));
        transition.setOnFinished(event -> {
            Attacker.enemyAttack();
        });
        transition.play();
    }
}
```

```
@FXML  1 usage
@Override
public void attackPressed(){
    if(selectedCard != null){
        attackButton.setVisible(false);
        attackMode = true;
        contentWindow.setText("Attack!!!\nSelect a pokemon from the red side");
    }else{
        System.out.println("Select a pokemon from your side first");
    }
}

public static void updateContentWindow(String text){  2 usages
    if (instance != null && instance.contentWindow != null){
        javafx.application.Platform.runLater(() -> {
            instance.contentWindow.setVisible(true);
            instance.contentWindow.setText(text);
        });
    }
}

@Override  2 usages
public void removeCardFromStage(ImageView cardImage){
    Card character = cardImageMap.get(cardImage);
    if (character != null){
        anchorPane.getChildren().remove(cardImage);
        characterImages.remove(cardImage);
        cardImageMap.remove(cardImage);
    }
    if(redTeamImages.contains(cardImage)){
        redTeamImages.remove(cardImage);
    }

    if(blueTeamImages.contains(cardImage)){
        blueTeamImages.remove(cardImage);
    }
}
```

```java
@Override  3 usages
public void checkWin(){
    if(redTeamImages.isEmpty()){
        contentWindow.setVisible(true);
        contentWindow.setText("Congratulations!!! - You win!!!");
        disableGame();
        whoWon = "Player won";
        SaveInFile.FileSave(whoWon);

        PauseTransition transition = new PauseTransition(Duration.seconds( v: 3));
        transition.setOnFinished(event -> {...});
        transition.play();
    }
    else if(blueTeamImages.isEmpty()){
        contentWindow.setVisible(true);
        contentWindow.setText("You've lost!!!");
        disableGame();
        whoWon = "CPU won";
        SaveInFile.FileSave(whoWon);

        PauseTransition transition = new PauseTransition(Duration.seconds( v: 3));
        transition.setOnFinished(event -> {...});
        transition.play();
    }
}
```

```java
if(redTeamImages.isEmpty()){
    contentWindow.setVisible(true);
    contentWindow.setText("Congratulations!!! - You win!!!");
    disableGame();
    whoWon = "Player won";
    SaveInFile.FileSave(whoWon);

    PauseTransition transition = new PauseTransition(Duration.seconds( v: 3));
    transition.setOnFinished(event -> {
        javafx.application.Platform.runLater(() -> {
            for (ImageView character : characterImages){
                character.setVisible(false);
            }
            environment.setVisible(false);
            evetButton.setVisible(false);
            hayirButton.setVisible(false);
            contentWindow.setVisible(false);
            attackButton.setVisible(false);
            evetButton.setVisible(false);
            hayirButton.setVisible(false);
            exitFromInGame.setVisible(false);
            inGameToMenu.setVisible(false);
            continueGame.setVisible(false);
            environment.setVisible(false);
            contPlaying.setVisible(true);
            finishGame.setVisible(true);

            pauseMenu.setText("Do you want to play again?");
        });
    });
    transition.play();
}
```

```java
else if(blueTeamImages.isEmpty()){
    contentWindow.setVisible(true);
    contentWindow.setText("You've lost!!!");
    disableGame();
    whoWon = "CPU won";
    SaveInFile.FileSave(whoWon);

    PauseTransition transition = new PauseTransition(Duration.seconds( v: 3));
    transition.setOnFinished(event -> {
        javafx.application.Platform.runLater(() -> {
            for (ImageView character : characterImages){
                character.setVisible(false);
            }
            environment.setVisible(false);
            evetButton.setVisible(false);
            hayirButton.setVisible(false);
            contentWindow.setVisible(false);
            attackButton.setVisible(false);
            evetButton.setVisible(false);
            hayirButton.setVisible(false);
            exitFromInGame.setVisible(false);
            inGameToMenu.setVisible(false);
            continueGame.setVisible(false);
            environment.setVisible(false);
            contPlaying.setVisible(true);
            finishGame.setVisible(true);

            pauseMenu.setText("Do you want to play again?");
        });
    });
    transition.play();
}
```

```java
private void disableGame(){  2 usages
    attackButton.setDisable(true);
    for (ImageView character : characterImages){
        character.setDisable(true);
        character.setOpacity(0.5);
    }

    contentWindow.setDisable(false);
    contentWindow.setVisible(true);
}

//ESC tuşuna basıldığında oyunu durdurur
@FXML  2 usages
@Override
public void pause(KeyEvent hareket) {
    if(hareket.getCode() == KeyCode.ESCAPE){
        //oyun durdurulur ve ögeler düzenlenir
        for(ImageView character : characterImages){
            character.setVisible(false);
        }
        environment.setVisible(false);
        evetButton.setVisible(false);
        hayirButton.setVisible(false);
        contentWindow.setVisible(false);
        attackButton.setVisible(false);
    }
}
```

```java
//sahnenin ana menüye dönmesini sağlar
@FXML  3 usages
@Override
public void menuBack() throws IOException {
    Stage window = (Stage) inGameToMenu.getScene().getWindow();
    MikuriController.getMedia().stop();
    if(window != null){
        Parent root = FXMLLoader.load(Objects.requireNonNull(getClass().getResource
                ( name: "/com/pokemonlikegame/mikuri_project/mikuri-StaLoaScreen.fxml")));
        window.setScene(new Scene(root,  v: 600,  v1: 450));
        window.setTitle("Mikuri - Menu");

        Rectangle2D screenLayout = Screen.getPrimary().getVisualBounds();
        window.setX((screenLayout.getWidth() - 600) / 2);
        window.setY((screenLayout.getHeight() - 450) / 2);
    }
}

@FXML  2 usages
@Override
public void gameDegisme() throws IOException {
    if(contPlaying.getScene() != null){
        Stage window = (Stage) contPlaying.getScene().getWindow();
        if(window != null){
            Parent root = FXMLLoader.load(Objects.requireNonNull(getClass().getResource
                    ( name: "/com/pokemonlikegame/mikuri_project/menuToGame.fxml")));
            window.setScene(new Scene(root,  v: 600,  v1: 450));
            window.setTitle("Mikuri - Menu");

            Rectangle2D screenLayout = Screen.getPrimary().getVisualBounds();
            window.setX((screenLayout.getWidth() - 600) / 2);
            window.setY((screenLayout.getHeight() - 450) / 2);
        }
    }
}
```

```
@FXML 2 usages
@Override
public void onExitGameButtonClick() {
    try{
        inGameToMenu.setVisible(false);
        continueGame.setVisible(false);
        exitFromInGame.setVisible(false);

        pauseMenu.setText("Are you sure?");

        evetButton.setVisible(true);
        hayirButton.setVisible(true);

        evetButton.setOnAction(event -> {...});
        hayirButton.setOnAction(event -> {...});
    } catch (GameCrashedException e) {
        throw new GameCrashedException("Fatal error! Please try it later.", e);
    }
}
@FXML 1 usage
@Override
public void continueGame() {
    environment.setVisible(true);
    contentWindow.setVisible(false);
    attackButton.setVisible(false);
    for(ImageView character : characterImages){
        character.setVisible(true);
    }
}
```

8. Attacker Sınıfı

```
package Cards;                                                                    48
import ...

public class Attacker {
    private static InGameController controller;  4 usages

    //çağrıldığında gelen controller nesnesini yenisi ile değiştirir
    public Attacker(InGameController controller) { Attacker.controller = controller; }

    //bilgisayarın (CPU'nun) hamlesini yaptığı metottur
    public static void enemyAttack() {  1 usage
        List<ImageView> blueTeamImages = InGameController.getBlueTeamImages();
        List<ImageView> redTeamImages = InGameController.getRedTeamImages();
        if (blueTeamImages.isEmpty()) {
            controller.checkWin();
            return;
        }
        if (redTeamImages.isEmpty()) {
            return;
        }
        //CPU burada rastgele hamleler yapacak bir biçimde tanımlanır
        Random random = new Random();
        ImageView randomTarget = blueTeamImages.get(random.nextInt(blueTeamImages.size()));
        Card targetCard = InGameController.getCardForImageView(randomTarget);
        if (targetCard == null) {
            return;
        }
        ImageView randomEnemyAttacker = redTeamImages.get(random.nextInt(redTeamImages.size()));

        //Hedeflenen kartın hp'sini azaltmak için yöntem aşırı yüklemesi yoluyla işlevi çağırır
        enemyAttack(randomEnemyAttacker, randomTarget, blueTeamImages);
    }
```

```
//
public static void enemyAttack(ImageView attacker, ImageView target, List<ImageView> blueTeamImages){
    if (attacker == null || target == null) {
        return;
    }

    Card targetCard = InGameController.getCardForImageView(target);
    Card attackerCard = InGameController.getCardForImageView(attacker);

    if(targetCard == null || attackerCard == null) {
        return;
    }

    //cana hasar verme mekaniği yazılıyor
    int damage = attackerCard.getAttacks();
    int updatedHp = targetCard.getHp() - damage;
    targetCard.setHp(updatedHp);

    //eğer CPU bir canlıyı yok etmişse o kartı siler ve kazanıp kazanmadığına bakar
    if(targetCard.getHp() <= 0){
        targetCard.setHp(0);
        controller.removeCardFromStage(target);
        blueTeamImages.remove(target);
        if(blueTeamImages.isEmpty()){
            controller.checkWin();
        } else{
            String faintedMessage = targetCard.getName() + " has fainted!!!";
            InGameController.updateContentWindow(faintedMessage);
        }
    }
    //eğer CPU bir canlıyı yok edememiş ise o canlıya verdiği hasarı ekrana yansıtır
    else {
        String attackMessage = "Enemy's " + attackerCard.getName() + " attacked\n" +
                targetCard.getName() + "\nfor " + attackerCard.getAttacks() + " damage!\n";
        InGameController.updateContentWindow(attackMessage);
    }
```

9. Card Sınıfı

```
package Cards;

public class Card extends BaseCard {
    private static int uniqueIdCounter = 1;  1 usage

    public Card(String[] characterDatum) {  1 usage
        super(uniqueIdCounter++, characterDatum[0],
                Integer.parseInt(characterDatum[1]),
                characterDatum[2],
                Integer.parseInt(characterDatum[3]));
    }
}
```

10. CardFunctionality Sınıfı

```
package Functionality;

public abstract class CardFunctionality {  2 usages  2 inheritors
    public abstract String getName();  1 implementation

    public abstract int getHp();  6 usages  1 implementation

    public abstract void setHp(int hp);  4 usages  1 implementation

    public abstract String getType();  2 usages  1 implementation

    public abstract int getAttacks();  5 usages  1 implementation

    public abstract String toString();  1 implementation
}
```

11. SaveInFile Sınıfı

```
package InFile;

import ...

public class SaveInFile {
    private static MikuriController controller;  4 usages

    public static void setController(MikuriController mikuriController) { controller = mikuriController; }

    public static void FileSave(String winner) {  2 usages
        String file = "results.txt";
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
        String time = LocalDateTime.now().format(formatter);
        String info = time + " - " + winner;

        try (BufferedWriter writer = new BufferedWriter(new FileWriter(file,  append: true))){
            writer.write(info);
            writer.newLine();
        } catch (IOException e) {
            //System.err.println("Error printing to file: " + e.getMessage());
        }
    }
```

```java
public static void printInfo(){  1 usage
    if(controller == null){
        System.out.println("Error: controller is null");
    }

    StringBuilder content = new StringBuilder();
    List<String> filteredResults = new ArrayList<>();

    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
    LocalDateTime now = LocalDateTime.now();

    try (BufferedReader reader = new BufferedReader(new FileReader( fileName: "results.txt"))){
        String line;
        while ((line = reader.readLine()) != null){
            content.append(line).append("\n");

            String []parts = line.split( regex: " - ");
            if(parts.length > 1){
                LocalDateTime entryTime = LocalDateTime.parse(parts[0], formatter);

                if(ChronoUnit.HOURS.between(entryTime, now) < 24){
                    filteredResults.add(line);
                }
            }
        }
        Platform.runLater(() -> {
            String allResults = content.toString();
            controller.updateVictoriesText(allResults);

            String filteredContent = String.join( delimiter: "\n", filteredResults);
            controller.updateVictoriesText( content: "In the last 24 hours:\n" + filteredContent);
        });
    } catch (IOException e) {
        System.out.println("There are not previous matches registered in the system.");
    } finally {
        System.out.println("Previous victory data processing finished.");
    }
}
```

12.InGameFunctionality Arayüzü

```java
package Functionality;

import javafx.scene.input.KeyEvent;

public interface InGameFunctionality {
    void pause(KeyEvent hareket);  2 usage

    void continueGame();  1 usage  1 implemer

    void gameStart();  1 usage  1 implementati

    void attackPressed();  1 usage  1 implem
}
```

13.BaseCard Sınıfı

```java
package Cards;


import Functionality.CardFunctionality;

public class BaseCard extends CardFunctionality {  1 inheritor
    private final int id;  1 usage
    private final String name;  3 usages
    private int hp;  4 usages
    private final String type;  3 usages
    private final int attack;  3 usages


    public BaseCard(int id, String name, int hp, String type, int attack) {  1 usage
        this.id = id;
        this.name = name;
        this.hp = hp;
        this.type = type;
        this.attack = attack;
    }

    @Override
    public String getName() { return name; }

    @Override  6 usages
    public int getHp() { return hp; }

    @Override  4 usages
    public void setHp(int hp) { this.hp = hp; }

    @Override  2 usages
    public String getType() { return type; }

    @Override  5 usages
    public int getAttacks(){ return attack; }

    @Override
    public String toString() {
        return "Card [name=" + name + ", hp=" + hp + ", type=" + type + ", " +
                ", Attacks =" + attack +"]";
    }
}
```

14.CharacterCards Arayüzü

```java
package Functionality;

import ...

public interface CharacterCards {  2 usages  1 implementation
    void cardToStage(Card card, double x, double y);  1 usage  1 implementation

    void removeCardFromStage(ImageView cardImage);  2 usages  1 implementation

    void checkWin();  3 usages  1 implementation
}
```

15.GameCrashedException Sınıfı

```java
package Exception;

public class GameCrashedException extends RuntimeException {  6 usages
        public GameCrashedException(String msg, Throwable cause) { super(msg, cause); }

        public GameCrashedException(Throwable cause) { super(cause); }
}
```

III. Sonuç

Oyunun hem yapım hem de bitim aşamasında birçok sorunla karşılaştık. En fazla zamanımızı alan hata/hatalar bu sorunlar(bugs) olmuştur. Çözüm olarak oyunumuzu test ede ede bu sorunların büyük bir kısmı çözülmüştür.

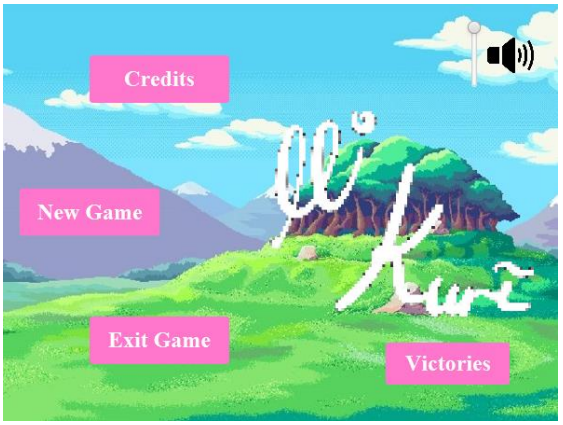
JavaFX'in kullanımında birkaç sıkıntı yaşandı, özellikle bu kütüphaneyi önceden hiç kullanmadığımızdan bu kütüphane ile oyun yapılmasında zorluklar yaşadık. Bu yüzden ilk başta JavaFX eğitimini aldık ve SceneBuilder ile işler daha basit hale getirilmiştir.

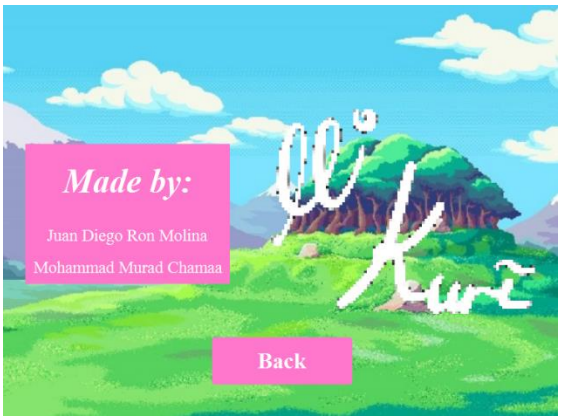
Canavarları sahneye yerleştirilmesi uzunca bir zamanımızı almıştır. Birçok başarısız deneme sonucunda çözümü ilk başta canavarların bilgilerini ArrayList dizisine tek tek for döngüsü ile kaydedip sonra da yine for döngüsü ile başka bir metoda atanıp takımlara göre sahneye atılmıştır.

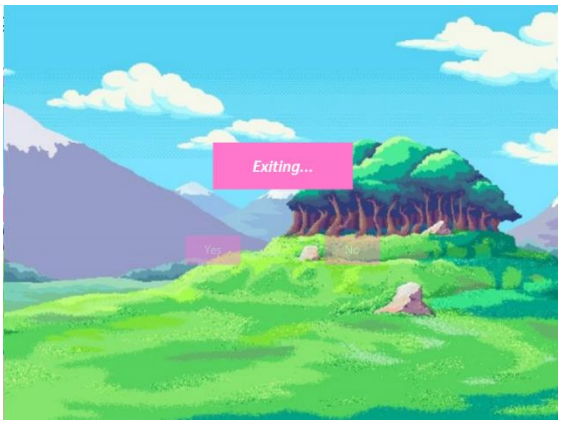
Her ne kadar Java, diğer programlama dillerine kıyasla biraz daha karmaşık olsa da Nesne Yönelimli Programlama (OOP) yaklaşımıyla sunduğu esneklik ve güçlü altyapısı sayesinde oyun geliştirme için etkili bir alternatif sunmaktadır.

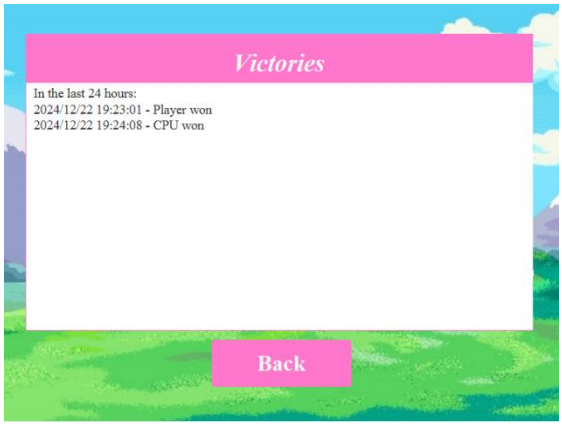
IV. Ekler

Oyunun Ekran Görüntüleri:

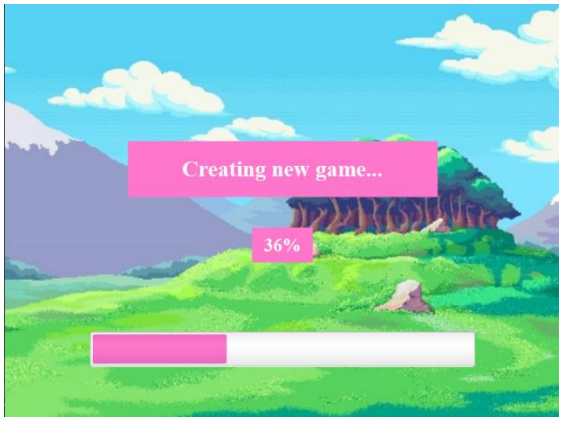




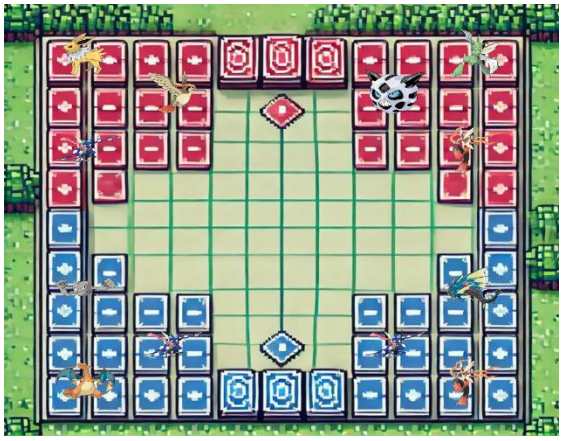



Victories
In the last 24 hours:
2024/12/22 19:23:01 - Player won
2024/12/22 19:24:08 - CPU won
Back


Are you sure?
Yes
No


Creating new game...
36%

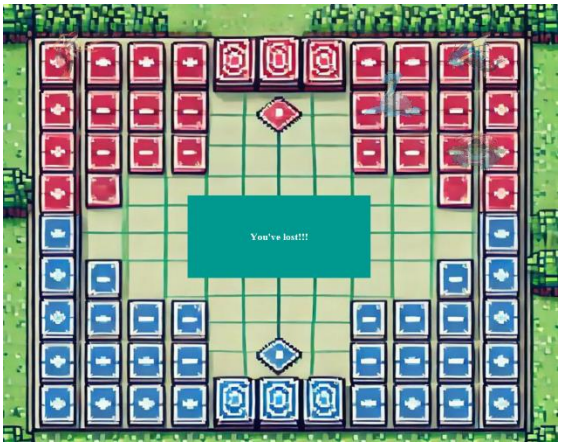

Exiting...
Yes
No

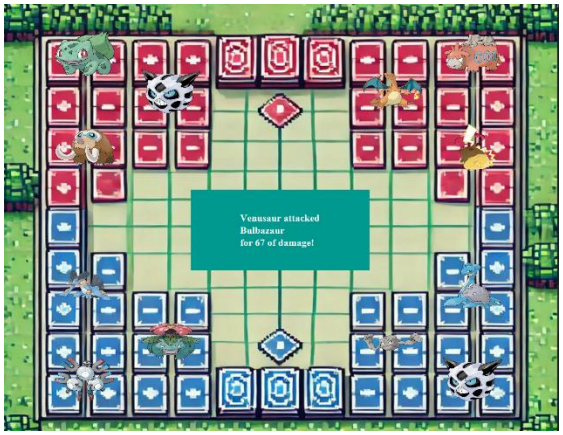


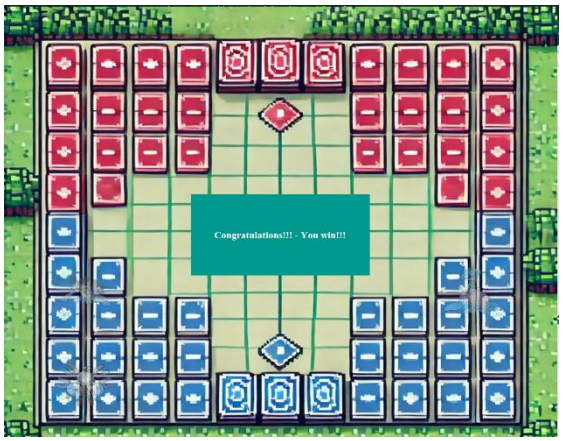

Charizard
HP: 120
Type: Fire
Attacks: 90


Game paused
Main menu
Continue game
Exit game


Venusaur
HP: 150
Type: Eco
Attacks: 67
ATTACK!







Kaynakça

[1] GeeksforGeeks contributers, “Top Applications of Java in Real World,” GeeksforGeeks, 2024. [Online]. Available: https://www.geeksforgeeks.org/top-applications-of-java-in-real-world/. [Accessed 22 Aralık 2024].

[2] Wikipedia contributors, “Java (programming language),” Wikipedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Java_(programming_language). [Accessed 22 Aralık 2024].

[3] contributors, Spectrum, “Top Programming Languages 2024,” IEEE Spectrum, 2024. [Online]. Available:

https://spectrum.ieee.org/top-programming-languages-2024. [Accessed 22 Aralık 2024].

[4] A. Rashid, “Developments in high performance concrete technology,” 1992. [Online]. Available: https://ieeexplore.ieee.org/document/714612. [Accessed 22 Aralık 2024].

[5] C. L. v. N. Wardrip-Fruin, “Game AI: The state of the industry,” 2008. [Online]. Available: https://www.cse.unr.edu/~sushil/class/gas/papers/GameAIp27-lewis.pdf. [Accessed 22 Aralık 2024].

[6] M. R. I. a. M. A. R. M. A. Hossain, “Design and implementation of a Java-based real-time web application for smart home monitoring system,” IEEE Access, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/2020. [Accessed 22 Aralık 2024].

[7] Oracle, “JavaFX Overview,” 2012. [Online]. Available: https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm. [Accessed 22 Aralık 2024].

[8] GeeksforGeeks contributors, “JavaFX Tutorial,” 2024. [Online]. Available: https://www.geeksforgeeks.org/javafx-tutorial/. [Accessed 22 Aralık 2024].