
**T.C**
**KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ**
**BİLGİSAYAR MÜHENDİSLİĞİ**
**YAZILIM LAB DERSİ**

# MAZERET PROJESİ

**HAZIRLAYANLAR**
**Selime Selin CAN – 210501005**
**Zeynep İrem AKYALÇIN - 210501008**

**DERS SORUMLUSU**
**DR. ÖĞR. ÜYESİ NUR BANU ALBAYRAK**

**19 HAZİRAN 2024**

# 1. AMAÇ

Projenin amacı, kullanıcıların Pokémon kartlarıyla oynayabileceği eğlenceli ve etkileşimli bir oyun geliştirmektir. Bu oyun, kullanıcı ve bilgisayar arasında geçen bir dizi kart karşılaştırması üzerine kuruludur ve oyuncuların strateji kullanarak rakiplerine üstünlük sağlamasını gerektirir.

Projede gerçekleştirilmesi beklenenler:

- Kullanıcı ve bilgisayar arasında Pokémon kart oyunu oluşturulması.
- Kullanıcının ve bilgisayarın kart seçimlerini yapabilmesi.
- Kartların hasar değerleri üzerinden karşılaştırılması.
- Kullanıcı arayüzü üzerinden oyun sürecinin takip edilebilmesi.
- Oyun sonunda kazananın belirlenmesi ve kullanıcının bilgilendirilmesi.

# 2. GEREKSİNİM ANALİZİ

## 2.1 Arayüz Gereksinimleri

- Kullanıcı arayüzü, kullanıcı ve bilgisayarın kart seçimlerini yapabileceği ve sonuçları görebileceği bir alan içermelidir.
- Kullanıcı arayüzü, ortadaki kartları ve her iki oyuncunun seçtiği kartları net bir şekilde göstermelidir.
- Oyun sırasında kullanıcıya yönlendirmeler yapacak butonlar (kart seç, devam et, oyna) yer almalıdır.
- Oyunun başlangıcında oyun modunu seçebilme yeteneği sunulmalıdır.
- Donanım arayüz gereksinimleri: Bilgisayar ve fare, oyun kontrolü için gereklidir.

## 2.2 Fonksiyonel Gereksinimler

- Kullanıcı ve bilgisayar için kart seçim fonksiyonları.
- Kartların hasar değerlerine göre karşılaştırma fonksiyonu.
- Puan hesaplama ve oyun sonucunu belirleme fonksiyonu.
- Kullanıcı arayüzünde kartları ve puanları görüntüleme fonksiyonu.
- Oyun modunu belirleme (kullanıcı vs bilgisayar veya bilgisayar vs bilgisayar).

# 3. TASARIM

## 3.1 Mimari Tasarım

- **Kullanıcı arayüzü modülü:** Kartların ve butonların görüntülenmesi.
- **Oyun mantığı modülü:** Kart seçimleri, karşılaştırma ve puan hesaplama.
- **Veri yönetimi modülü:** Kart verilerinin saklanması ve işlenmesi.

## 3.2 Kullanılacak Teknolojiler

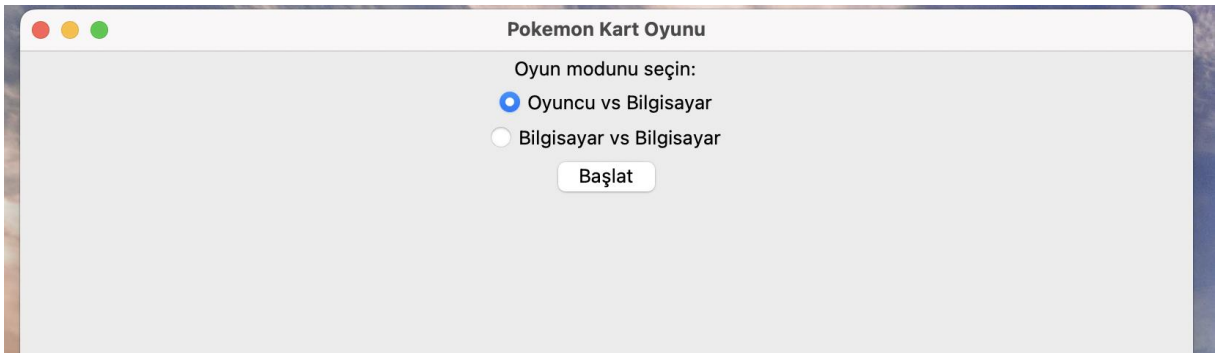
- **Yazılım dili:** Python
- **Harici kütüphaneler:**
  - Tkinter (kullanıcı arayüzü için)
  - Random (rastgele seçimler için)
- **Diğer teknolojiler:** Python'un standart kütüphaneleri ve özellikleri kullanılacaktır.

## 3.3 Veritabanı Tasarımı

Bu proje için veri tabanı tasarımı gerekmemektedir.

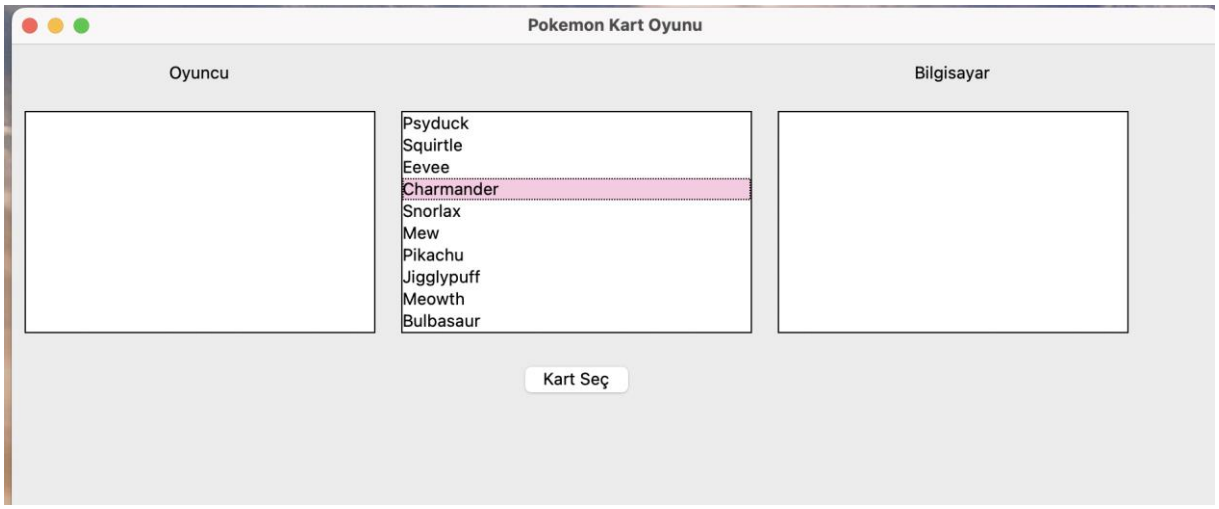
## 3.4 Kullanıcı Arayüzü Tasarımı

- **Ana ekran:** Kullanıcı, bilgisayar ve ortadaki kartların görüntülendiği alan.
- **Seçim ekranı:** Kartların seçimi ve karşılaştırılması için butonlar.
- **Oyun ekranı:** Sonuçların ve puanların gösterildiği alan.
- **Uygulamanın çalıştırılması:** Ana Python dosyasının çalıştırılmasıyla başlar.
- **Ekran çıktıları:**
  - Oyun başlangıç ekranı (mod seçimi).
  - Kart seçimi ekranı.
  - Oyun sonucu ekranı.



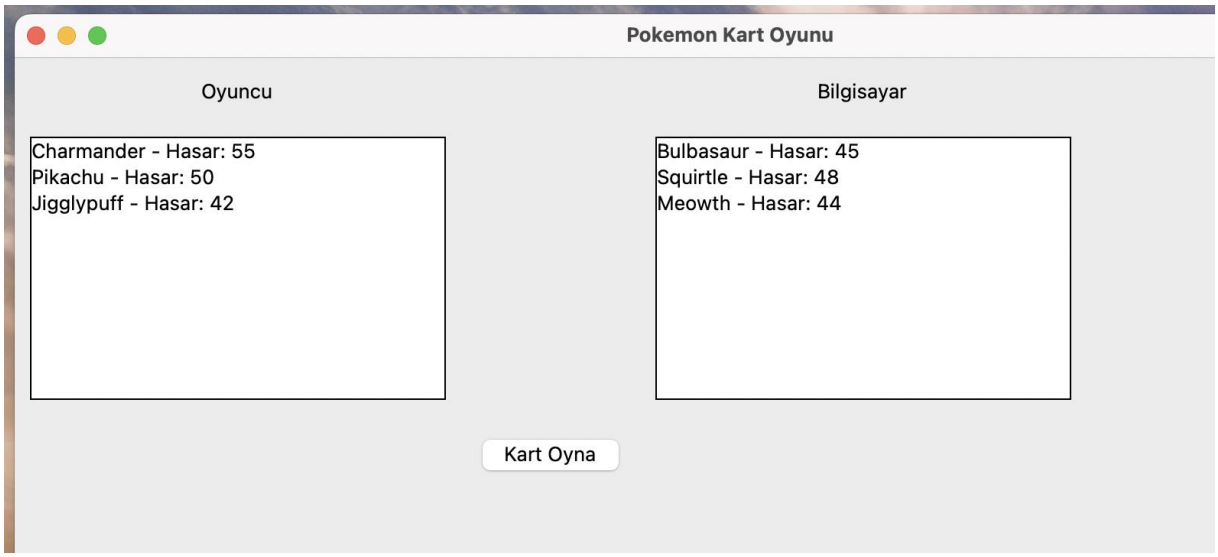

Öncelikle Pokemon Kart Oyunu penceresi açılıyor ve oyun modu seçimi yapılıyor.

Oyuncu vs Bilgisayar seçtiğimizi varsayarsak aşağıdaki görseller eşliğinde oyun devam ediyor.

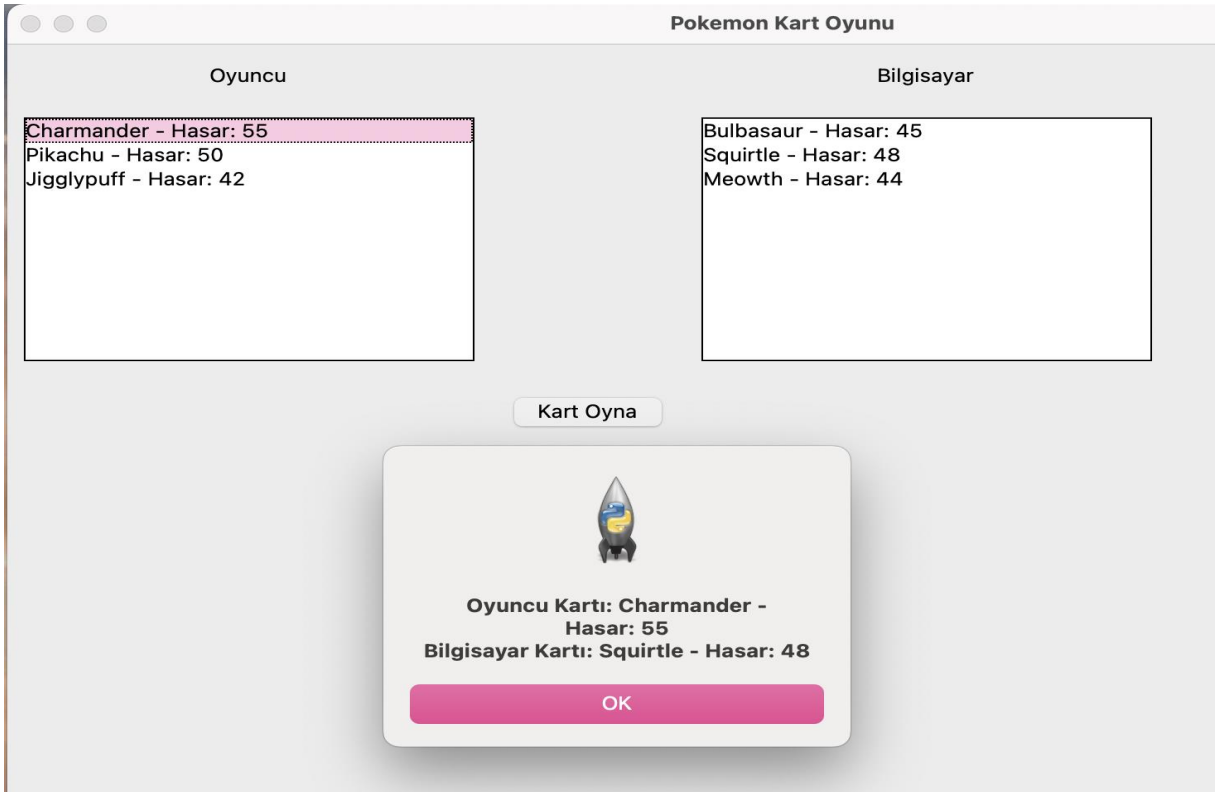



Bu aşamada ortada bulunan kartların hasar değerleri gözükmüyor.

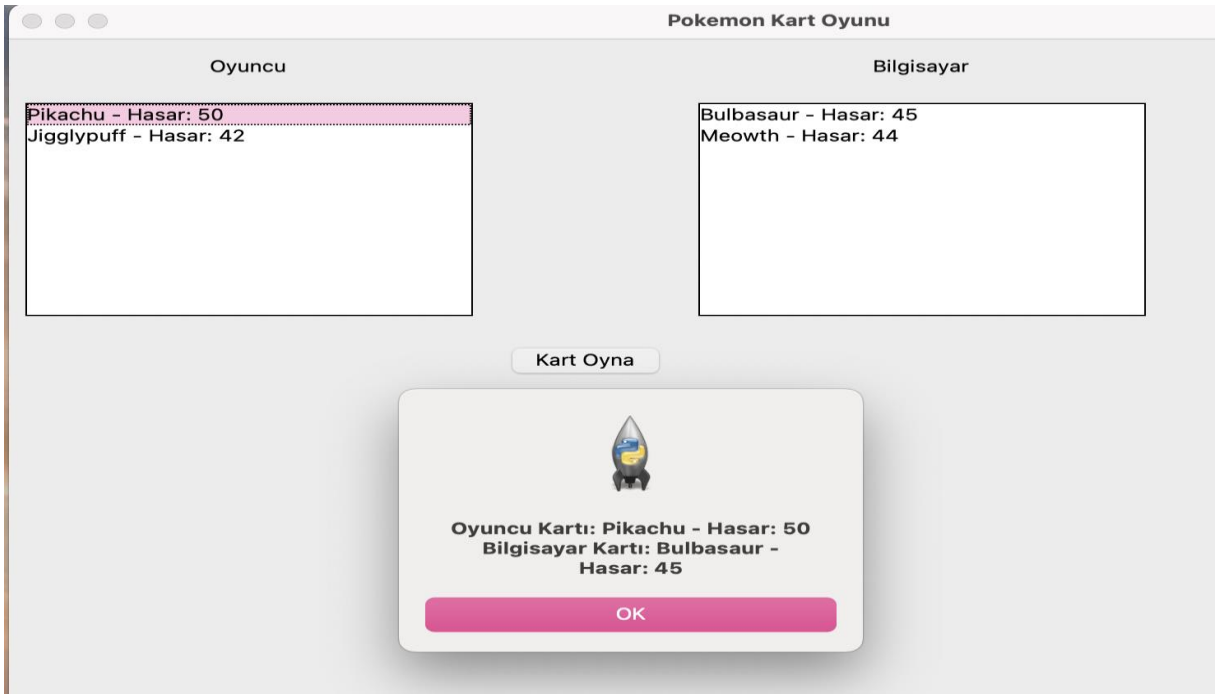
Kullanıcı olarak biz 1 kart seçtiğimizde bilgisayar da otomatik 1 kart seçiyor. Bu işlem 3'er kart seçene kadar devam ediyor.



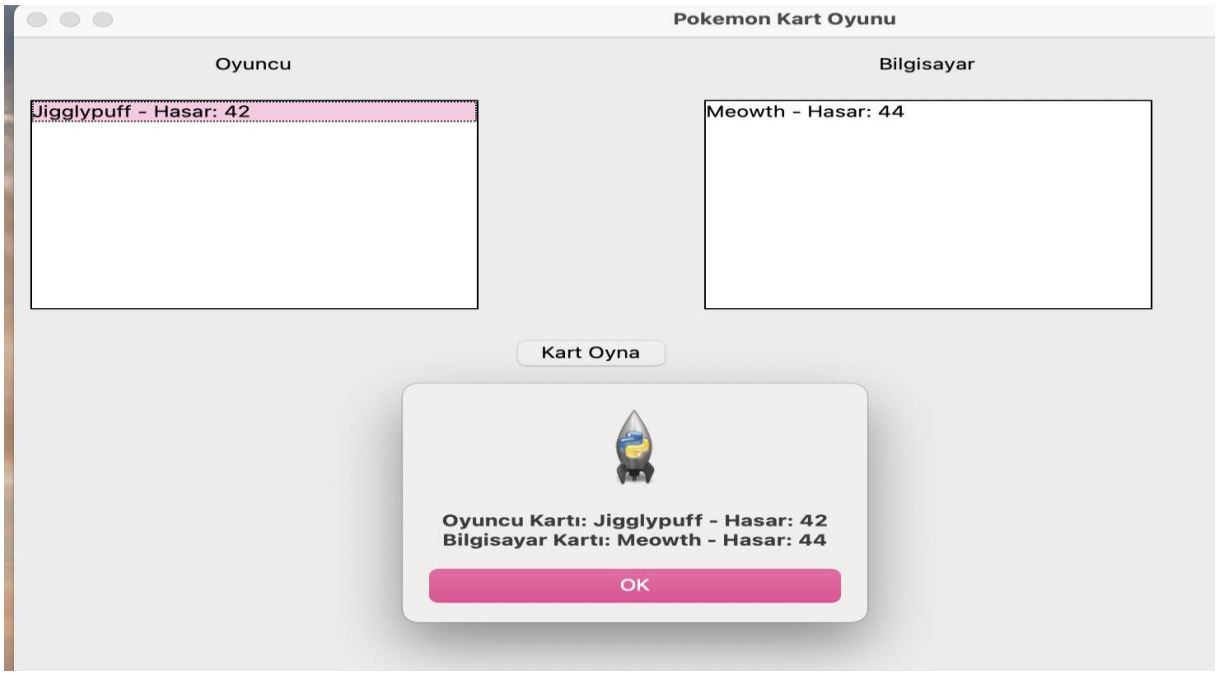

Kartları seçtikten sonra hasar değerleri artık gözüküyor.



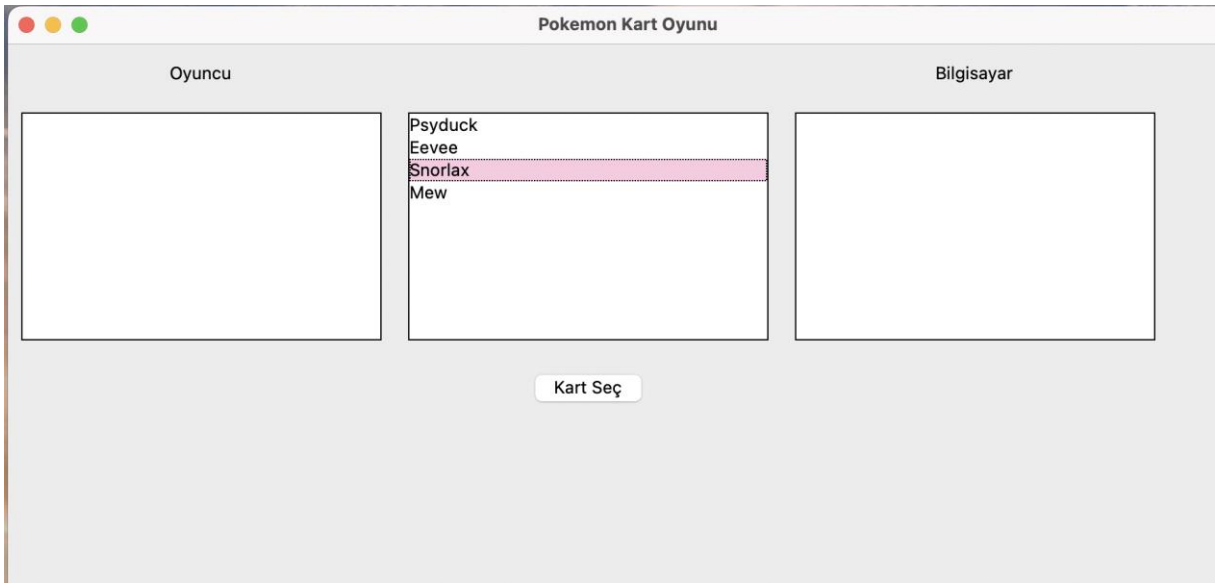

Kullanıcı olarak biz bir kart seçip 'Kart Oyna' butonuna bastığımızda bilgisayar da rastgele bir kartı seçiyor. Kartlar karşılaştırılıyor ve hasar değeri büyük olan oyuncu 5 puanı hanesine yazdırıyor.




Oyunun 2. eli görselde gözüküyor.

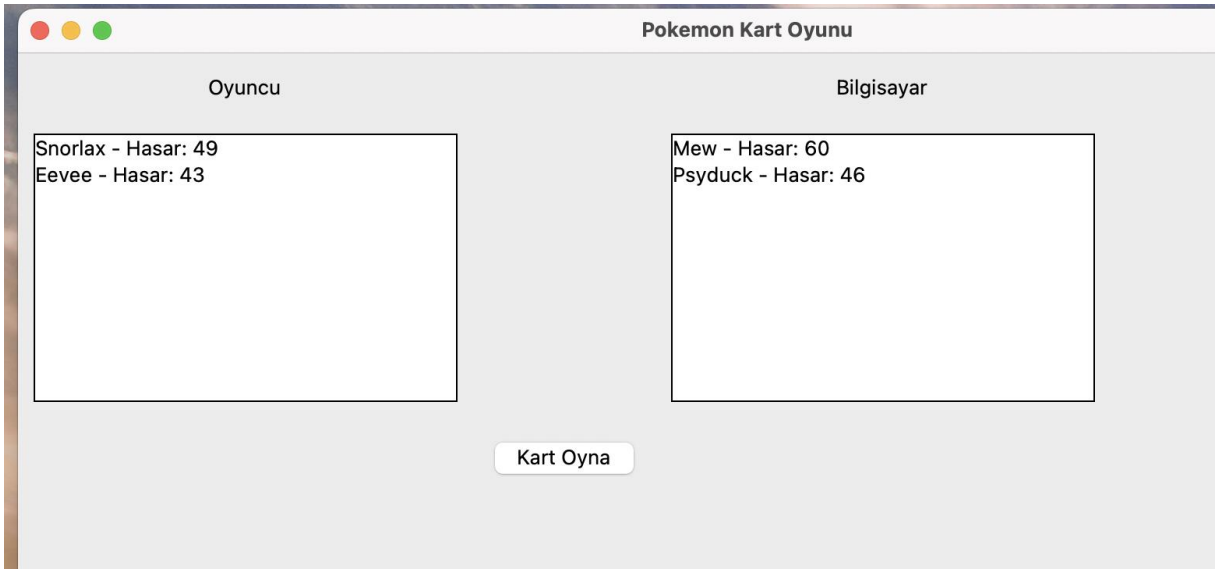



Oyunun 3. eli görselde gözüküyor.

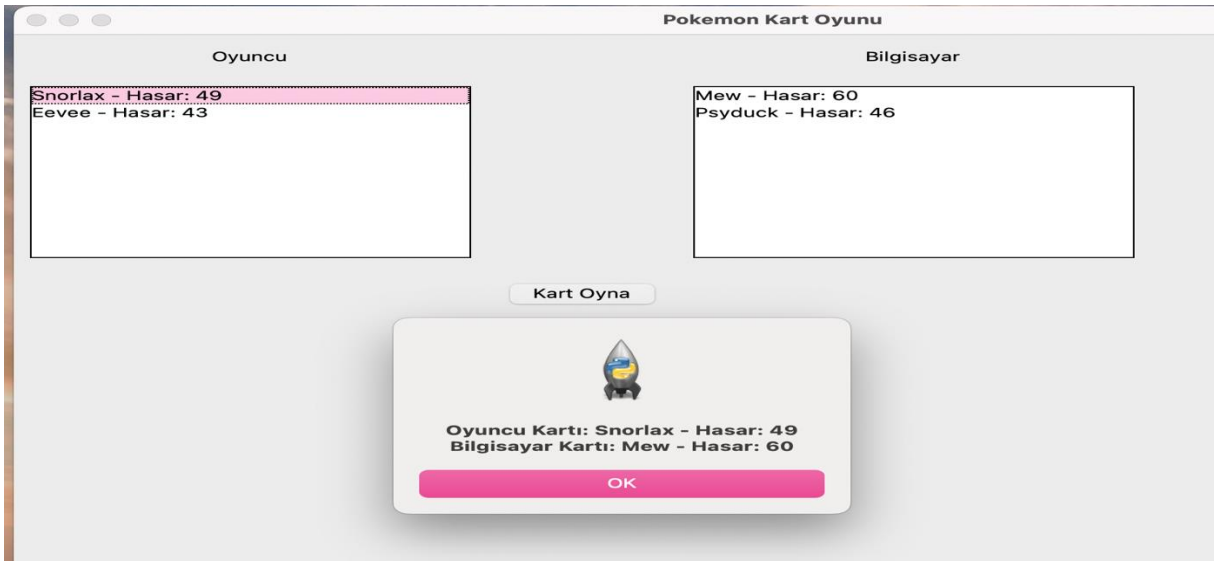



3 el oyun oynadıktan sonra elimizde kart kalmıyor ve ortadaki 4 karttan 2'şer tane seçiyoruz.

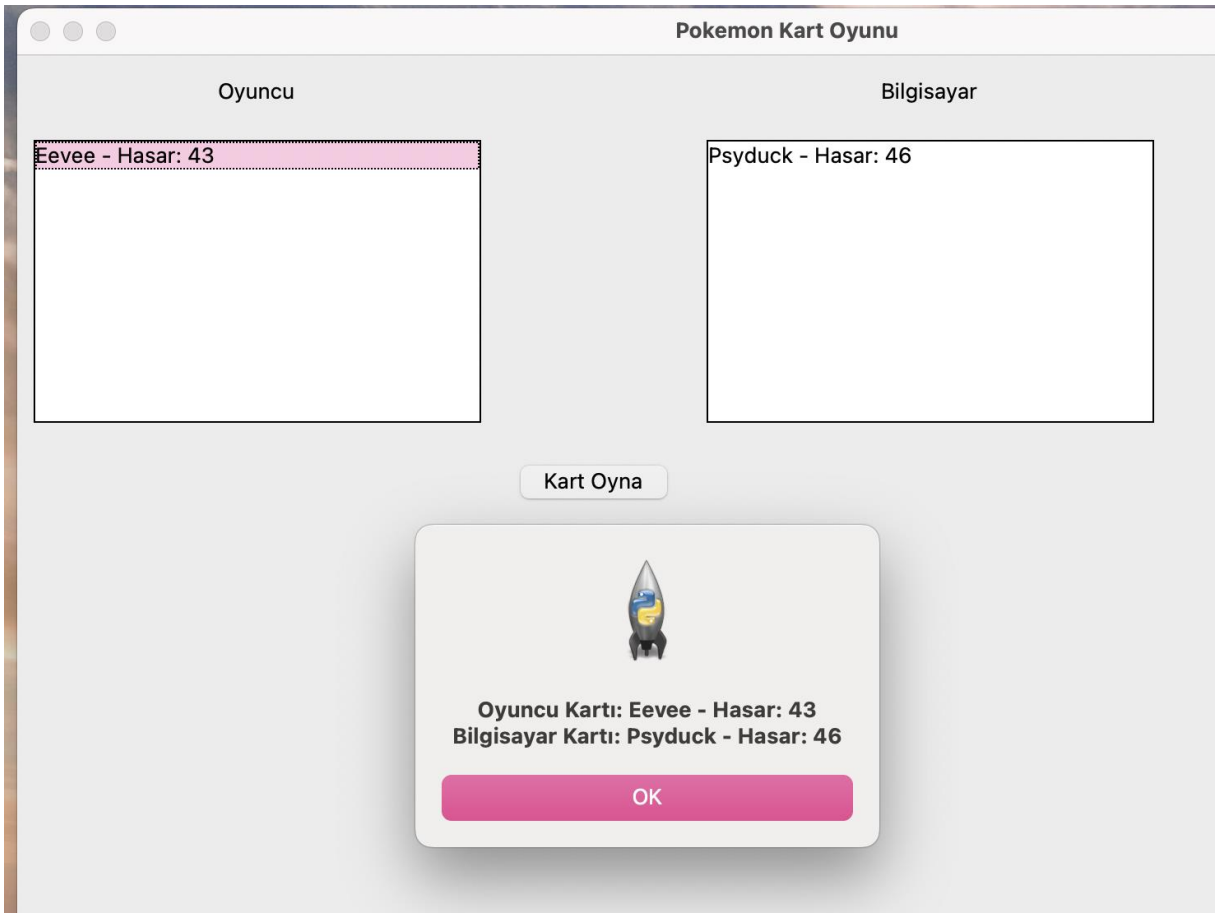
Yine aynı şekilde biz 1 kart seçince ardından bilgisayar da otomatik olarak 1 kart seçiyor.




Kartların seçilmiş olduğunu görüyoruz.



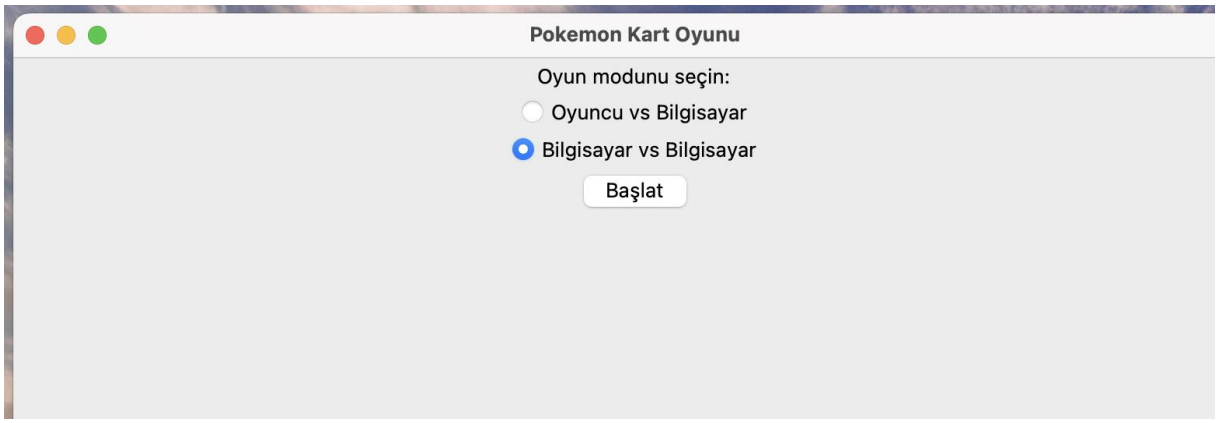

Tekrar kartları seçip oyunu oynuyoruz. Karşılaştırmalar yapılıyor.



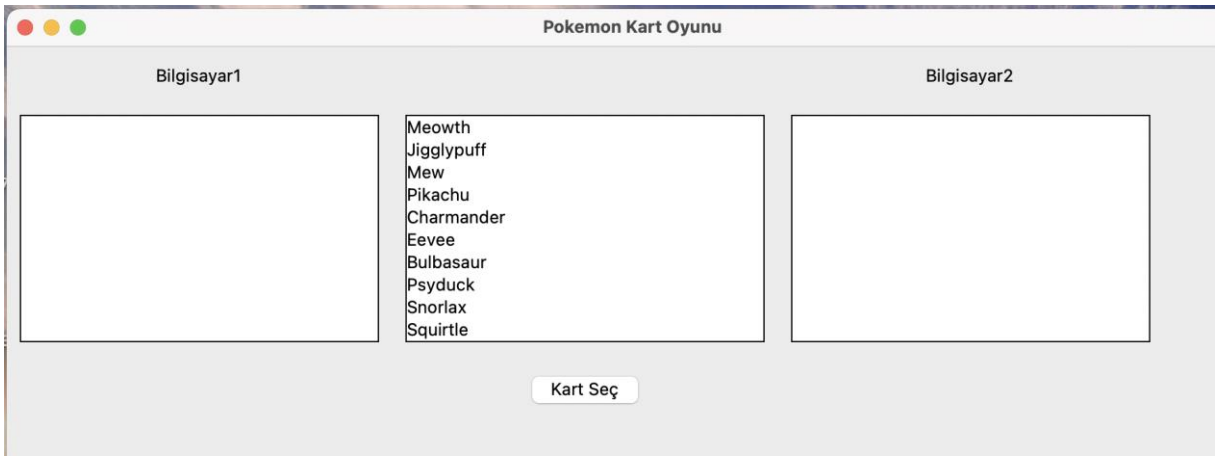

Oyunun son eli görselde gözüküyor.


Oyun sonunda kazanan oyuncu bu ekranda gösteriliyor.

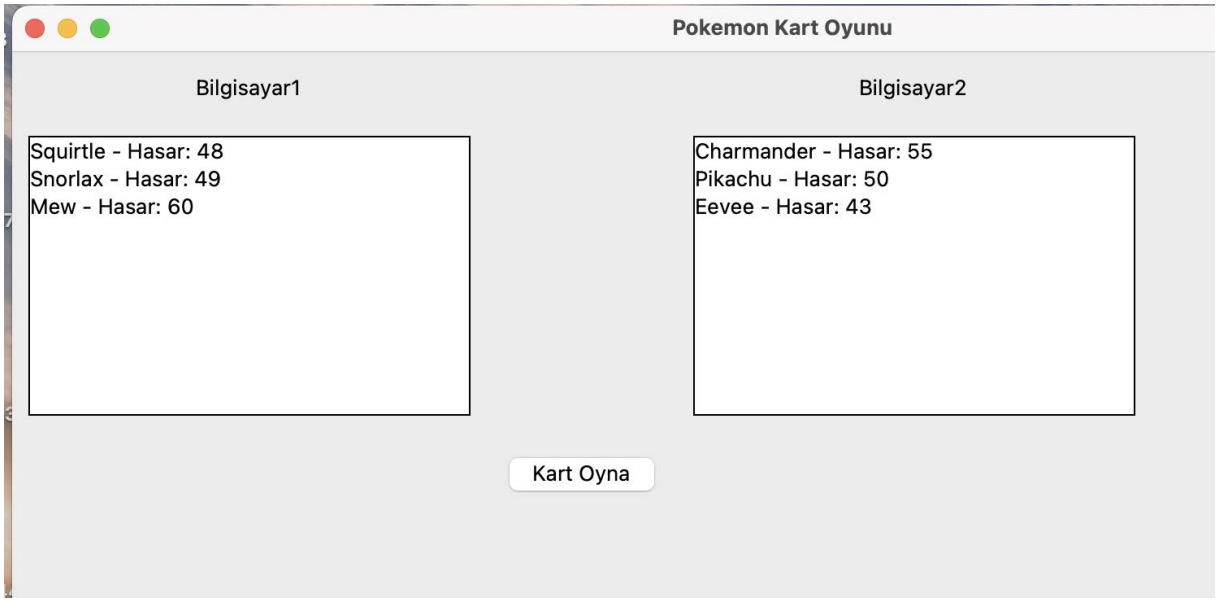
Şimdi Bilgisayar vs Bilgisayar senaryosuna göz atalım.



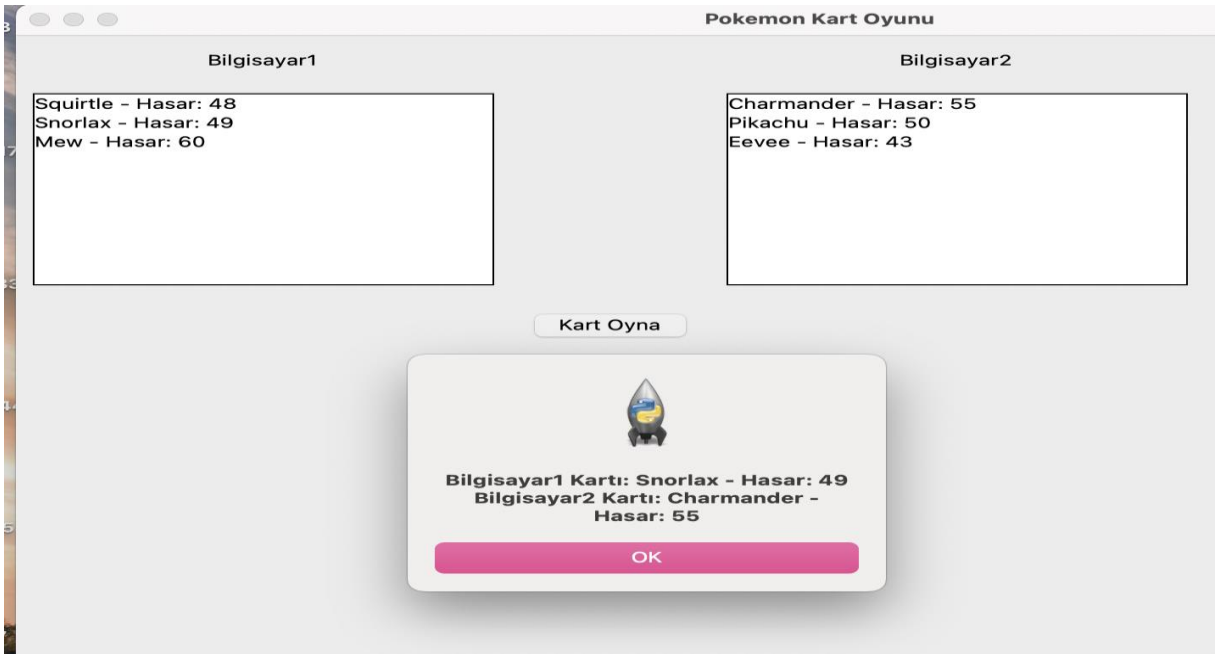

Seçimi yaptık.



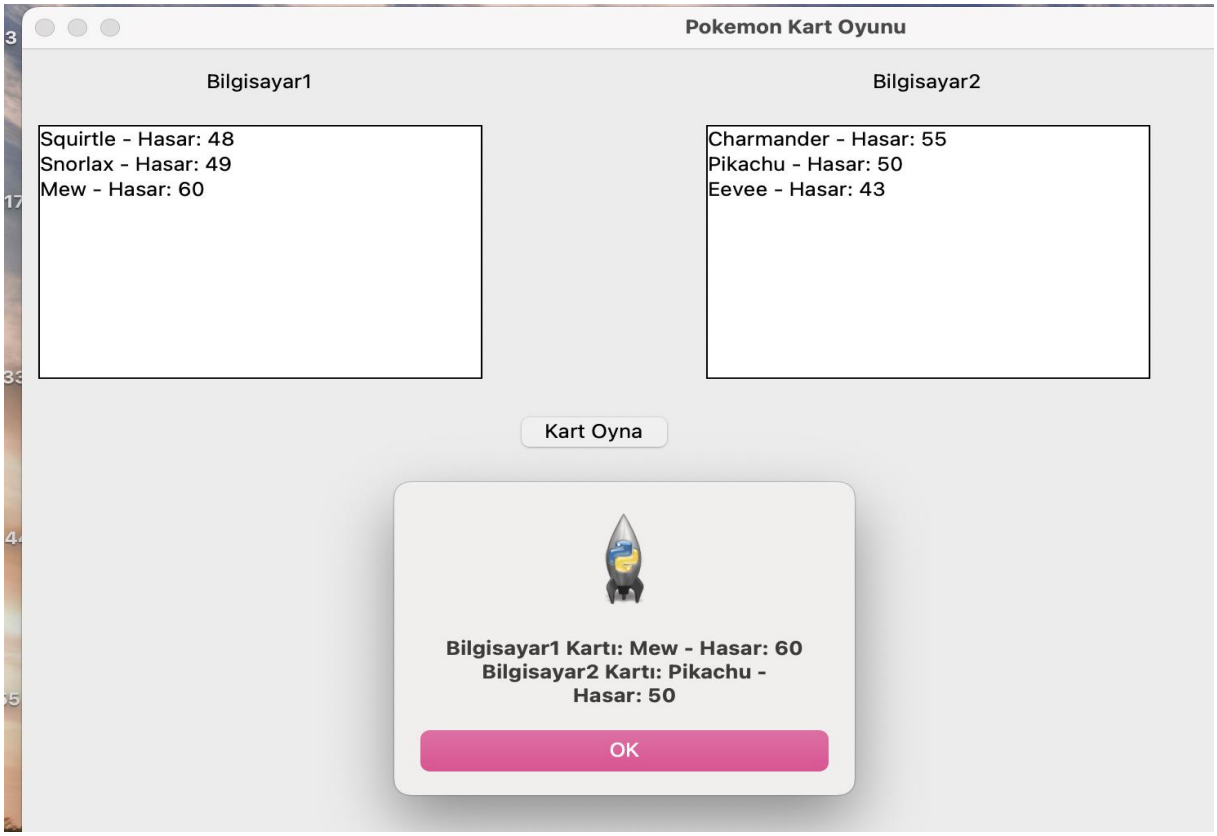

Ortada 10 adet kart bulunuyor. Kart seç dediğimizde kartlar otomatik olarak her iki oyuncuya da 3'er 3'er seçiliyor.

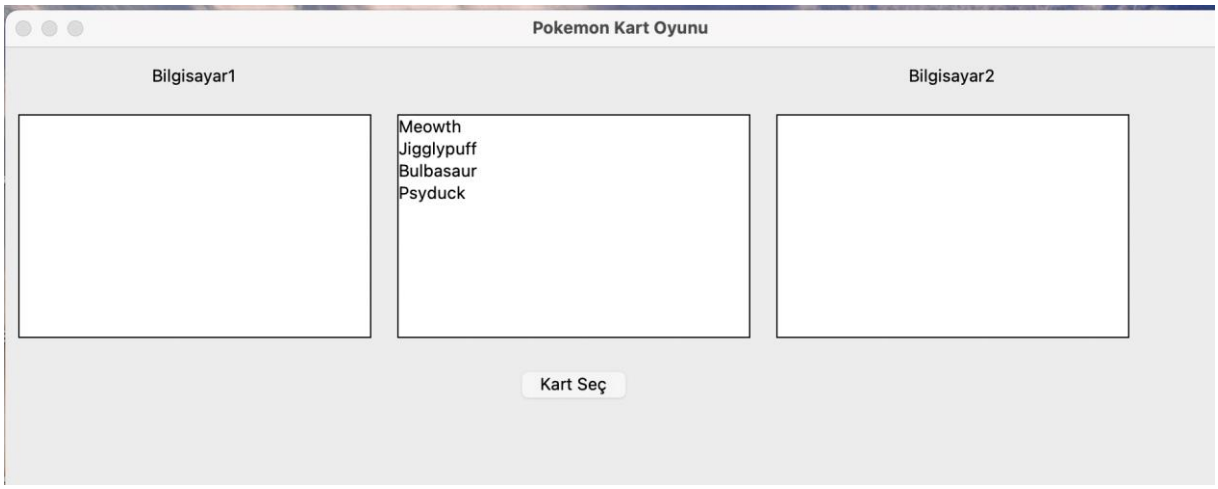



Kartlar seçilmiş durumda.

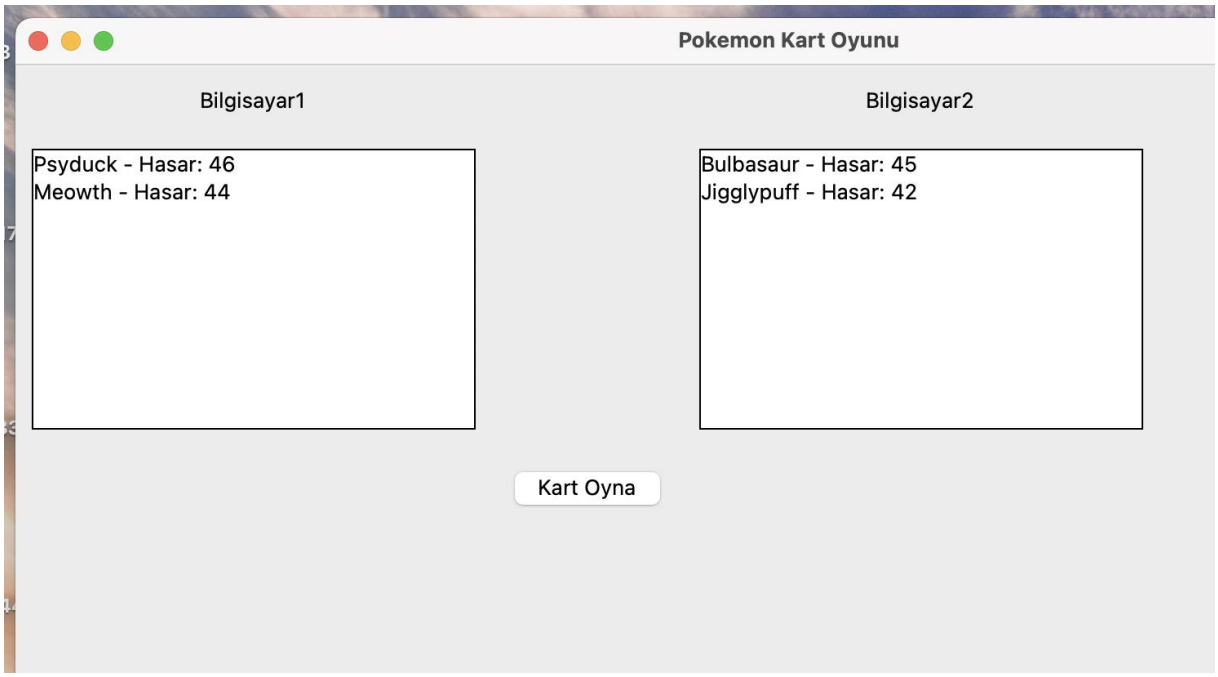



Kart oyna dediğimizde otomatik olarak kartlar öne sürülüyor ve kaşılaştırma yapılıyor.

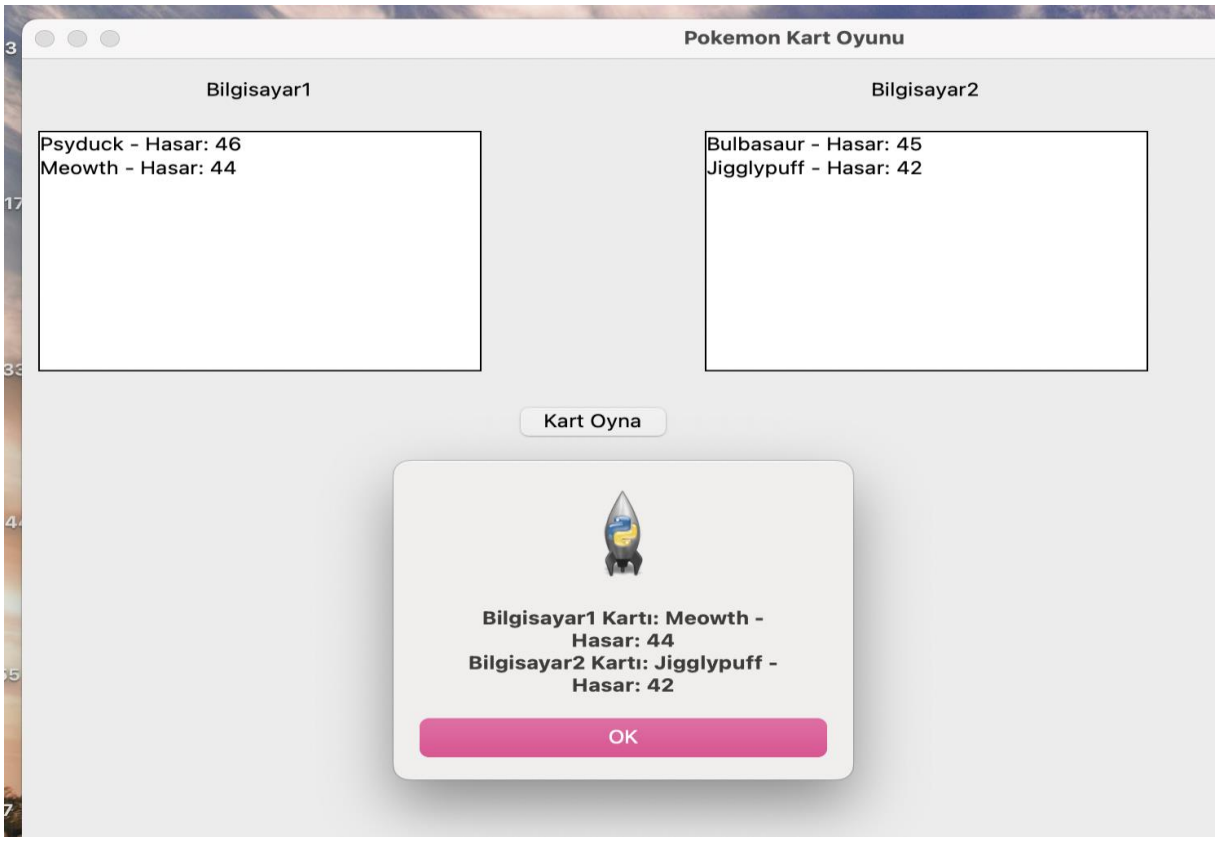




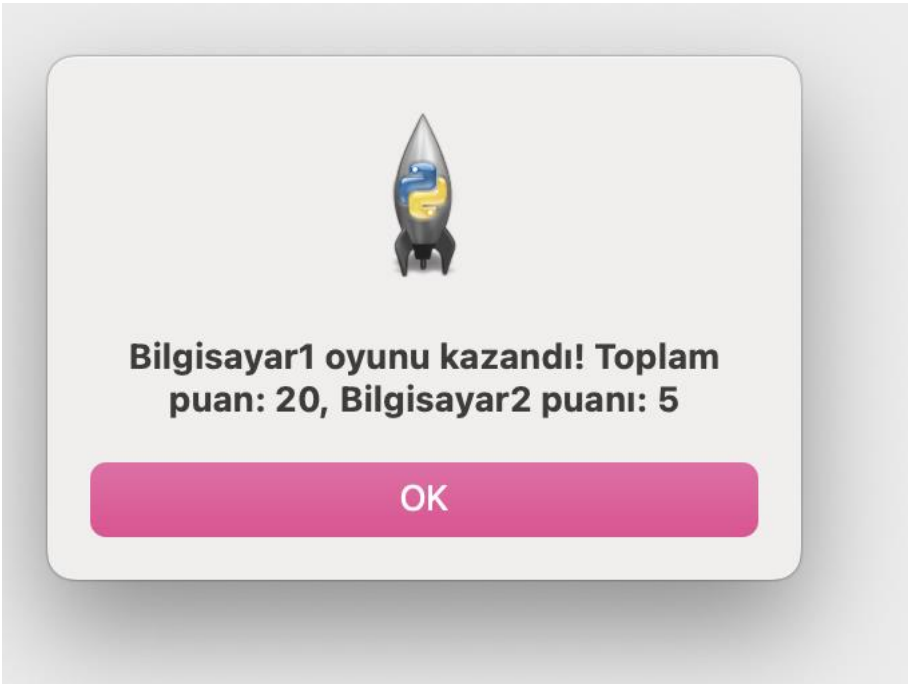

Ortada kalan 4 kart için de 2'şerli seçimler 'Kart Seç' butonuna tıkladığımızda otomatik olarak yapılıyor.




Son 4 kart da paylaştırıldı.




Oyun oynanıyor, kartlar karşılaştırılıyor.




Oyun sonunda puan durumu ve kazanan oyuncu ekranda gösteriliyor.

# 4. UYGULAMA

## 4.1 Kodlanan Bileşenlerin Açıklamaları

- **Card sınıfı:** Kart nesnelerini oluşturur.
- **Player sınıfı:** Oyuncu ve bilgisayar için kart seçim ve puanlama fonksiyonlarını içerir.
- **Game sınıfı:** Oyun mantığını ve kurallarını yönetir.
- **PokemonGameGUI sınıfı:** Kullanıcı arayüzünü ve oyun akışını kontrol eder.

Projede kullanılan tüm sınıflar ve özellikleri, nesne yönelimli programlama (OOP) prensipleri kullanılarak tasarlanmıştır. Projede Encapsulation, Inheritance, Polymorphism, ve Abstraction gibi OOP kavramları kullanılmıştır. Aşağıda bu kavramların her birinin nasıl kullanıldığı ve neden kullanıldığı açıklanmıştır.

### 1. Encapsulation (Kapsülleme)

Kapsülleme, sınıfın özelliklerini ve metotlarını gizleyerek dışarıdan doğrudan erişimi engellemeyi ve bu özelliklere/metotlara yalnızca belirli metotlar aracılığıyla erişim sağlamayı içerir. Kodumuzda, kapsülleme şu şekilde uygulanmıştır:

- Sınıf içi değişkenler (örneğin isim ve hasar özellikleri) private (__) olarak tanımlanmıştır.
- Bu değişkenlere erişim ve değişiklik yapmak için get ve set metotları tanımlanmıştır.

**Card Sınıfı**

```
class Card:
    def __init__(self, isim="", hasar=0):
        self.__isim = isim
        self.__hasar = hasar

    def get_isim(self):
        return self.__isim

    def set_isim(self, isim):
        self.__isim = isim

    def get_hasar(self):
        return self.__hasar

    def set_hasar(self, hasar):
        self.__hasar = hasar

    def __str__(self):
        return self.__isim
```

**Player Sınıfı**

```
class Player:
    def __init__(self, isim=""):
        self.__isim = isim
        self.__el = []
        self.__puan = 0

    def get_isim(self):
        return self.__isim

    def set_isim(self, isim):
        self.__isim = isim

    def get_el(self):
        return self.__el

    def set_el(self, el):
        self.__el = el

    def get_puan(self):
        return self.__puan

    def set_puan(self, puan):
        self.__puan = puan

    def kart_cek(self, kart):
        self.__el.append(kart)

    def kart_oyna(self, index):
        return self.__el.pop(index)

    def puan_ekle(self, puan):
        self.__puan += puan
```

### 2. Inheritance (Kalıtım)

Kalıtım, bir sınıfın başka bir sınıftan türetilerek onun özelliklerini ve metotlarını miras almasını sağlar. Bu projede, kalıtım örnekleri çok belirgin bir şekilde kullanılmamış olsa da, farklı oyuncu türlerini temsil eden sınıflar (örneğin ComputerPlayer ve HumanPlayer) oluşturularak Player sınıfından türetilebilir.

### 3. Polymorphism (Çok Biçimlilik)

Polymorphism, aynı isimdeki metodların farklı şekillerde davranmasını sağlar. Bu projede, polimorfizm örneği olarak kart_oyna ve kart_cek metodları, hem Player sınıfında hem de bu sınıftan türetilen sınıflarda farklı şekillerde kullanılabilir.

### 4. Abstraction (Soyutlama)

Soyutlama, gereksiz detayları gizleyerek kullanıcıya sadece gerekli olan bilgileri sunma işlevidir. Bu projede soyutlama, sınıf ve metot tanımları ile gerçekleştirilmiştir. Örneğin, Game sınıfı oyunun mantığını kapsarken, detayları gizler ve kullanıcıya sadece gerekli fonksiyonları sunar.

Sonuç olarak, projede OOP prensipleri kapsamında kullanılan Encapsulation, Inheritance, Polymorphism ve Abstraction yapıları, sınıfların ve metotların daha modüler, okunabilir ve sürdürülebilir olmasını sağlamıştır. Her sınıf için oluşturulan yapı, oyun mantığını ve akışını doğru bir şekilde yönetirken, kullanıcı arayüzünün ve oyun mekaniklerinin düzgün çalışmasını sağlar. Bu yapılar, yazılım geliştirme sürecinde kod tekrarını azaltmak, bakımı kolaylaştırmak ve kodun genişletilebilirliğini artırmak için kullanılmıştır.

### 4.2 Görev Dağılımı

Projenin her aşaması ortaklaşa yapılmıştır.

### 4.3 Karşılaşılan Zorluklar ve Çözüm Yöntemleri

- **Kart seçiminde index hataları:** Kullanıcı seçiminde doğru kartın seçilmesi için arayüzde ek kontroller yapıldı.
- **Arayüz güncellemeleri:** Seçim sonrası arayüzün doğru güncellenmesi için fonksiyonlar optimize edildi.
- **Oyuncu kartlarının doğru gösterilmesi:** Listboxların doğru güncellenmesi ve gösterilmesi sağlandı.

### 4.4 Proje İsterlerine Göre Eksik Yönler

**Tam olarak kodlanamayan işlevler:**

- Orta kart seçiminde yaşanan hataların düzeltilmesi.
- Tüm oyun akışının kullanıcı dostu bir şekilde sağlanması.

## 5. TEST VE DOĞRULAMA

### 5.1 Yazılımın Test Süreci

- **Test uygulaması:** Her fonksiyonun doğru çalışıp çalışmadığını kontrol edecek şekilde yazıldı.
- **Test fonksiyonları:** Kart seçimi, karşılaştırma ve puan hesaplamaları için ayrı test fonksiyonları geliştirildi.
- **Testlerin tekrarlanabilirliği:** Testler, kullanıcı ve bilgisayarın her turda doğru kartları seçtiğini ve doğru sonuçları aldığını doğrular.

## 5.2 Yazılımın Doğrulanması

**Test sonuçları:**

- **Doğru çalışan bileşenler:** Kart seçimi, hasar karşılaştırma ve puanlama.
- **Eksik veya hatalı çalışan bileşenler:** Orta kart seçiminde bazı durumlarda hatalar oluşabilir.