Okey Skor Hesaplayıcı

Digitoy Games – Okey Assignment

Cem Tezgelen

Backend Developer  cemtezgelenn@gmail.com

İçindekiler

[1. Giriş 2](#_Toc131976352)

[1.1. Oyun Kuralları 2](#_Toc131976353)

[1.2. Programlama Dili 2](#_Toc131976354)

[1.3. Çalışma Ortamı (IDE) 2](#_Toc131976355)

[2. Proje Amacı 2](#_Toc131976356)

[2.1. Oyunun Hazırlanması 2](#_Toc131976357)

[2.2. Oyuncuların Skorlarının Hesaplanması 3](#_Toc131976358)

[2.2.1. Çift Dizilim 3](#_Toc131976359)

[2.2.2. Seri Dizilim 3](#_Toc131976360)

[2.2.3. Set Dizilimi 3](#_Toc131976361)

[2.2.4. Skor Hesaplama Algoritması 3](#_Toc131976362)

[3. Veri Modeli 3](#_Toc131976363)

[3.1. OkeyGame 4](#_Toc131976364)

[3.2. Player 4](#_Toc131976365)

[3.3. Tile 4](#_Toc131976366)

[3.4. Deck 5](#_Toc131976367)

[3.5. ConsoleColors 5](#_Toc131976368)

[3.6. Utility 6](#_Toc131976369)

[4. Toggl Track Sonuçları 7](#_Toc131976370)

[5. Sonuç 8](#_Toc131976371)

# Giriş

Bu raporda, Digitoy Games tarafından kodlanması talep edilen ‘Okey’ oyunu kodlanmıştır. Oyunun aktif olarak oyuncular tarafından oynanması beklenmemektedir. Aşağıda oyun hakkında teorik ve teknik detaylara yer verilmiştir.

## Oyun Kuralları

Okey oyunu, 4 Farklı renkte, 13 farklı değere sahip toplam 52 taş ile oynanan, Türk kültüründe yer edinmiş bir masa oyunudur. Bahsedilen 52 taş haricinde 1 adet Sahte Okey taşı da bulunmaktadır.

Oyun başlangıcında her oyuncuya 14, ilk başlayacak oyuncuya da fazladan 1 taş rastgele olarak dağıtılır. Oyuncular ellerindeki taşlar çift, seri veya set olarak dizerek elindeki bütün taşları kullanmaya çalışırlar. Elindeki bütün taşları uygun bir per yapan oyuncu, fazlalık bir taşını açarak oyunu kazanır.

## Programlama Dili

Bu proje, Java dilinde kodlanmıştır. Konsol ekranı üzerinde kullanıcıya programın ve hesaplamaların doğruluğunu kanıtlayan çıktılar verilmiştir. GUI (Graphical User Interface) ihtiyacı duyulmamıştır, talep halinde JavaFX, Java Swing veya talep edilen kütüphane kullanılarak kolayca entegrasyon sağlanabilecektir.

## Çalışma Ortamı (IDE)

Proje, Eclipse IDE’de kodlanmıştır. Eclipse, otomatik kaynak oluşturma, checkstyle desteği ve geniş eklenti kütüphanesi bulundurması sebebiyle tercih edilmiştir.

# Proje Amacı

Bu projede, okey oyununun oynatılması amaçlanmamıştır. Projede, bir Okey oyununda taşların 4 oyuncuya dağıtıldıktan sonra, oyuncuların mevcut skorlarının hesaplanması hedeflenmiştir.

Skor hesaplama işlemi Çift, Seri veya Per olarak taşların sıralanmasıyla sağlanır. Algoritma detaylarına aşağıda yer verilmiştir.

## Oyunun Hazırlanması

Oyun başlatılmadan önce, 4 oyuncunun hazırlanarak ellerinin dağıtılması gerekmektedir. Oyuncu sayısı, oyuncu başına taş sayısı gibi sabit değişkenleri ‘final’ etiketi ile tanımladıktan sonra döngüler ve oluşturulan bazı metotlar aracılığıyla oyuncuların, sınıf içerisindeki bir listeye eklenmesi ve ellerinin dağıtılması sağlanmıştır. Her oyun başlangıcında, oyuna başlayacak oyuncu rastgele olarak belirlenir ve bu oyuncuya 1 taş fazla dağıtılır. Taş dağıtma işleminden hemen önce, gösterge taşı seçilir ve bu gösterge taşının mutlaka sahte okey olmaması garantilenir. Okey taşı da, gösterge taşının 1 fazlasıdır ve eğer gösterge taşı 13 ise, okey taşı otomatik olarak ‘1’ olarak belirlenir. Ardından her oyuncuya desteden rastgele 14(+1) taş dağıtılır ve oyun başlamaya hazır hale getirilir.

## Oyuncuların Skorlarının Hesaplanması

Projede beklenen, seri veya çift dizilime göre oyuncunun elindeki skorun hesaplanmasıydı. Ancak ben bunun yeterli olmayacağını düşünerek ‘Set’ hesaplamasını da dahil ettim.

## Çift Dizilim

Oyuncunun ıstakasında aynı renge ve aynı değere sahip 2 taş bulunuyorsa, bu bir çifttir.

## Seri Dizilim

Oyuncunun ıstakasında, aynı renge sahip ardışık 3 taş bulunuyorsa, bu bir seri dizilimdir. 3’den fazla taş da seri dizilime eklenebilmektedir. 13’ün ardında 1 değerindeki taş eklenerek seri dizilime dahil edilebilir.

## Set Dizilimi

Oyuncunun ıstakasında, farklı renge ve aynı değere sahip en az 3 taş bulunuyorsa, bu bir set dizilimdir. Set dizilimler en az 3, en fazla 4 taştan oluşabilir.

## Skor Hesaplama Algoritması

Okey oyununda, bir dizilimde kullanılan bir taş, başka bir dizilimde kullanılamamaktadır. Bu sebeple oyuncunun elde edebileceği en optimal skor, her bir dizilimin sırayla gerçekleştirilerek hesaplanması gerekmektedir. Bu durumda basit bir permütasyon hesabı ile 6 farklı skor hesaplama işlemi gerçekleştirilmesi gerektiğini anlıyoruz. Aşağıdaki tabloda, muhtemel sıralamalara yer verilmiştir.

|  |  |  |
| --- | --- | --- |
| 1. | 2. | 3. |
| Çift | Seri | Set |
| Çift | Set | Seri |
| Seri | Çift | Set |
| Seri | Set | Çift |
| Set | Seri | Çift |
| Set | Çift | Seri |

Her bir dizilim için, ayrı birer metot yer alıyor. Her bir skor hesaplaması sonrasında, taşların kullanım durumu güncellenerek aynı taşların kullanılmaması sağlanır. Eğer bir skor hesaplama seti son bulduysa, kullanılan taşlar sıfırlanarak, bir sonraki skor hesaplama grubu için hazır hale getirilir.

Her bir hesaplama grubunun hesaplaması sona erdiğinde, en yüksek skora sahip grubun skoru geçici bir dizide tutulur. Algoritma bütün olasılıkları hesapladıktan sonra en yüksek skoru oyuncunun skoru olarak belirler.

# Veri Modeli

Aşağıda, projede kullanılan her bir sınıfa ait özellikler ve yöntemlere(metot) yer verilmiştir. Main sınıfı dahil edilmemiştir.

## OkeyGame

**Özellikler**

|  |  |
| --- | --- |
| **Özellik İsmi** | **Veri Tipi** |
| PLAYER\_COUNT | Int |
| TILE\_COUNT | Int |
| TILE\_PER\_HAND | Int |
| indicatorTile | Tile |
| okeyTile | Tile |
| deck | Deck |
| players | ArrayList<Player> |

**Yöntemler**

|  |  |
| --- | --- |
| **Yöntem Adı** | **Dönüş Tipi** |
| OkeyGame() | - |
| startGame() | void |
| calculatePlayerScores() | void |
| chooseOkeyTile() | void |
| createPlayers() | void |
| drawHands() | void |
| printBestPlayers() | void |
| printGameInformation() | void |

## Player

**Özellikler**

|  |  |
| --- | --- |
| **Özellik İsmi** | **Veri Tipi** |
| hand | ArrayList<Tile> |
| playerNumber | Int |
| score | Int |

**Yöntemler**

|  |  |
| --- | --- |
| **Yöntem Adı** | **Dönüş Tipi** |
| Player(int) | - |
| addTile(Tile) | void |
| getHand() | ArrayList<Tile> |
| getPlayerNumber() | Int |
| setHand(ArrayList<Tile>) | Void |
| setScore(int) | void |
| toString() | String |

## Tile

**Özellikler**

|  |  |
| --- | --- |
| **Özellik İsmi** | **Veri Tipi** |
| Value | Int |
| Color | String |
| Used | Boolean |
| fakeOkey | Boolean |
| isWildcard | Boolean |

**Yöntemler**

|  |  |
| --- | --- |
| **Yöntem Adı** | **Dönüş Tipi** |
| Tile(int, String) | - |
| compareTo(Tile) | Int |
| equals(Object) | Boolean |
| getColor() | String |
| getValue() | Int |
| hashCode() | Int |
| isFakeOkey() | Boolean |
| isUsed() | Boolean |
| isWildcard() | Boolean |
| setFakeOkey(boolean) | Void |
| setUsed(boolean) | Void |
| setWildcard(boolean) | Void |
| toString() | String |

## Deck

**Özellikler**

|  |  |
| --- | --- |
| **Özellik İsmi** | **Veri Tipi** |
| Tiles | ArrayList<Tile> |

**Yöntemler**

|  |  |
| --- | --- |
| **Yöntem Adı** | **Dönüş Tipi** |
| Deck() | - |
| createDeck() | Void |
| drawNonFakeTile() | Tile |
| drawTile() | Tile |
| shuffleDeck() | Void |

## ConsoleColors

**Özellikler**

|  |  |
| --- | --- |
| **Özellik İsmi** | **Veri Tipi** |
| ANSI\_BLACK | String |
| ANSI\_BLUE | String |
| ANSI\_CYAN | String |
| ANSI\_GREEN | String |
| ANSI\_PURPLE | String |
| ANSI\_RED | String |
| ANSI\_RESET | String |
| ANSI\_WHITE | String |
| ANSI\_YELLOW | String |

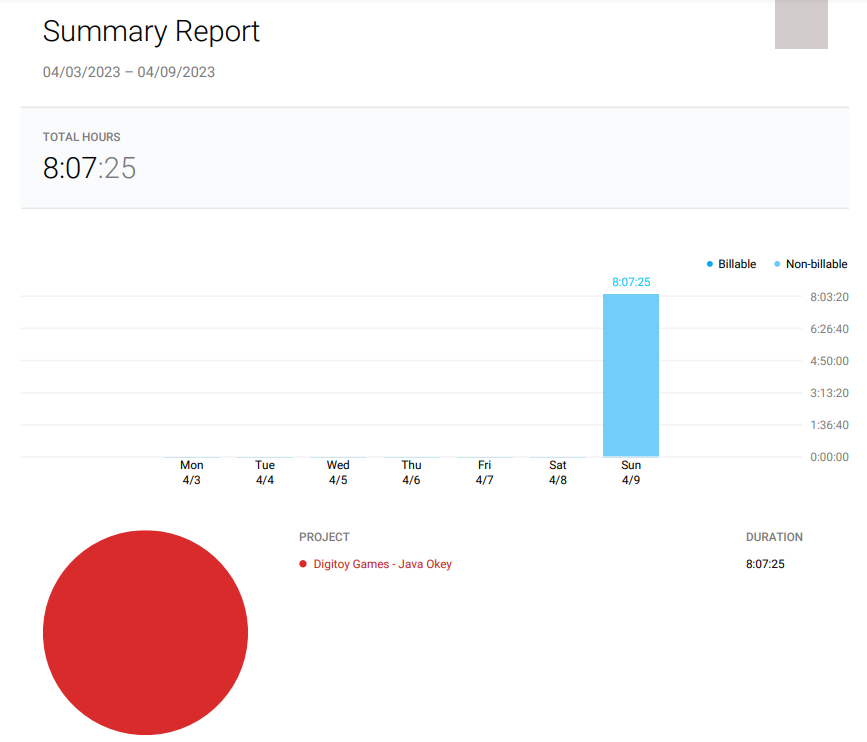
## Utility

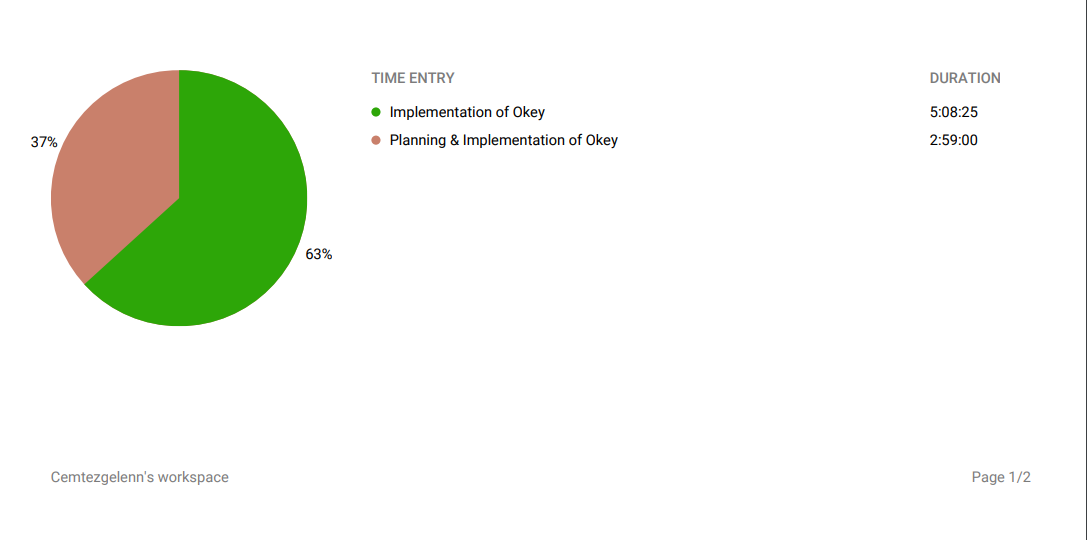
**Yöntemler**

|  |  |
| --- | --- |
| **Yöntem Adı** | **Dönüş Tipi** |
| calculateScore(List<Tile>, int[]) | Int |
| countPairs(List<Tile>) | Int |
| countSeries(List<Tile>) | Int |
| countSets(List<Tile>) | Int |
| groupByColor(List<Tile>) | HashMap<String, List<Tile>> |
| nextPermutation(int[]) | Boolean |
| processHand(Player, Tile) | void |

# Toggl Track Sonuçları

‘Toggl Track’ isimli, Clockify benzeri ücretsiz bir platform üzerinden proje üzerinde harcadığım süreyi kayıt altına aldım.

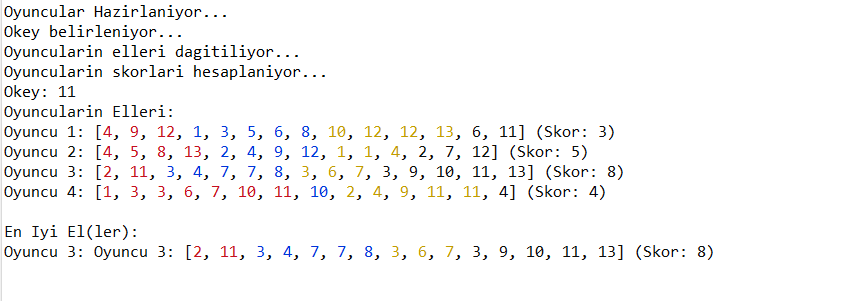




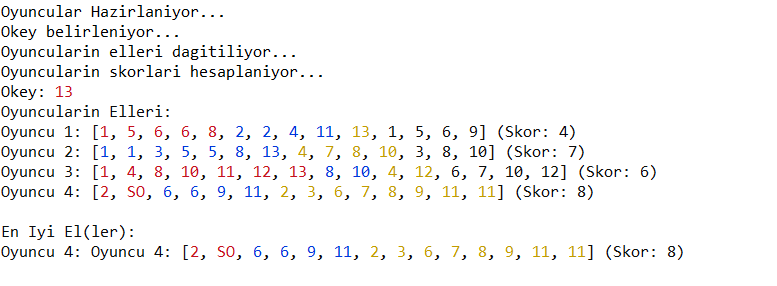
# Sonuç

Uzun testler, hata ayıklamaları ve araştırmalar sonucunda proje doğru şekilde çalıştırılmıştır. Aşağıda, bir kaç örnek çalışmaya yer verilmiştir.

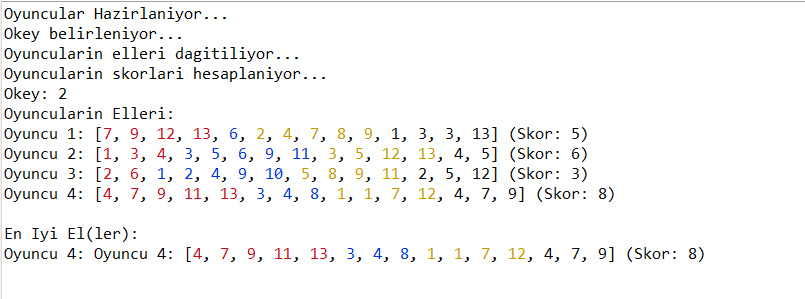
* Çalışma 1



* Çalışma 2



* Çalışma 3



* Çalışma 4

