Project 3:
Zar Oyunu

Final notu olarak kullanılacaktır.

Gereksinimler

- Primitive Data Types
- Loops
- Arrays
- Classes
- Interfaces

Öğrenme Hedefleri

- Using interfaces
- Using simulation to test alternative strategies

Giriş

Bu ödevde zar oyunu isminde multiplayer bir oyun yapmanız istenmektedir.

Oyunun kuralı: Her oyuncu 100 puan almak için yarışmaktadır. Her turda bir oyuncu zarı 1 gelene kadar yada kendisi bırakana kadar atabilmektedir:

- 1 geldiğinde bu turdan sıfır almakta
- kendisi bıraktığında bu turdan almış olduğu skor toplam skoruna eklenmektedir.

Yani oyuncu oynarken iki şekilde karar verebilmektedir:

1. **roll** - bu durumda zar tekrar atılmakta
    - **1**: geldiğinde oyuncu bu turdan hiç puan almamış oluyor ve sıra rakibine geçiyor.
    - **2 - 6**: Oyuncunun tur puanına atılan sayı ekleniyor ve oyuncu zar atmaya devam edebiliyor.
2. **hold** - Oyuncunun bu turda almış olduğu skor toplam skoruna ekleniyor ve sıra rakibine geçiyor.

Problemin Tanımı

Bu oyun oynanırken farklı stratejiler denenebilir. Sizden bu oyun için, farklı stratejilerin bilgisayar tarafından denenebileceği bir object oriented implementasyon yapmanız istenmektedir.

Stratejiler

Hold at 20 or goal (20 de yada kazanınca bırak): Her turda

- ya 1 gelene kadar
- yada tur skoru 20 yada büyük olana kadar
- yada alınan tur skoru ile toplam skorun toplamı 100 yada büyük olana kadar atmaya devam et

Hold at 25 or goal (25 de yada kazanınca bırak): Her turda

- ya 1 gelene kadar
- yada tur skoru **25** yada büyük olana kadar
- yada alınan tur skoru ile toplam skorun toplamı 100 yada büyük olana kadar atmaya devam et

Keep Pace and End Race (son düzlükte hızlan): oyuncunun toplam skoru ***`tsi`*** *olsun,* ***`maxTS`*** *de tüm oyuncular arasındaki maksimum toplam skor olsun.*

- *Eğer* ***`tsi`*** >= 71 yada ***`maxTS`*** >= 71, toplam 100'ü bulana (yani oyunu kazanana) kadar atmaya devam et.
- Diğer durumlarda
    - tur skoru >= 21 + round((***`maxTS`*** - ***`tsi`***) / 8), olana kadar
    - yada tur skoru ile toplam skorun toplamı 100 yada büyük olana kadar atmaya devam et.

Yapılacak class ve interfaceler

Notlandırmayı kolaylaştırmak için asgari aşağıdaki class ve interfaceleri kullanınız. Tüm class ve interfaceleri **public** olarak tanımlayınız.

class Die

Genel 6 yüzlü zarı implement eden class
Bu class içerisindeki `nextRoll()` methodu zarın atılması sonucu gelen sayıyı vermektedir.

Metod:

`int nextRoll()`: (1-6) aralığında rastgele sayı return eder. Bu metod da Random sınıfından `nextInt()` metodunu kullanabilirsiniz.

interface DiePlayer

Bu genel oyuncuyu tanımlamaktadır.

Metod:

```
boolean isRolling(int myScore, int turnTotal, int maxScore, int rolls)
```

- `myScore:` şu anki oyuncunun skoru (score of the current player)
- `turnTotal:` tbu turda alınan skor (he accumulated score in the current turn)
- `maxScore:` tüm oyuncular arasında maksimum skor (the maximum score among all players)
- `rolls:` oyuncunun bu turda zarı kaç defa attığını belirtiyor (number of times the player has rolled in the current turn)

Bu metod eğer oyuncu atmaya devam edecekse return değeri olarak true veriyor. Atmaya devam edip etmediğini bu metodu implement eden classda kullanılan strateji ile belirlenecek.

class UserDiePlayer

Bu class **`DiePlayer`** interface'ni implement etmekte ve oyuncunun consoledan oyunu oynamasına yardımcı olmaktadır.
Kullanıcının oyunu oynayabilmesi için kullanıcıya tur toplamını göstermekte ve **"Roll or Hold?"** şeklinde sorarak kullanıcının kararını almaktadır:

- Eğer empty string girildiyse (yani hiç bir şey girilmeden entera basıldıysa) devam
- Diğer durumlarda hold (yani bırak)

Eng: Prompt the user with a message that includes the "turn total" and the option to "Roll or Hold?" An empty input (i.e., pressing Enter) indicates that the user wishes to roll. Any entered line of non-zero length indicates that the user wishes to hold.

class HoldAt20DiePlayer

Bu class **`DiePlayer`** interface'ni implement etmekte ve oyunu "**Hold at 20 or goal"** stratejisine göre devam ettirmektedir.

class HoldAt25DiePlayer

Bu class **`DiePlayer`** interface'ni implement etmekte ve oyunu "**Hold at 25 or goal"** stratejisine göre devam ettirmektedir.

class RaceToGoalDiePlayer

Bu class **`DiePlayer`** interface'ni implement etmekte ve oyunu "**Keep Pace and End Race"** stratejisine göre devam ettirmektedir.

class MyDiePlayer

Bu class **`DiePlayer`** interface'ni implement etmekte ve kendi belirlediğiniz stratejiye göre oynamaktadır. İstediğiniz stratejiyi seçebilirsiniz.

class DieGame

Bu class oyunun temel kurallarını gerçekleştirmektedir.

Fields (data members):

```
public static final int GOAL_SCORE = 100; // the goal to be
reached,kazanmak icin alinacak skor

private DiePlayer[] players; // Array of Players
```

İhtiyacınız oldukça private data memberlar ekleyebilirsiniz

Constructor:

`public DieGame(DiePlayer[] players):` DieGame'i verilen oyuncular dizisi ile birlikte başlatmaktadır. Her oyuncunun başlangıç skoru 0 dır.

Diğer metodlar:

`int playTurn(DiePlayer p, int playerNum):` Oyuncunun 1 turdaki oyununu gerçekleştirmektedir. *players[] arrayde* indexi **`playerNum`** `olan` oyuncu **`p`** `için` oyunu bir tur oynamakta ve oyuncunun aldığı tur skorunu return etmektedir.

Eng: simulates one turn of the DiePlayer p at index playerNum in the players[] array). This method returns the turn total for the player p.

`public int play():` Oyunun 1 turunu tüm oyuncular için oynamaktadır (yukarıdaki metodu tüm oyuncular için çağırmanız gerekiyor): Yani tüm oyuncular birer defa oynamaktadır. Bu metod kazanan oyuncunun dizideki indexini return etmektedir.

Eng: Plays an instance of the Die game with the players. This method returns the index in the array of the winning player.

Yine şu metodları implement etmeniz faydalı olabilir:

> **`int getMaxScore()`:** Oyuncular arasındaki maximum skoru return etmekte
> *eng: returns the maximum score among all players.*
>
> **`void printScores()`:** Tüm oyuncuların skorlarını yazdırmaktadır. Bunu kullanarak her tur sonunda bunu kullanarak yazdığınız kodun doğruluğunu kontrol edebilirsiniz.
> *eng:prints the scores of all the players. You may use this method to print scores of all players at the end of each players turn to check your implementation.*

class PlayDie

Bu class main metodunu içermektedir. Bu metod içerisinde farklı şekillerde oyunlar dizayn edebilirsiniz. Mesela bir kaç tane DiePlayer interfaceni implement eden herhangi bir classdan player oluşturursunuz, bir tane UserPlayer oluşturursunuz. Böylelikle user bilgisayara karşı oynamış olur.
Burada farklı stratejileri deneyerek hangi stratejinin daha iyi çalıştığını belirleyiniz. Bunun için mesela

- 4 tane farklı stratejilerde oynayan oyuncu oluşturunuz
- ve bu oyuncuları bir birlerine karşı oynatıp kazananı not ediniz.
- Sonra bunu 1000 defa tekrarlayıp hangi stratejinin kaç defa kazandığını belirleyiniz.

Sonuç olarak statistics.txt dosyası içerisine

- Herbir stratejinin kazanma oranlarını
- Hangi stratejinin neden en iyi strateji olduğunu
  - Yine `MyDiePlayer` class ile oluşturmuş olduğunuz kendi stratejinizin verilenlerden daha mı iyi yada kötü olduğunu
- Ve diğer yorumlarınızı giriniz.

Teslim

`Classroom üzerinden dosyaları ziplemeden ayrı ayrı giriniz:`

1. **`Die.java`**
2. **`DiePlayer.java`**
3. **`UserDiePlayer.java`**
4. **`HoldAt20DiePlayer.java`**
5. **`HoldAt25DiePlayer.java`**
6. **`RaceToGoalDiePlayer.java`**
7. **`MyDiePlayaer.java`**

8. **DieGame.java**
9. **PlayDie.java**
10. **statistics.txt**

Değerlendirme

- 10% on coding style
- 5% statistics.txt
- 5% for the Die class
- 50% for DiePlayer strategies (i.e. 10% for each class implementing the DiePlayer interface)
- 30% for the DieGame class