SAKARYA UYGULAMALI BİLİMLER ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

NESNEYE YÖNELİK PROGRAMLAMA

PROJE RAPORU

----Hazırlayanlar:----
Fatma YAŞAR (23010903055)
Sümeyye GÜL (23010903049)

Ödev Konusu: Bir Oyun Tasarımı ve Geliştirilmesi

1. Oyun Tanımı ve Temel Kurallar

Oluşturduğumuz xox oyunumuz, temel nesne tabanlı programlama (NTP) prensiplerini kullanarak tasarlandı ve geliştirildi. Bu oyun, iki oyuncu arasında veya bir oyuncu ile bilgisayar arasında oynanabilen bir strateji oyunudur. Oyun modları olarak "İnsan vs İnsan" veya "İnsan vs Bilgisayar" seçenekleri sunulmaktadır. Oyun, oyuncuların genellikle tahtada hamle yaparak "X" ve "O" sembollerini yerleştirmelerine dayanır. Amaç, oyuncuların sırayla tahtaya hamle yaparak belirli kurallara göre kazananı belirlemektir. Bir oyuncu üç sembollü yatay, dikey veya çapraz olarak hizalayarak oyunu kazanır. Oyun, klasik "X ve O" tarzı bir oyun olarak tasarlanmıştır ve belirli bir düzenin oluşturulmasıyla sonlanır.

1.1 Oyun Tasarımı

Proje, Nesne Tabanlı Programlama (OOP) sistemlerine dayalı olarak tasarlanmıştır. Bu yapıyı oluştururken şu esaslara dikkat edildi:

Soyutlama : Oyunun temel resimleri soyutlayan sınıflar kullanılmıştır. OyunNesnesi sınıfı, soyut sınıf olarak kullanılarak yalnızca önemli fonksiyonlar ortaya konmuş, implementasyon detayları alt sınıflara bırakılmış.

Kalıtım : Oyuncular, bilgisayar oyuncusu gibi türetilmiş sınıflarla modellenmiştir.

Polimorfizm : Oyun görüntüleri (oyuncular ve tahtalar), sanal işlevlerle yönetilmiştir. display() ve hamleYap() gibi metodlar, alt sınıflarda farklı şekilde çalışarak polimorfizmi göstermektedir.

Enkapsülasyon : Tahta ve oyuncu sınıflarında, her nesnenin kendi içinde saklanmıştır. Özellikler private olarak korunmuş ve sadece public metodlarla erişilmiş.

Oyun, İnsan vs İnsan veya İnsan vs Bilgisayar olmak üzere iki oyun modunda çalışır. Oyuncular, konsol üzerinden hamle yapar, geçerli bir hamle yapıldıktan sonra tahta ekran basılır ve oyun sonuna kadar devam eder.

1.2 Proje Yapısı ve Kullanılan Sınıflar

Projede kullanılan belirli sınıflar şunlardır:

OyunNesnesi (Soyut sınıf ve Tahta sınıfı icin temel sınıf):

Tüm oyunların kalıtımında yer alan temel sınıftır. Burada, tahtayı yazdırma işlemi için display()fonksiyon soyut olarak ayarlanabilir.

Oyuncu (Temel sınıf):

Oyuncu sınıfı, oyuncunun adı, skoru ve yaptığı hamleleri içerir. Ayrıca oyuncunun hamlesini alma, skoru artırma ve kaydetme gibi işlemler. hamleYap(), setAd(), setSkor(), ve kaydetVeriTabani() gibi metotlar içerir.

BilgisayarOyuncu (Oyuncu sınıfından türetilmiş):

Bilgisayar oyuncusunun hamlesi rastgele olarak yapılır.

Tahta (OyunNesnesi sınıfından türetilmiş):

3x3'lük bir tahta nesnesi içerir ve tahtadaki sütunları, geçerli hamleleri ve kazananı kontrol eder. Tahtadaki alanları güncelleyebilmek için hamleYap(), display(), alanDoluMu(), ve kazananiBelirle() gibi fonksiyonlar içerir.

Oyun (Ana sınıf):

Oyun sırasında oyuncuların hamle yapmasını sağlar ve oyunun bitip bitmediğini kontrol eder.

1.3 İlişkiler:

OyunNesnesi ve Tahta arasında inheritance (kalıtım) ilişkisi vardır.

Oyuncu ve BilgisayarOyuncu arasında inheritance (kalıtım) ilişkisi vardır.

Oyun sınıfı, Tahta ve Oyuncu sınıflarını composition (bileşim) olarak kullanır.

2. UML Diyagramı

Aşağıda projede kullanılan sınıfların UML diyagramı verilmiştir.

- **Kalıtım okları**:
- OyunNesnesi sınıfından Tahta sınıfına.
- Oyuncu sınıfından BilgisayarOyuncu sınıfına.
- **Tahta ve Oyun**: Oyun sınıfı, Tahta sınıfını bir bileşen olarak içerir (- tahta : Tahta).

- **Oyuncu**: Oyun sınıfı, iki farklı oyuncuyu (oyuncu1, oyuncu2) işaretçi olarak tanımlar.
- **Oyun**, Oyuncu ve Tahta sınıflarını kullanır. Ancak bu kullanım bir kalıtım ilişkisi değildir, bu nedenle burada oklarla belirtilmemiştir.

3. Kod Açıklaması

Başlangıçta oyunumuzun tablo seklini başta şu şekilde ayarladık

----->#define SIZE 3 // Tahtanın boyutu (3x3)

3.1 OyunNesnesi Sınıfı

Bu sınıf, display() fonksiyonu ile tahtayı ekrana basma işlevi sağlar. Bu, soyut bir sınıf olup, diğer sınıflar tarafından türetilir. Temel bir sınıftır. Sanal yıkıcı, sınıfın alt sınıflarından türetilmiş nesneler silindiğinde, doğru şekilde temizlenmesini sağlar. Default derleyicinin yıkıcıyı otomatik olarak oluşturmasını sağlar.

```
 class OyunNesnesi {

public:

   virtual void display() const = 0;

   virtual ~OyunNesnesi() = default;

};
```

3.2 Oyuncu Sınıfı

Oyuncu sınıfı, bir oyuncunun adı ve skorunu tutar. Oyuncunun hamlesi, adı ve skoru ile ilgili işlemler burada sağlanır. Bu sınıf, oyuncuya kadar tüm oyuncuları temsil etmek için temel sınıf olarak kullanılır.

```
class Oyuncu {

protected:

   string ad;

   int skor;

public:

   Oyuncu(string oyuncuAd) : ad(oyuncuAd), skor(0) {}

   virtual void setAd(string oyuncuAd) { ad = oyuncuAd; }  // Oyuncunun adını ayarlama

   virtual string getAd() { return ad; }  // Oyuncunun adını alma


   virtual void setSkor(int yeniSkor) { skor = yeniSkor; }  // Skoru ayarlama
```

```
    virtual int getSkor() { return skor; }  // Skoru alma


    virtual void arttirSkor() { skor += 1; }  // Skoru artırma (her kazançta)
    virtual void arttirSkor()  // Skoru artırma (her kazançta)
    virtual void hamleYap(int& satir, int& sutun);   // Oyuncunun hamlesini alma
    virtual void kaydetVeriTabani(); // Oyuncu skorunu bir dosyaya kaydetme
    virtual ~Oyuncu() = default; // Sanal yıkıcı (temizleme işlemi için)
};
```

3.3 BilgisayarOyuncu Sınıfı

Bilgisayar oyuncusu, rastgele bir hamle yapacak şekilde özelleştirilmiştir. HamYap() fonksiyonu, bilgisayar hamlesini rastgele seçer. Oyuncu sınıfından türetilmiş.

```
class BilgisayarOyuncu : public Oyuncu {
public:
    BilgisayarOyuncu(string oyuncuAd) : Oyuncu(oyuncuAd) {} // Yapıcı
    void hamleYap(int& satir, int& sutun) override; // Bilgisayarın hamlesini yapma (rastgele)
};
```

3.4 Tahta Sınıfı

Tahta sınıfı, 3x3'lük bir oyun tahtası içerir ve tahtada hamle yapma, tahtayı görüntülemeyi basmayı, kazanmayı ve doluluğu kontrol etmeyi sağlar. OyunNesnesi tarafından türetilmiştir.

```
class Tahta : public OyunNesnesi {
private:
    char tahta[SIZE][SIZE];
public:
    Tahta(); // Yapıcı fonksiyon, tahtayı boşluklarla başlatır
    char getAlan(int satir, int sutun);     // Belirli bir alandaki sembolü alma
    void setAlan(int satir, int sutun, char sembol);    // Alanı sembol ile güncelleme
```

```
    void display() const override;   // Tahtayı ekrana basma
    bool hamleYap(int satir, int sutun, char sembol);   // Hamleyi tahtada yapma (geçerli bir hamle olup olmadığını kontrol eder)
    bool alanDoluMu();   // Tahtadaki tüm alanlar dolu mu kontrolü
    char kazananiBelirle();    // Kazananı belirleme (3'lü aynı sembol yatay, dikey ya da çapraz)
};
```

3.5 Oyun Sınıfı

Oyun sınıfını, oyunu yönetir. Her iki oyuncunun da ilerlemesini sağlar, kazananı belirler ve oyunun sonlanıp sonlanmadığını kontrol eder. Oyun sınıfı, iki oyuncunun (insan veya bilgisayar) karşılaştığı ana sınıf. Oyuncular arasında geçiş yapma ve oyun bitene kadar devam etme işlevleri eklendi. Kazanan belirlendikten sonra skor arttırma ve kaydetme işlemi tanımlandı.

```
class Oyun {
private:
    Tahta tahta;
    Oyuncu* oyuncu1;
    Oyuncu* oyuncu2;
    Oyuncu* aktifOyuncu;   // Şu anki aktif oyuncu
public:
    Oyun(Oyuncu* o1, Oyuncu* o2) : oyuncu1(o1), oyuncu2(o2), aktifOyuncu(o1) {}   // Yapıcı
    void oynaOyun();   // Oyunu oynama fonksiyonu
};
```

3.6 Ana Program

Ana programda kullanıcıdan oyun modu seçmesi isteniyor. Oyun modu olarak "İnsan vs İnsan" veya "İnsan vs Bilgisayar" seçilebiliyor. Seçime göre uygun oyuncular oluşturuluyor. Program sonunda belleği temizlemek için delete operatörleri eklenmiştir. Oyuncu nesneleri dinamik olarak oluşturulduğu için programın sonunda bellekten silinir.

```
int main() {
srand(time(0));    // Rastgelelik için tohumlama
Oyuncu* oyuncu1 = NULL; // Oyuncu1 için işaretçi (başlangıçta NULL)
```

```
Oyuncu* oyuncu2 = NULL;  // Oyuncu2 için işaretçi (başlangıçta NULL)

Oyun oyun(oyuncu1, oyuncu2);  // Oyun nesnesi oluştur

oyun.oynaOyun();  // Oyunu başlat

delete oyuncu1;  // Bellek temizleme

 delete oyuncu2;
```

4. Oyun Kuralları

•Oyuncular sırayla hamle yapar.

•Her hamle, bir satır ve bir sütun belirleyerek tahtanın üzerine 'X' veya 'O' sembollerini yerleştirir.

•Oyunculardan biri, yatay, dikey veya çapraz olarak 3 aynı sembol sıralanırsa kazanır.

•Kimse kazanamadığında oyun berabere biter. cout << "Beraberlik!" << endl;

•Tahtanın boyutundan büyük sayı girilirse cout << "Gecersiz hamle, tekrar deneyin." << endl; hatası veriliyor.

•Oyuncu kazandığında, skoru artırılır ve oyuncunun adı ve skoru actığımız ofstream dosya("oyuncu_skorları.txt", ios::app); dosyasına otomatik ekleniyor.

5.Oyunumuzu Oynarken Oluşan Ekran Görüntüleri: