KALITIM VE POLİMORFİZM

Örnek: Bir araç kiralama sistemi için bir C# programı tasarlayınız. Bu programda aşağıdaki gereksinimleri karşılayacak bir yapı oluşturunuz:

1. **Kalıtım Kullanımı:**
    - Temel bir Arac sınıfı tanımlayınız. Bu sınıf tüm araçlar için ortak özellikleri içermelidir:
        - Marka (string),
        - Model (string),
        - Yıl (int),
        - GunlukKiraBedeli (double).
    - Arac sınıfından türeyen iki sınıf oluşturunuz:
        - **Otomobil**: Bu sınıfta ek olarak KapiSayisi (int) özelliği bulunmalıdır.
        - **Motosiklet**: Bu sınıfta ek olarak KaskDahilMi (bool) özelliği bulunmalıdır.
2. **Polimorfizm Kullanımı:**
    - Tüm sınıflarda bir BilgiYazdir() metodu tanımlayınız.
        - BilgiYazdir() metodu, polimorfizm kullanılarak farklı araç türleri için özelleştirilmelidir.
3. **Kullanıcı Etkileşimi:**
    - Kullanıcıdan araç türünü (Otomobil veya Motosiklet) ve ilgili bilgileri alınız.
    - Girilen bilgilere göre uygun türde bir araç oluşturunuz.
4. **Ekran Çıktısı:**
    - Oluşturulan aracın tüm bilgilerini ekrana yazdırınız.

Kodu:

```
using System;

namespace AracKiralamaSistemi
{
   // Temel Araç Sınıfı
   class Arac
   {
      public string Marka { get; set; }
      public string Model { get; set; }
      public int Yil { get; set; }
      public double GunlukKiraBedeli { get; set; }

      // Polimorfizm için virtual metot
      public virtual void BilgiYazdir()
      {
```

```
        Console.WriteLine($"Marka: {Marka}, Model: {Model}, Yıl: {Yil}, Günlük Kira Bedeli:
{GunlukKiraBedeli} TL");
    }
}

// Otomobil Sınıfı
class Otomobil : Arac
{
    public int KapiSayisi { get; set; }

    public override void BilgiYazdir()
    {
        base.BilgiYazdir();
        Console.WriteLine($"Kapı Sayısı: {KapiSayisi}");
    }
}

// Motosiklet Sınıfı
class Motosiklet : Arac
{
    public bool KaskDahilMi { get; set; }

    public override void BilgiYazdir()
    {
        base.BilgiYazdir();
        Console.WriteLine($"Kask Dahil Mi: {(KaskDahilMi ? "Evet" : "Hayır")}");
    }
}

// Program Sınıfı
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Araç türünü seçiniz (1: Otomobil, 2: Motosiklet): ");
        int secim = int.Parse(Console.ReadLine());

        Arac arac; // Polimorfizm için temel sınıf referansı

        if (secim == 1)
        {
            arac = new Otomobil();
            Console.Write("Marka: ");
            arac.Marka = Console.ReadLine();
            Console.Write("Model: ");
            arac.Model = Console.ReadLine();
            Console.Write("Yıl: ");
            arac.Yil = int.Parse(Console.ReadLine());
            Console.Write("Günlük Kira Bedeli: ");
            arac.GunlukKiraBedeli = double.Parse(Console.ReadLine());
            Console.Write("Kapı Sayısı: ");
            ((Otomobil)arac).KapiSayisi = int.Parse(Console.ReadLine());
        }
        else if (secim == 2)
        {
            arac = new Motosiklet();
            Console.Write("Marka: ");
            arac.Marka = Console.ReadLine();
            Console.Write("Model: ");
            arac.Model = Console.ReadLine();
            Console.Write("Yıl: ");
            arac.Yil = int.Parse(Console.ReadLine());
            Console.Write("Günlük Kira Bedeli: ");
```

```
            arac.GunlukKiraBedeli = double.Parse(Console.ReadLine());
            Console.Write("Kask Dahil Mi (Evet/Hayır): ");
            ((Motosiklet)arac).KaskDahilMi = Console.ReadLine().ToLower() == "evet";
        }
        else
        {
            Console.WriteLine("Geçersiz seçim! Program sonlandırılıyor.");
            return;
        }

        Console.WriteLine("\nAraç Bilgileri:");
        arac.BilgiYazdir();
    }
  }
}
```

Örnek: Bir **e-ticaret sistemi** için ürün ve sipariş yönetimi yapacak bir yazılım geliştirin. Sistem, farklı türde ürünleri ve bu ürünlere ait siparişleri yönetmek üzere tasarlanmalıdır. Aşağıdaki gereksinimleri karşılayacak şekilde bir sınıf yapısı oluşturun:

Gereksinimler:

1. **Ürün Sınıfı** (Temel Sınıf):
   - Ürün adı, açıklaması, fiyatı gibi ortak özelliklere sahip olmalıdır.
   - Ayrıca her ürünün bir Satılabilir özelliği olacak (bazı ürünler stokta olmayabilir).
   - Ürün bilgilerini yazdıran bir metot (BilgileriYazdir) bulunmalıdır.
2. **Fiziksel Ürün Sınıfı**:
   - Fiziksel ürünler için ek özellikler (ağırlık, boyutlar vb.) ve fiyat hesaplamaları yapılmalıdır.
   - Ürün bilgilerini yazdıran bir metot (BilgileriYazdir) olmalı.
3. **Dijital Ürün Sınıfı**:
   - Dijital ürünler için ek özellikler (dosya formatı, indirilebilir link vb.) ve fiyat hesaplamaları yapılmalıdır.
   - Ürün bilgilerini yazdıran bir metot (BilgileriYazdir) olmalı.
4. **Sipariş Sınıfı**:
   - Siparişler, birden fazla ürün içerebilir. Siparişin durumu (hazırlanıyor, gönderildi, teslim edildi) gibi özellikler bulunmalıdır.
   - Siparişle ilişkili bir ödeme işlemi olacak.
5. **Program**:
   - Kullanıcıdan ürün türünü (fiziksel veya dijital) seçmesini ve ürün bilgilerini girmesini isteyin.
   - Ürünle ilişkilendirilmiş bir sipariş oluşturun ve siparişin tüm bilgilerini ekrana yazdırın.

Kodu:

```
using System;
using System.Collections.Generic;

// Ürün sınıfı (Temel sınıf)
public class Urun
{
    public string Ad { get; set; }
```

```
    public string Aciklama { get; set; }
    public decimal Fiyat { get; set; }
    public bool Satilabilir { get; set; }

    public virtual void BilgileriYazdir()
    {
        Console.WriteLine($"Ürün Adı: {Ad}");
        Console.WriteLine($"Açıklama: {Aciklama}");
        Console.WriteLine($"Fiyat: {Fiyat} TL");
        Console.WriteLine($"Satılabilir: {Satilabilir}");
    }
}

// Fiziksel Ürün sınıfı, Urun sınıfından türetilmiştir
public class FizikselUrun : Urun
{
    public double Ağırlık { get; set; }
    public string Boyutlar { get; set; }

    public override void BilgileriYazdir()
    {
        base.BilgileriYazdir();
        Console.WriteLine($"Ağırlık: {Ağırlık} kg");
        Console.WriteLine($"Boyutlar: {Boyutlar}");
    }
}

// Dijital Ürün sınıfı, Urun sınıfından türetilmiştir
public class DijitalUrun : Urun
{
    public string DosyaFormat { get; set; }
    public string IndirmeLinki { get; set; }

    public override void BilgileriYazdir()
    {
        base.BilgileriYazdir();
        Console.WriteLine($"Dosya Formatı: {DosyaFormat}");
        Console.WriteLine($"İndirme Linki: {IndirmeLinki}");
    }
}

// Sipariş sınıfı
public class Siparis
{
    public int SiparisNo { get; set; }
    public List<Urun> Urunler { get; set; }
    public string Durum { get; set; }
    public decimal ToplamTutar { get; set; }

    public void SiparisHesapla()
    {
        ToplamTutar = 0;
        foreach (var urun in Urunler)
        {
            ToplamTutar += urun.Fiyat;
        }
    }

    public void BilgileriYazdir()
    {
        Console.WriteLine($"Sipariş Numarası: {SiparisNo}");
        Console.WriteLine($"Sipariş Durumu: {Durum}");
        Console.WriteLine($"Toplam Tutar: {ToplamTutar} TL");
```

```
        Console.WriteLine("Sipariş Ürünleri:");
        foreach (var urun in Urunler)
        {
            urun.BilgileriYazdir();
        }
    }
}

// Program sınıfı
public class Program
{
    public static void Main()
    {
        // Kullanıcıdan ürün türünü seçmesini isteyin
        Console.WriteLine("Ürün türünü seçin (1: Fiziksel Ürün, 2: Dijital Ürün): ");
        int secim = int.Parse(Console.ReadLine());

        Urun urun = null;

        if (secim == 1)
        {
            urun = new FizikselUrun();
            Console.Write("Ürün Adı: ");
            urun.Ad = Console.ReadLine();
            Console.Write("Açıklama: ");
            urun.Aciklama = Console.ReadLine();
            Console.Write("Fiyat: ");
            urun.Fiyat = decimal.Parse(Console.ReadLine());
            Console.Write("Satılabilir mi? (true/false): ");
            urun.Satilabilir = bool.Parse(Console.ReadLine());
            Console.Write("Ağırlık: ");
            ((FizikselUrun)urun).Ağırlık = double.Parse(Console.ReadLine());
            Console.Write("Boyutlar: ");
            ((FizikselUrun)urun).Boyutlar = Console.ReadLine();
        }
        else if (secim == 2)
        {
            urun = new DijitalUrun();
            Console.Write("Ürün Adı: ");
            urun.Ad = Console.ReadLine();
            Console.Write("Açıklama: ");
            urun.Aciklama = Console.ReadLine();
            Console.Write("Fiyat: ");
            urun.Fiyat = decimal.Parse(Console.ReadLine());
            Console.Write("Satılabilir mi? (true/false): ");
            urun.Satilabilir = bool.Parse(Console.ReadLine());
            Console.Write("Dosya Formatı: ");
            ((DijitalUrun)urun).DosyaFormat = Console.ReadLine();
            Console.Write("İndirme Linki: ");
            ((DijitalUrun)urun).IndirmeLinki = Console.ReadLine();
        }

        // Sipariş oluşturuluyor
        Console.WriteLine("Sipariş Oluşturuluyor...");
        var siparis = new Siparis
        {
            SiparisNo = new Random().Next(1000, 9999),
            Urunler = new List<Urun> { urun },
            Durum = "Hazırlanıyor"
        };
        siparis.SiparisHesapla();
        siparis.BilgileriYazdir();
    }
```

}

Ödevler

1. Bir şirketin çalışanlarını yöneten bir program yazmanız isteniyor. Temel bir Calisan sınıfı oluşturun ve bu sınıftan türeyen Yazilimci ve Muhasebeci sınıflarını tanımlayın.

   Her çalışan, ad, soyad, maaş ve pozisyon bilgilerine sahiptir. Yazilimci sınıfında yazılım dili bilgisi, Muhasebeci sınıfında ise kullandığı muhasebe yazılımı bilgisi bulunsun.

   Aşağıdaki işlemleri gerçekleştirin:

   - Calisan sınıfında bir BilgiYazdir() metodu oluşturun. Bu metodun çıktısı, her türetilmiş sınıf için farklı şekilde çalışmalı.
   - Yazilimci ve Muhasebeci sınıflarında BilgiYazdir() metodunu override edin.
   - Program, kullanıcıdan çalışan türünü seçmesini ve ardından ilgili çalışanın bilgilerini alarak yazdırmasını sağlasın.

   Beklenen Çıktı:

   - Eğer kullanıcı Yazilimci seçerse, yazılımcı bilgileri yazdırılmalı.
   - Eğer kullanıcı Muhasebeci seçerse, muhasebeci bilgileri yazdırılmalı.

2. Bir hayvanat bahçesi yönetim sistemi yazınız. Temel bir Hayvan sınıfı oluşturun ve bu sınıftan türeyen Memeli ve Kus sınıflarını tanımlayın.

   Her hayvanın adı, türü ve yaş bilgisi vardır. Memeli sınıfında TuyRengi, Kus sınıfında ise KanatGenisligi özelliği bulunsun. Ayrıca her hayvanın bir ses çıkarmasını sağlamak için SesCikar() metodunu her sınıfta farklı şekilde tanımlayın.

   Aşağıdaki işlemleri gerçekleştirin:

   - Hayvan sınıfında bir SesCikar() metodu oluşturun. Bu metodun çıktısı her türetilmiş sınıf için farklı olmalı.
   - Memeli ve Kus sınıflarında SesCikar() metodunu override edin.
   - Program, kullanıcıdan hayvan türünü seçmesini ve ardından seçilen hayvanın bilgilerini ve sesini yazdırmasını sağlasın.

   Beklenen Çıktı:

   - Eğer kullanıcı Memeli seçerse, memelinin bilgileri ve ses çıkarma durumu yazdırılmalı.
   - Eğer kullanıcı Kus seçerse, kuşun bilgileri ve ses çıkarma durumu yazdırılmalı.

3. Bir banka sistemi için temel bir Hesap sınıfı oluşturun. Bu sınıftan türeyen VadesizHesap ve VadeliHesap sınıflarını tanımlayın.

   Her hesap, hesap numarası, bakiye ve hesap sahibine sahiptir. VadeliHesap sınıfında vade süresi ve faiz oranı, VadesizHesap sınıfında ise ek hesap limiti bulunsun.

Aşağıdaki işlemleri gerçekleştirin:

- Her hesap türü için bir ParaYatir() metodu oluşturun. Bu metodun çıktısı her türetilmiş sınıf için farklı olmalı.
- Her hesap türü için bir ParaCek() metodu oluşturun. VadeliHesap sınıfında vade dolmadan para çekme işlemi yapılmaya çalışılırsa kullanıcıya uyarı verilsin.
- Hesap sınıfında genel bir BilgiYazdir() metodu tanımlayın.
- Program, kullanıcıdan hesap türünü seçmesini, hesap bilgilerini almasını ve işlem yapmasını sağlayacak şekilde çalışmalıdır.

Beklenen Çıktı:

- Kullanıcı, `VadesizHesap` veya `VadeliHesap` türlerinden birini seçebilir.
- Hesaba para yatırma, para çekme ve hesap bilgilerini yazdırma işlemleri doğru şekilde gerçekleştirilmeli.
- Vadeli hesapta vade dolmadan para çekme işlemi yapılmaya çalışıldığında, uygun uyarı verilmelidir.

ABSTRACT CLASS

Normal Classlardan Farkları:

1. **Kalıtım:**
   - **Abstract class:** Soyut sınıflar kalıtım yoluyla türetilir, ancak doğrudan bir örneği (instance) oluşturulamaz. Abstract class içinde soyut (abstract) metotlar bulunabilir, bu metotlar alt sınıflarda implement (uygulanmak) edilmek zorundadır.
   - **Normal class:** Normal sınıflar ise doğrudan örneği oluşturulabilir ve türemek için herhangi bir zorunluluk yoktur. Yani, bir normal sınıf bir base (temel) sınıf olabilir, ancak alt sınıflarda metotları implement etmek zorunda değilsiniz.
2. **Soyut (Abstract) Metotlar:**
   - **Abstract class:** Soyut sınıflar içinde soyut metotlar (implementasyonu olmayan metotlar) tanımlanabilir. Bu metotlar, soyut sınıfı miras alan alt sınıflarda mutlaka implement edilmelidir.
   - **Normal class:** Normal sınıflarda metotlar genellikle tam olarak uygulanmış olur. Soyut metotlar tanımlanmaz.
3. **Örnekleme (Instantiation):**
   - **Abstract class:** Soyut sınıflardan doğrudan bir nesne oluşturulamaz. Bunun yerine, soyut sınıfın alt sınıflarından nesne oluşturulur.
   - **Normal class:** Normal sınıflardan doğrudan nesne oluşturulabilir

Örnek: Bir şekil çizimi uygulaması geliştirmek istiyorsunuz. Temel bir `Sekil` sınıfı oluşturun ve bu sınıftan türeyen `Daire` ve `Dikdortgen` sınıflarını tanımlayın.

- `Sekil` sınıfı, tüm şekillerin ortak özelliklerine sahip olacak (örneğin, `Ad` ve `Renk` gibi) ve her şekil için bir `Ciz()` soyut metodunu içerecek.
- `Daire` ve `Dikdortgen` sınıflarında, `Ciz()` metodunu kendi türlerine uygun şekilde uygulayın. Örneğin, dairede bir çember çizme, dikdörtgende ise dörtgen çizme davranışı olacak.

- `Sekil` sınıfı bir abstract sınıf olacak, böylece `Ciz()` metodunu her türetilen sınıf (örneğin, `Daire` ve `Dikdortgen`) kendine göre implement edecektir.
- Program, kullanıcıdan bir şekil türünü seçmesini ve seçilen şeklin bilgilerini (ad, renk) alarak o şekli ekrana çizecek.

Beklenen Çıktı:

- Eğer kullanıcı `Daire` seçerse, dairenin bilgileri ve çizimi yapılacak.
- Eğer kullanıcı `Dikdortgen` seçerse, dikdörtgenin bilgileri ve çizimi yapılacak.

Kodu:

```
using System;

// Abstract class - Soyut sınıf
public abstract class Sekil
{
    public string Ad { get; set; }
    public string Renk { get; set; }

    // Soyut metot
    public abstract void Ciz();

    // Genel şekil bilgisi yazdırma metodu
    public void BilgiYazdir()
    {
        Console.WriteLine($"Şekil Adı: {Ad}");
        Console.WriteLine($"Renk: {Renk}");
    }
}

// Daire sınıfı, Sekil sınıfından türetilmiştir
public class Daire : Sekil
{
    public int YariCap { get; set; }

    // Ciz() metodunun implementasyonu
    public override void Ciz()
    {
        Console.WriteLine($"{Ad} adlı daire, {Renk} renkli olarak çiziliyor. Yarıçap: {YariCap}");
    }
}

// Dikdortgen sınıfı, Sekil sınıfından türetilmiştir
public class Dikdortgen : Sekil
{
    public int Uzunluk { get; set; }
    public int Genislik { get; set; }

    // Ciz() metodunun implementasyonu
    public override void Ciz()
    {
        Console.WriteLine($"{Ad} adlı dikdörtgen, {Renk} renkli olarak çiziliyor. Uzunluk: {Uzunluk}, Genişlik:
{Genislik}");
    }
}

// Program sınıfı
public class Program
{
```

```
    public static void Main()
    {
        // Kullanıcıdan seçim alalım
        Console.WriteLine("Şekil Seçin (1: Daire, 2: Dikdörtgen): ");
        int secim = int.Parse(Console.ReadLine());

        Sekil sekil = null;

        // Kullanıcı seçimine göre nesne oluşturuluyor
        if (secim == 1)
        {
            sekil = new Daire();
            Console.Write("Daire Adı: ");
            sekil.Ad = Console.ReadLine();
            Console.Write("Daire Rengi: ");
            sekil.Renk = Console.ReadLine();
            Console.Write("Daire Yarıçapı: ");
            ((Daire)sekil).YariCap = int.Parse(Console.ReadLine());
        }
        else if (secim == 2)
        {
            sekil = new Dikdortgen();
            Console.Write("Dikdörtgen Adı: ");
            sekil.Ad = Console.ReadLine();
            Console.Write("Dikdörtgen Rengi: ");
            sekil.Renk = Console.ReadLine();
            Console.Write("Dikdörtgen Uzunluğu: ");
            ((Dikdortgen)sekil).Uzunluk = int.Parse(Console.ReadLine());
            Console.Write("Dikdörtgen Genişliği: ");
            ((Dikdortgen)sekil).Genislik = int.Parse(Console.ReadLine());
        }
        else
        {
            Console.WriteLine("Geçersiz seçim.");
            return;
        }

        // Şekil bilgisini yazdırıyoruz
        sekil.BilgiYazdir();

        // Şekil çizme işlemi
        sekil.Ciz();
    }
}
```

Ödevler

1. **Açıklama:** Bir banka hesabı yönetim sistemi tasarlamanız isteniyor. Bu sistemde, soyut sınıflar ve arayüzler kullanarak farklı hesap türlerinin özelliklerini yönetmelisiniz.

   Görevler:

   1. **Soyut Hesap sınıfını oluşturun**: Bu sınıfın HesapNo, Bakiye gibi özellikleri ve ParaYatir(decimal miktar) ve ParaCek(decimal miktar) metodları olmalı.
   2. **BirikimHesabi ve VadesizHesap sınıflarını oluşturun**: Bu sınıflar Hesap sınıfından türetilmeli.
      - BirikimHesabi sınıfı, birikim hesabına özel bir özellik tutmalı (örneğin faiz oranı) ve ParaYatir() metodunda faizi hesaplamalı.
      - VadesizHesap sınıfı, ParaCek() metodunda işlem ücreti uygulamalıdır.

3. **IBankaHesabi arayüzü oluşturun**: Bu arayüz, her hesabın HesapAcilisTarihi gibi bir özelliği ve HesapOzeti() gibi bir metod içermelidir.
4. Banka hesabı yönetimini test etmek için **birkaç hesap oluşturun** ve her hesap için işlem yaparak hesap özetini ekrana yazdırın.

2. **Açıklama:** Bir mağaza yönetim sistemi tasarlamanız isteniyor. Bu sistemde, farklı türdeki ürünlerin fiyatlarını ve ödeme hesaplamalarını yönetmek için soyut sınıflar ve polimorfizm kullanmalısınız.

Görevler:

1. **Soyut Urun sınıfı oluşturun**: Bu sınıfın Ad, Fiyat gibi özellikleri ve HesaplaOdeme() adlı soyut bir metodu olmalı.
2. **Kitap ve Elektronik sınıflarını oluşturun**: Bu sınıflar Urun sınıfından türetilmeli.
    - Kitap sınıfı, kitapla ilgili bilgi tutacak ve ödeme hesaplamasında %10'luk bir vergi ekleyecek.
    - Elektronik sınıfı, elektronik ürünle ilgili bilgi tutacak ve ödeme hesaplamasında %25'lik bir vergi ekleyecek.
3. **BilgiYazdir() metodu ekleyin**: Her ürün türü için, ürün bilgilerini ekrana yazdıracak bir metod tanımlayın.
4. Programınızda bir **List<Urun>** koleksiyonu oluşturun ve farklı türdeki ürünleri bu koleksiyona ekleyin. Ürün bilgilerini ve ödenecek tutarı ekrana yazdırın.

Not: HesaplaOdeme() metodunu her sınıf için kendi mantığına göre yeniden yazmayı unutmayın.

3. **Açıklama:** Bir yayıncı ve abone sistemi tasarlamanız isteniyor. Bu sistemde, yayıncılar değişiklik olduğunda abone olan kullanıcıları bilgilendirecek. Bu görev için **Observer Design Pattern** kullanmalısınız.

Görevler:

1. **IYayinci ve IAbone arayüzlerini oluşturun**:
    - IYayinci arayüzü, abone ekleme, abone çıkarma ve abone listeleme metodlarına sahip olmalı.
    - IAbone arayüzü, her abone için BilgiAl() metoduna sahip olmalı.
2. **Yayinci sınıfı oluşturun**: Bu sınıf, IYayinci arayüzünü implement etmeli ve tüm abonelere güncellemeleri bildirmelidir.
3. **Abone sınıfı oluşturun**: Bu sınıf, IAbone arayüzünü implement etmeli ve her güncellemeyi alarak ekrana yazdırmalıdır.
4. **Observer pattern'ı test etmek için**:
    - Bir yayıncı oluşturun ve birkaç abone ekleyin.
    - Yayıncı bir değişiklik yaptığında tüm abonelere bu değişiklikleri bildirin.

Not: Observer pattern'ı uygularken, bir yayıncı değişiklik yaptığında tüm abonelerin bu değişikliği almasını sağlamalısınız.