T.C.
MİLLÎ EĞİTİM BAKANLIĞI

BİLİŞİM TEKNOLOJİLERİ

NESNE TABANLI PROGRAMLAMADA SINIFLAR

482BK0163

Ankara, 2011

- Bu modül, mesleki ve teknik eğitim okul/kurumlarında uygulanan Çerçeve Öğretim Programlarında yer alan yeterlikleri kazandırmaya yönelik olarak öğrencilere rehberlik etmek amacıyla hazırlanmış bireysel öğrenme materyalidir.
- Millî Eğitim Bakanlığınca ücretsiz olarak verilmiştir.
- PARA İLE SATILMAZ.

İÇİNDEKİLER

AÇIKLAMALAR

KOD	**482BK0163**
ALAN	**Bilişim Teknolojileri**
DAL/MESLEK	**Veritabanı Programcılığı**
MODÜLÜN ADI	**Nesne Tabanlı Programlamada Sınıflar**
MODÜLÜN TANIMI	Nesne tabanlı programlama dilinde sınıfları oluşturma ve kullanma yeterliliğinin kazandırıldığı bir öğrenme materyalidir.
SÜRE	40/24
ÖN KOŞUL	Nesne Tabanlı Programlamada Karar ve Döngü Yapıları modülünü almış olmak
YETERLİK	Nesne tabanlı programlama dili ile sınıf oluşturmak
MODÜLÜN AMACI	**Genel Amaç** Bu modül ile gerekli ortam sağlandığında sınıfları ve nesneleri yönetebileceksiniz. **Amaçlar** **1.** Sınıf tanımlamalarını hatasız olarak uygun yerlerde yazabileceksiniz. **2.** Gerektiğinde static metotları kullanabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	**Ortam:** Veritabanı Programcılığı Atölyesi **Donanım:** Nesne tabanlı programlama için nesne tabanlı programlama yazılımı yüklü bilgisayar
ÖLÇME VE DEĞERLENDİRME	Modül içinde yer alan her öğrenme faaliyetinden sonra verilen ölçme araçları ile kendinizi değerlendireceksiniz. Öğretmen modül sonunda ölçme aracı (çoktan seçmeli test, doğru-yanlış testi, boşluk doldurma, eşleştirme vb.) kullanarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek sizi değerlendirecektir.

GİRİŞ

Sevgili Öğrenci,

Tüm çağdaş programlama dilleri tarafından desteklenen “Nesne Tabanlı Programlama” ya da “Nesne Yönelimli Programlama” (Object Oriented Programming) 1960’lı yılların sonuna doğru ortaya çıkan bir programlama yaklaşımıdır. O tarihte bile yazılımların karmaşıklığı, boyutları ve bakımı konusunda beliren bir bunalımın sonucu ortaya çıkmıştır. “Nesne Tabanlı Programlama” yaklaşımını bu sorunlara karşı bir çözüm hâline getiren en önemli özellik yazılımda modülerliği benimsemesidir. “Nesne Tabanlı Programlama” ayrıca bilgi gizleme, veri soyutlama, çok biçimlilik ve kalıtım gibi yazılımın bakımını kolaylaştıran ve aynı yazılım üzerinde birden fazla kişinin çalışabilmesini mümkün kılan avantajlara sahiptir. Sağladığı bu avantajlardan dolayı “Nesne Tabanlı Programlama” günümüzde geniş çaplı yazılım projelerinde yaygın olarak kullanılmaktadır.

“Nesne Tabanlı Programlama” yaklaşımında benimsenen modülerliğin ana fikri, her bilgisayar programının etkileşim içerisinde olan nesneler kümesinden oluştuğu varsayımıdır. Bu yaklaşımı kullanarak yazılım geliştirmek isteyen bir programcının nesnelerin nasıl oluşturulduğunu ve kullanıldığını çok iyi bilmesi gerekir. İşte bu nesneleri oluşturmak için öğrenmeniz gereken sınıflar konusunu Nesne Tabanlı Programlamada Sınıflar modülü yardımı ile öğreneceksiniz. Elinizdeki bu modül size program geliştirme aşamasında sınıfların nasıl tanımlandığını ve kullanıldığını anlatmak için hazırlanmıştır.

ÖĞRENME FAALİYETİ–1

AMAÇ

Bu öğrenme faaliyeti sonunda ihtiyacınıza uygun sınıfları tanımlayabileceksiniz. Uygun anahtar kelimeler ile sınıflar ve sınıflar içerisindeki elemanların erişilebilirlik seviyelerini belirleyebileceksiniz. Sınıfları ilk kez kullanıma hazır hâle getirmek için kurucuları kullanabileceksiniz.

ARAŞTIRMA

- Çevrenizdeki beş farklı nesnenin (masa, televizyon, bisiklet vb.) özelliklerini (18 vites, 10 km/sa., 26” jant vb.) ve davranışlarını (vites değiştirme, fren yapma, hızlanma vb.) yazınız.
- Arama motoru kullanarak nesne tabanlı programlamada nesne ve sınıf kavramları arasındaki farkları bulunuz.
- Kişisel kullanım için geliştireceğiniz örnek bir uygulama belirleyiniz. Bu uygulama için hangi sınıfları tanımlamanız gerektiğini bulmaya çalışınız.

1. SINIFLAR

Günlük hayatta kullanılan pek çok kelime bir sınıflandırmanın sonucu ortaya çıkmıştır. Örneğin insan, ev, araba vb. kelimeler bir tek nesneyi değil, ortak davranış ve özellikleri olan nesnelerin tümünü ifade etmek için kullanılır. Araba kelimesi ile tekerlekleri olan motorlu veya motorsuz her türlü kara taşıtından bahseder Bütün arabaların ortak davranışları (hızlandırılabilir, durdurulabilir, yönlendirilebilir vb.) ve ortak özellikleri (tekerlekleri, direksiyonları vb.) vardır. Bu ortak davranışlara ve ortak özelliklere sahip bütün nesneler için araba kelimesi kullanılabilir. İşte her an belki farkında olmadan yapılan bu sınıflandırma, nesne tabanlı programlama yaklaşımının temelinde de vardır.

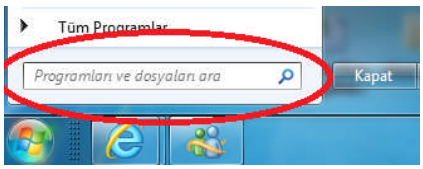
Nesne tabanlı programlamadan önceki tek yaklaşım olan yordamsal programlamada programlar sadece bir komut dizisi veya birer işlev (fonksiyon) kümesi olarak görülmekteydi. Bu yaklaşımla çok karmaşık problemlerin üstesinden gelmek oldukça zordu. Ancak nesne tabanlı programlama yaklaşımının benimsemiş olduğu ve insanoğlunun düşünme şekline daha yakın olan sınıflandırma yöntemi ile daha karmaşık problemleri kolayca çözmek mümkün olmuştur.

Nesne tabanlı programlama ile uygulama geliştirirken ilk olarak hangi varlıklar üzerinde çalışılacağını tasarlamak gerekir. Bir varlık (entity), hangi sınıfları oluşturmak gerektiği gösterir. Örneğin, kişinin ve sınıf arkadaşlarının okulda girmiş olduğu sınavlardan

alınan puanları kaydetmek ve ortalamaları hesaplatmak istenilen basit bir uygulama geliştirildiği düşünülsün. Bu durumda öğrenci, ders ve sınav sınava girenlerin varlıkları olacaktır. ogrenci, ders ve sinav adında üç sınıf tanımlaması yapmak gerekir. Tanımlanan bu üç sınıf ile varlıklar programlama dili ile modellenmiş olunur. Ardından tanımlanacak sınıflar için hangi bilgilerin gerekli olduğu ve sınıfların hangi işlevleri gerçekleştirmesi gerektiği düşünülmelidir. Yani sınıflarınızdan üretilecek olan nesnelerin ortak davranışları ve özellikleri belirlenmelidir.

1.1. Sınıf (Class) Mantığı

Nesne tabanlı programlamada gereken nesneleri sınıflar yardımıyla oluşturmaktır.Sınıfları, nesneleri oluşturmak için önceden oluşturulmuş bir kalıp veya ilk örnek (prototip) olarak da düşünülebilir. Örneğin, kâğıt üzerine detayları çizilmiş bir ev planını kullanarak aynı evden pek çok örnek ev oluşturulabilir. İşte burada kâğıt üzerindeki bu plan sınıftır. Bu plana bakılarak yapılan ev ise sınıftan oluşturulan nesne örneğidir.



Resim 1.1: Başlat menüsünde bulunan arama-filtreleme metin kutusu

Yukarıdaki resimde bulunan metin kutusu işletim sistemini yazan programcılar tarafından sınıf olarak kodlanmış ve işletim sistemi içerisine entegre edilmiştir. İşletim sistemi içerisinde yer alan bu kontrole ihtiyaç duyulduğunda var olan sınıf kodundan (prototip) nesne oluşturularak kullanılmıştır.

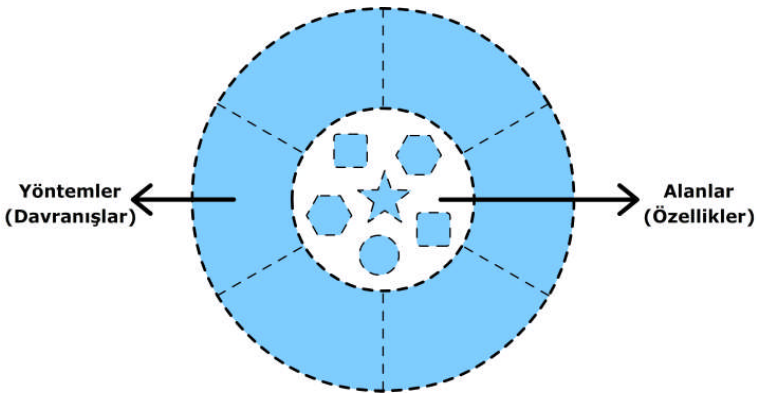
Resim 1.2: Windows gezgini penceresinde yer alan arama-filtreleme metin kutusu

Yukardaki resim windows gezgini penceresinden alınmıştır. Burada yer alan kontrol de aslında Resim 1.1'de yer alan kontrolün birebir aynısıdır. Burada aynı kontrol tekrar kodlanmamış daha önce bu kontrol için oluşturulmuş olan sınıfın bir nesnesi daha oluşturularak pratik bir şekilde eklenmiştir

Yukarıdaki örneklerden anlaşılacağı üzere sınıf kavramının sağladığı en büyük fayda bir defa kodlanarak her zaman kullanılabilecek nesneler oluşturabilme imkânı vermesidir.

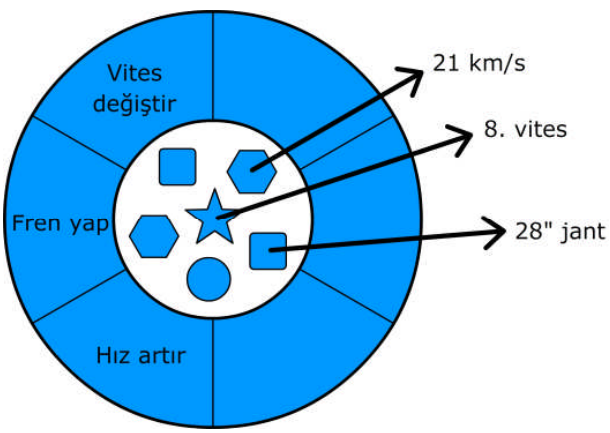
Bir sınıftan nesne oluşturma işlemine *örnekleme* (instantiating) denir. Aynı sınıftan istenilen sayıda nesne örnek oluşturulabilir. Oluşturulan tüm nesneler daha önce sınıfta belirlenen özelliklere ve davranışlara sahip olur.

Sınıfların hangi özelliklere sahip olacağını yani hangi bilgileri saklayacağını sınıflar içerisinde tanımlanan *alanlar* (üye değişkenleri) yardımıyla belirlenebilir. Sınıfların hangi davranışlara sahip olduğu ya da hangi işlevleri yerine getireceğini ise sınıf tanımı içerisinde yazılan *metotlar* (yöntemler) ile belirlenir (Şekil 1.1) .



Şekil 1.1: Bir yazılım nesnesi ilk örneği (sınıf)

Örneğin, bir bisikleti yazılım nesnesi olarak modellersek bisiklet sınıfta hız, vites, jant büyüklüğü gibi alanlar tanımlanabilir. Ayrıca vites değiştirmek, fren yapmak ve hızı artırmak için metotlar tanımlamak gerekir (Şekil 1.2).



Şekil 1.2: Yazılım nesnesi olarak modellenmiş bir bisiklet

Bisiklet nesnesini yukarıdaki şekilde gösterildiği gibi modellendiğinde dış dünyanın onu nasıl kullanılacağı da belirlenir. Örneğin, oluşturulan bisiklet örneğinin 18 vitesi varsa vitesi değiştirmek için yazılan metodun 1'den küçük ve 18' den büyük sayıları kabul etmemesi gerekir.

1.1.1. Kapsüllemenin Amacı

Giriş bölümünde bilgi gizlemenin nesne tabanlı programlamanın avantajlarından biri olduğu hatırlatılmıştı. Kapsülleme (encapsulation) sayesinde bir sınıfı kullanırken o sınıfın nasıl çalıştığını bilmek zorunda kalınmaz. Sınıfın bir örneğini oluşturur ve ihtiyaç duyulan metotlar çağrılır. Metotlar yerine getirmeleri gereken işlevleri gerçekleştirdikleri sürece,

metotların nasıl çalıştığı ile ilgilenmez ve böylece karışıklıktan uzak durmuş olunur. Kapsülleme aynı zamanda sınıfların metotlarını, verilerini ve alanlarını diğer nesnelerden saklayarak ve bunlara erişimi sınırlandırarak sizi yanlış kullanımlardan korur. Bunun nasıl yapıldığı erişilebilirlik konusunda ayrıntılı olarak görülecektir.

1.2. Sınıf Tanımlama

Nesne tabanlı programlamada gereken nesneler sınıflar yardımıyla oluşturulacaktır. Şimdi ilk olarak programlama dilinde "Bisiklet" sınıfının nasıl oluşturulduğu görülsün:

```
class Bisiklet
  {
        int hiz = 0; //Burada int tipinde bir alan tanımlanıyor
        int vites = 1; //Burada int tipinde bir alan tanımlanıyor
        public void VitesDegistir(int yeniDeger)
        {
              vites = yeniDeger;
        }
        public void Hizlan(int artis)
        {
              hiz = hiz + artis;
        }
        public void FrenYap(int azalma)
        {
              hiz = hiz - azalma;
        }
        public void BilgileriYaz()
        {
              Console.WriteLine("Vites: {0}", vites);
              Console.WriteLine("Hız: {0}", hiz);
        }
  }
```

Görüldüğü gibi *Bisiklet* sınıfının *hiz* ve *vites* adında iki alanı, ayrıca *VitesDegistir*, *Hizlan*, *FrenYap* ve *BilgileriYaz* adında dört metodu bulunmaktadır. Alanların aslında sınıf içerisinde tanımlanan değişkenlerden başka bir şey olmadığına dikkat edilmelidir. Nesne Tabanlı Programlamaya Giriş modülünde değişkenlerin nasıl tanımlandığı ve Nesne Tabanlı Programlamada Metotlar modülünde metotların nasıl bildirildiği öğrenilmişti.

Aslında Bisiklet sınıfının tanımında hemen hemen hiç yeni söz dizimi yoktur. Burada sadece daha önce kullanılan kavramlar *class* anahtar kelimesi ile belirtilen bir sınıf kapsamı içerisine aldık ve ilerleyen konularda öğrenilecek metotların erişilebilirlik seviyeleri için *public* anahtar sözcüğü kullanıldı.

1.3. Sınıfı Kullanma

Daha önce bir sınıftan nesne oluşturma işlemine örnekleme denildiği belirtilmişti. Bir sınıf tanımlandığında aslında yeni bir değişken türü tanımlamış olunur. Daha önce oluşturulan "Bisiklet" türünden aşağıdaki gibi bir değişken tanımlansın:

Bisiklet b;

Bunun daha önce tanımlanan değişkenlerden hiçbir farkı yoktur. Örneğin daha önce tamsayı (int) türünde bir değişkeni şöyle tanımlanır:

int sayac;

Sınıf tanımlamalarında farklı olan durum, değişkene *new (yeni)* anahtar sözcüğü ile geçerli bir başlangıç verisinin atanması gerektiğidir.

```
Bisiklet b;
b = new Bisiklet();
```

Oysa tam sayı türünde tanımlanan değişken için böyle bir şeye gerek yoktu.

```
int sayac;
sayac = 1;
```

Görüldüğü gibi *new* anahtar sözcüğü bir sınıfın yeni bir örneğini yani yeni bir nesneyi oluşturmak için kullanılmaktadır. *Bisiklet* sınıfından iki bisiklet nesnesi oluşturan ve bunların metotlarını kullanan aşağıdaki örnek dikkatlice incelenmelidir ve ekran çıktısı anlamaya çalışılmalıdır.

```
class Program
{
        class Bisiklet
        {
                //Alan tanımlarının public ifadesi ile başlamadığına dikkat ediniz.
                int hiz = 0; //int tipinde bir alan tanımlanıyor
                int vites = 1; //int tipinde bir alan tanımlanıyor

                public void VitesDegistir(int yeniDeger)
                {
                        vites = yeniDeger;
                }
                public void Hizlan(int artis)
                {
                        hiz = hiz + artis;
                }
                public void FrenYap(int azalma)
```

```
        {
            hiz = hiz - azalma;
        }
        public void BilgileriYaz()
        {
            Console.Write("Vites: {0}", vites );
            Console.WriteLine("Hız: {0}", hiz);
        }
    }
    static void Main(string[] args)
    {
        // İki farklı bisiklet nesnesi oluşturuluyor
        Bisiklet bisiklet1 = new Bisiklet();
        Bisiklet bisiklet2 = new Bisiklet();

        bisiklet1.Hizlan(10);
        bisiklet1.VitesDegistir(2);
        bisiklet1.BilgileriYaz();

        bisiklet2.Hizlan(10);
        bisiklet2.VitesDegistir(2);
        bisiklet2.Hizlan(10);
        bisiklet2.VitesDegistir(3);
        bisiklet2.BilgileriYaz();
    }
}
```

Ekran çıktısı:
Vites: 2 Hız: 10
Vites: 3 Hız: 20

1.4. Erişilebilirlik

Bir sınıf içindeki üyelere dışarıdan erişim sağlamak veya sınırlandırmak için erişim türleri belirlenmiştir. Nesne tabanlı programlama da beş tane erişim türü bulunmaktadır.

- public
- private
- protected
- internal
- protected internal

public: Bu erişim türü ile tanımlanmış sınıf üyelerine her yerden erişim izni verilmiş olur.

private: Bu erişim türünde tanımlanmış üyelere sadece aynı sınıf içerisinde tanımlanmış diğer üyeler tarafından erişim izni verilmiş olur.

protected: Bu erişim türünde tanımlanmış üyelere tanımlandık sınıf ve bu sınıftan türetilmiş diğer sınıflardan erişim izni verilmiş olur.

İnternal: Bu erişim türünde tanımlanmış üyelere aynı assembly içinden erişim izni verilmiş olur.

protected internal: Bu erişim türü protected ve internal erişim türlerinin veya işlemiyle birleştirilmiş halidir.

class veya *struct*'lar **public** ya da **internal** olabilir, varsayılan olarak **internal** erişim türüne sahiptir.

Erişim türü belirlenmediği sınıf üyeleri varsayılan olarak **private** olarak kabul edilir.

Yukarıdaki bisiklet örneğini ele alınsın. Bir tane bisiklet adında sınıf tanımlanmıştır. Bu sınıfın içinde integer tipinde hız ve vites adında iki tane de değişken tanımlanmıştır. Dikkat edersen bu değişkenlere erişim türü atanmamıştır. Dolayısıyla program tarafından varsayılan olarak private olarak tanınacaklardır. Private olarak tanındıkları için sadece bisiklet sınıfı içinden erişim sağlanacaktır. Programın ana blokunda bisiklet sınıfından türetilen nesneler tarafından erişilemeyecektir.

bisiklet1.vites = 2; // Hata verir (vites için sınıf içerisinde bir erişim belirleyici kullanılmadığı için private kabul edilecektir)

Fakat bisiklet sınıfının içinde tanımlanmış Hizlan metodu public olarak tanımlandığı için her yerden erişim imkânı vardır. Dolayısıyla programın ana blokunda tanımlanmış nesneler tarafından da erişilebilecektir.

bisiklet1.Hizlan(10); // Hata vermeyecektir.

1.4.1. Adlandırma Kuralları

Erişilebilirlik kuralları yazılan kod içerisinde daha kolay görebilmek için aşağıdaki adlandırma kurallar yardımcı olur:

- *Public* alan ve metot adları büyük harfle başlamalıdır. Örneğin, *VitesDegistir* metodu *public* olduğundan " *V* " ile yazılmıştır. Bu adlandırma kuralına ilk defa Pascal programlama dilinde kullanıldığı için PascalCase adı verilmektedir.
- Private tanımlayıcılar küçük harfle başlamalıdır. Mesela *vites* alanı private olduğu için " *v* " ile yazılmıştır.

1.5. Kurucular (Constructors)

Kurucu (Constructor – Konstraktır diye okunur.) bir sınıfın örneği oluşturulduğunda kendiliğinden çalışan özel bir metottur. Sınıf ile aynı ada sahip olmak zorundadır, parametreler alabilir ama geri değer döndürmez. Tüm sınıfların bir kurucusu olmak zorundadır. Aşağıdaki örnek Bisiklet sınıfının varsayılan kurucusunda *vites* alanına 2 değerini atamaktadır:

```
class Bisiklet
        {
                public Bisiklet()
                        {
                                vites = 2;
                        }
        }
```

Dikkat edilirse kurucunun tanımında *void* anahtar sözcüğü kullanılmamıştır. Bunun nedeni kurucuların *void* (boş) değer bile döndürmemesidir. Yukarıdaki örnek bize kurucuların neden gerekli olduğuna dair ipucu verir. Bisiklet sınıfının örnekleri oluşturulurken otomatik olarak çalışan bu metot her örnek için vites değerini 2 olarak belirleyecektir. Böylece oluşturulan her bisiklet yola 2. vites ile başlayacaktır. Görülen kurucular sınıf başlatılırken yapılması gereken işlemleri yapmak için mükemmel bir yerdir.

1.5.1. Varsayılan Kurucu

Programlama dilinde varsayılan (default) kurucu herhangi bir parametre almayan kurucudur. *Bisiklet* sınıfını tanımlarken bizim de yapılan bu kurucu yazılmasa bile derleyici, kişinin yerine hiçbir şey yapmayan bir varsayılan kurucu üretir.

1.5.2. Aşırı Yüklenmiş Kurucu

Tüm metotlar gibi kurucular da aşırı yüklenebilir. Tanımlanan sınıftan örneği oluşturulan bazı nesnelerin farklı durumlarda başlatılması gerektiğinde kurucular aşırı yüklenebilir. Bunu daha iyi açıklamak için Bisiklet sınıfını oluşturulacak her bisiklet örneği, farklı vites sayısı ile oluşturabilmeyi sağlayan metot aşağıdaki gibi değiştirildi.

```
class Bisiklet
{
        int hiz = 0;
        int vites = 1;
        int vitesSayisi = 6;

        public Bisiklet()
        {
                vites = 2;
        }

        public Bisiklet(int deger)
        {
                vites = 2;
                vitesSayisi = deger;
        }
}
```

Yukarıdaki kod incelenirse artık varsayılan kurucuyu kullanarak oluşturulan bisikletlerin 6 vitesli olduğu görülebilir. Eğer 21 vitesli bir bisiklet örneği oluşturmak

istenirse kurucunun parametre alan sürümü kullanılarak bunu gerçekleştirmek gerekir. Kurucunun aldığı parametre değeri *vitesSayisi* alanının başlangıç değerinin üzerine yazacaktır. Çünkü kurucu, alanların ilk değerleri atandıktan sonra çalıştırılır.

```
Bisiklet cocukBisikleti = new Bisiklet(); // 6 vitesli bisiklet
Bisiklet dagBisikleti = new Bisiklet(18); // 18 vitesli bisiklet
```

Nesnenin örneği oluşturulurken derleyici *new* ifadesinde kullanılan parametrelere bakarak hangi kurucuyu çalıştırması gerektiğin belirler. Yukarıdaki örnekte *dagBisikleti* nesnesini oluştururken tam sayı türünde parametre girildiği için derleyici *int* parametre alan kurucuyu çalıştırır.

UYGULAMA FAALİYETİ

Sınıflara neden ihtiyaç duyulduğunu açıklayınız.

İşlem Basamakları	Öneriler
➢ Nesne tabanlı programlama yazılımını başlatınız.	➢ Örnek projelerinizi kendinize ait bir klasör içerisinde oluşturursanız başka ortamda çalışmanız daha kolay olacaktır.
➢ SinifOrnegi adında yeni bir konsol uygulaması oluşturunuz.	➢ Programlama dilini seçmeyi unutmayınız.
➢ Solution explorer'de Program.cs dosyasına çift tıklayınız.	➢ Solution explorer penceresi görünmüyorsa View menüsünü kullanınız.
➢ Program sınıfında Main metodunu bulun ve bu metodun üst kısmında "Bisiklet" sınıfını 8. sayfadaki koddan faydalanarak oluşturunuz.	➢ Yazdığınız sınıfın "Program" sınıfının içinde olmasına dikkat ediniz.
➢ Main metodunun içerisinde iki tane bisiklet nesnesi oluşturunuz.	➢ Oluşturduğunuz nesnelere anlamlı isimler veriniz (bisiklet1, bisiklet2 vb.).
➢ Oluşturduğunuz nesnelerin VitesDegistir, Hizlan, Frenyap metotlarını farklı kereler kullanınız. ➢ BilgileriYaz metodunu kullanarak ekrana yazılan değerleri inceleyiniz.	➢ İki nesne için kullandığınız metotlarda farklı değerler kullanınız.
➢ Gereken anahtar sözcük ile vites ve hiz alanlarının erişilebilirliklerini değiştiriniz. ➢ Bu alanların değerlerini metot kullanmadan değiştiriniz. ➢ Bisiklet sınıfına varsayılan kurucu yazınız. ➢ VitesSayisi adında yeni bir alan ekleyiniz ve bu alana bir ilk değer atayınız. ➢ Tamsayı türünde bir parametre alan ve bu parametre değerini VitesSayisi alanına atayan bir kurucu daha yazın.	➢ Sayfa 10 ve 11'deki örneklerden faydalanınız.

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet,** kazanamadıklarınız için **Hayır** kutucuklarına (X) işareti koyarak öğrendiklerinizi kontrol ediniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Nesne tabanlı programlama yazılımını doğru şekilde başlattınız mı?		
2. Konsol uygulamasını belirli bir klasörde oluşturdunuz mu?		
3. Program.cs dosyasını kod düzenleyicide açtınız mı?		
4. Sınıf tanımlamasını doğru yerde yaptınız mı?		
5. Nesne örneklerini oluşturdunuz mu?		
6. Oluşturduğunuz nesnelerin metotlarını kullandınız mı?		
7. Alanların erişilebilirliklerini değiştirdiniz mi?		
8. Alanların değerini metot kullanmadan değiştirdiniz mi?		
9. Varsayılan kurucu yazdınız mı?		
10. Kurucuyu aşırı yüklediniz mi?		

DEĞERLENDİRME

Değerlendirme sonunda “Hayır” şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız “Evet” ise “Ölçme ve Değerlendirme” ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

1. …… nesneleri oluşturmak için önceden oluşturulmuş bir kalıp veya ilk örnek (prototip) olarak da düşünülebilir.

2. Bir sınıftan nesne oluşturma işlemine …… denir.

3. Sınıfların hangi özelliklere sahip olacağı yani hangi bilgileri saklayacağı sınıflar içerisinde tanımlanan…… yardımıyla belirlenir.

4. Sınıfların hangi davranışlara sahip olduğu ya da hangi işlevleri yerine getireceği sınıf tanımı içerisinde yazılan …… ile belirlenir.

5. …… sayesinde bir sınıfı kullanırken o sınıfın nasıl çalıştığı bilinmek zorunda kalınmaz.

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

6. Aşağıdakilerden hangisi sınıf kapsamı dışında yapılır?
A) Alan tanımlama
B) Metot tanımlama
C) Kurucu oluşturma
D) Sınıfı örnekleme (Sınıfın kurulması)

7. Aşağıda verilen sınıftan örneklenen bisikletin nesnesi için aşağıdakilerden hangisi derleyici hatasına neden olur?

```
class Bisiklet
{
        private int hiz = 0;
        public int vites = 1;
        int vitesSayisi = 6;
        public Bisiklet()
        {
                vites = 2;
        }
        public void VitesDegistir(int yeniDeger)
        {
                vites = yeniDeger;
        }
        public void Hizlan(int artis)
        {
                hiz = hiz + artis;
        }
}
```

A) bisikletim.VitesDegistir(3);
B) bisikletim.vitesSayisi = 18;
C) bisikletim.Hizlan(20);
D) bisikletim.vites = 5;

8. Tanımlanan sınıftan örneği oluşturulan bazı nesnelerin farklı durumlarda başlatılması gerektiğinde aşağıdakilerden hangisi kullanılır?
A) Örnekleme
B) Aşırı yüklenmiş kurucu
C) Metot
D) Alan

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ–2

AMAÇ

Gerektiğinde static metotları kullanabileceksiniz.

ARAŞTIRMA

- Çevrenizdeki 5 farklı nesnenin (masa, televizyon, bisiklet, vb.) özelliklerini (18 vites, 10 km/sa., 26” jant vb.) ve davranışlarını (vites değiştirme, fren yapma, hızlanma vb.) yazınız.
- Arama motoru kullanarak nesne tabanlı programlamada nesne ve sınıf kavramları arasındaki farkları bulunuz.
- Kişisel kullanım için geliştireceğiniz örnek bir uygulama belirleyiniz. Bu uygulama için hangi sınıfları tanımlamanız gerektiğini bulmaya çalışınız.

2. “STATİC” METOTLAR

Şimdiye kadar metotlara ancak tanımlanan sınıfın bir örneği oluşturularak erişildi. Ama gerçekte bu her zaman gerekli değildir. Örneğin, Console sınıfının WriteLine metodu hatırlanmalıdır.

Console.WriteLine("Merhaba");

Görüldüğü gibi ekrana “Merhaba” yazmak için Console sınıfının bir örneği oluşturulmadı. Çünkü bunu yapmak için nesneye gerek yoktur, gerekli olan sadece metottur. Ancak tüm metotlar bir sınıfın içinde tanımlanmak zorundadır. Neyse ki bir yöntem ya da alanı *static* (statik) olarak tanımlanırsa herhangi bir nesne oluşturmadan sadece sınıfın adı kullanılarak metot çağrılabilir veya alana erişilebilir. Buradan da anlaşılacağı üzere *Console* sınıfı içerisinde *WriteLine* metot aşağıdaki söz dizimi ile *static* olarak tanımlandığı için onu bu şekilde kullanılabilmektir.

public static void WriteLine(string value)

2.1. Paylaşılan Alan Oluşturma

Sınıf içerisinde tanımlanan alanlar da static olarak oluşturulabilecek. Ancak burada dikkat edilmesi gereken önemli bir durum mevcuttur. Alanlar bilgi sakladıkları için static olarak tanımlanan alanlar o sınıftan oluşturulmuş bütün nesneler tarafından paylaşılır. Bu, bir örnekle açıklamaya çalışılsın.

```
class Bisiklet
{
  int hiz = 0;
  int vites = 1;
  public static int BisikletSayisi = 0;

  public Bisiklet()
  {
        vites = 2;
        BisikletSayisi++; // Alanın değeri 1 artırılıyor
  }

  public void VitesDegistir(int yeniDeger)
  {
        vites = yeniDeger;
  }

  public void Hizlan(int artis)
  {
        hiz = hiz + artis;
  }

  public void BilgileriYaz()
  {
        Console.Write("Bisiklet Sayısı: {0} ", BisikletSayisi);
        Console.Write("Vites: {0} ", vites);
        Console.WriteLine("Hız: {0}", hiz);
  }
}

static void Main(string[] args)
{
  Bisiklet bisiklet1 = new Bisiklet();
  bisiklet1.BilgileriYaz();

  Bisiklet bisiklet2 = new Bisiklet();
  bisiklet2.BilgileriYaz();


  Bisiklet cocukBisikleti = new Bisiklet();
  cocukBisikleti.BilgileriYaz();

  Bisiklet dagBisikleti = new Bisiklet();
  dagBisikleti.BilgileriYaz();
}
```

Ekran çıktısı:

Bisiklet Sayısı: 1 Vites: 2 Hız: 0
Bisiklet Sayısı: 2 Vites: 2 Hız: 0
Bisiklet Sayısı: 3 Vites: 2 Hız: 0
Bisiklet Sayısı: 4 Vites: 2 Hız: 0

Tüm Bisiklet nesneleri *BisikletSayisi* alanını paylaşır. Bu sebeple *BisikletSayisi++* ifadesi her yeni örnek oluşturulduğunda aynı veriyi bir artırır. Ayrıca bu alan aşağıdaki şekilde de yazdırılabilir:

```
Console.WriteLine("Bisiklet Sayısı: {0}", Bisiklet.BisikletSayisi);
```

2.2. “const” ile Static Bir Alan Oluşturma

Bir alanın hem statik olduğunu hem de değerinin hiçbir zaman değiştirilmeyeceği *const (constant : sabit)* anahtar sözcüğü ile belirtilir. *Const* bir alan bildiriminde ayrıca *static* anahtar kelimesi kullanılmalıdır. Aşağıda *Math* sınıfının *PI* adındaki *const* alanın tanımlaması görülmektedir.

```
public const double PI = 3.14159;
```

2.2.1. Static Sınıflar

Programlama dilinde sınıflar da *static* olarak tanımlanabilir. *Static* sınıflar diğer sınıflardan farklı olarak sadece *static* alanlar ve metotlar içerebilir. *Static* sınıflar, uygulama içerisinde pek çok durumda işe yarayacak yararlı metot ve alanları bir arada tutmak için kullanılır. *Static* olarak tanımlanmış sınıflardan *new* anahtar kelimesi ile yeni bir nesne örneklemek istendiğinde derleyici hatası ile karşılaşılır. Math sınıfının sadece *static* üyeler içeren sürümü aşağıdaki gibi tanımlanabilir:

```
public static class Math
{
  public const double E = 2.71828;
  public const double PI = 3.14159;
  public static int Max(int val1, int val2) {. . .}
  public static byte Min(int val1, int val2) {. . .}
  public static double Pow(double x, double y) {. . .}
}
```

2.2.2. Anonim Sınıflar

Sınıf adı kullanmadan sadece *new* anahtar sözcüğünün devamında küme parantezleri içerisinde alan tanımlaması yaparak oluşturulan sınıflara anonim sınıflar denir. Aşağıdaki örnekte *ogretmen* değişkenine anonim bir sınıf değer olarak aktarılmıştır.

```
var ogretmen = new { Ad = "Kemal", Yas = 33 };
```

Anonim sınıflar metot tanımlaması içeremez ve sahip oldukları alanların hepsi *public* olmalıdır. Ayrıca bütün alanlara bir başlangıç değeri verilmelidir.

UYGULAMA FAALİYETİ

Sınıf tanımlamalarını hatasız olarak uygun yerlerde yazınız.

İşlem Basamakları	Öneriler
➢ Nesne tabanlı programlama yazılımını başlatınız.	➢ Örnek projelerinizi kendinize ait bir klasör içerisinde oluşturursanız başka ortamda çalışmanız daha kolay olacaktır.
➢ StaticOrnegi adında yeni bir konsol uygulaması oluşturunuz.	➢ Programlama dilini seçmeyi unutmayınız.
➢ Solution Explorer'da Program.cs dosyasına çift tıklayınız.	➢ Solution Explorer pencerisi görünmüyorsa View menüsünü kullanınız.
➢ Program sınıfında Main metodunu bulunuz ve bu metodun üst kısmında Nokta sınıfını aşağıdaki gibi oluşturunuz. *class Nokta* *{* *private int x, y;* *private static int noktaSayisi = 0;* *public Nokta()* *{* *this.x = 0;* *this.y = 0;* *noktaSayisi++;* *}* *public Nokta(int x, int y)* *{* *this.x = x;* *this.y = y;* *noktaSayisi++;* *}* *}*	➢ Yazdığınız sınıfın Program sınıfının içinde olmasına dikkat ediniz.
➢ Nokta sınıfına aşağıdaki static metodu da ekleyiniz. *public static int NoktaSayisi()* *{* *return noktaSayisi;* *}*	➢ Yazdığınız metodun Nokta sınıfının içinde olmasına dikkat ediniz.

➢ Main metodunun içerisinde üç tane nokta nesnesi oluşturunuz.	➢ Oluşturduğunuz nesnelere anlamlı isimler veriniz (nokta1, nokta2 vb.).
➢ NoktaSayisi() metodunu kullanarak kaç nokta oluşturulduğunu ekrana yazdırınız.	➢ NoktaSayisi() metodunun statik olduğunu unutmayınız.

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet,** kazanamadıklarınız için **Hayır** kutucuklarına (X) işareti koyarak öğrendiklerinizi kontrol ediniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Nesne tabanlı programlama yazılımını doğru şekilde başlattınız mı?		
2. Konsol uygulamasını belirli bir klasörde oluşturdunuz mu?		
3. Program.cs dosyasını kod düzenleyicide açtınız mı?		
4. Sınıf tanımlamasını doğru yerde yaptınız mı?		
5. Statik metodu doğru şekilde tanımladınız mı?		
6. Nesne örneklerini oluşturdunuz mu?		
7. Statik metodu kullanarak nokta sayısını yazdırdınız mı?		

DEĞERLENDİRME

Değerlendirme sonunda **"Hayır"** şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız **"Evet"** ise "Ölçme ve Değerlendirme"ye geçiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

1. …… metotları kullanmak için nesne oluşturmamıza gerek yoktur.

2. Statik alanlar o sınıftan oluşturulmuş bütün nesneler tarafından ……

3. Uygulama içerisinde pek çok durumda işe yarayacak yararlı metot ve alanları bir arada tutmak için …… kullanılır.

4. …… içerisinde metot tanımı bulunduramaz ve sahip oldukları alanlara başlangıç değeri verilmek zorundadır.

Aşağıdaki cümlelerin başında boş bırakılan parantezlere, cümlelerde verilen bilgiler doğru ise D, yanlış ise Y yazınız

5. ( ) Statik metotları kullanmak için nesnenin bir örneğini oluşturmak gerekir.

6. ( ) *const* anahtar cümlesi ile tanımlanmış bir alanın değeri değiştirilemez ve o alan *static* olur.

7. ( ) Static sınıflar, new anahtar kelimesi ile oluşturulur.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise "Modül Değerlendirme"ye geçiniz.

MODÜL DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Aşağıdaki anahtar sözcüklerden hangisi sınıfın yeni bir örneğini oluşturmak için kullanılır?
A) class
B) new
C) public
D) private

2. Aşağıdakilerden hangisi Oyun adlı bir sınıfın varsayılan kurucu tanımı olabilir?
A) public Oyun(int x, int y)
B) public int Oyun()
C) public Oyun()
D) public void Oyun()

3. Oyun adında bir sınıftan aşağıdaki ifade ile bir nesne oluşturulurken aşağıdaki kuruculardan hangisi çalıştırılır?

Oyun yeniOyun = new Oyun(400, 500);

A) public Oyun(int x, int y)
B) public Oyun(double x, int y)
C) public Oyun(double x, double y)
D) public Oyun()

4. Aşağıdaki alan bildirimlerinden hangisi sınıf dışından erişilebilir bir alan tanımlar?
A) int a = 0;
B) private b;
C) int c;
D) public int d = `0;`

5. Statik bir sınıf içerisine yazılan aşağıdaki tanımlamalardan hangisi derleyici hatasına neden olur?
A) private int b;
B) public const double PI = 3.14159;
C) public static BilgileriYaz();
D) public static a = 1;

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki modüle geçmek için öğretmeninize başvurunuz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1'İN CEVAP ANAHTARI

1	**Sınıf**
2	**Örnekleme**
3	**Alanlar**
4	**Metotlar**
5	**Kapsülleme**
6	**D**
7	**C**
8	**B**

ÖĞRENME FAALİYETİ-2'İN CEVAP ANAHTARI

1	**Static**
2	**Paylaşılır**
3	**Static sınıflar**
4	**Anonim sınıflar**
5	**Yanlış**
6	**Doğru**
7	**Yanlış**

MODÜL DEĞERLENDİRMENİN CEVAP ANAHTARI

1	**B**
2	**C**
3	**A**
4	**D**
5	**A**

KAYNAKÇA

- Sharp John, **Adım Adım Microsoft C# 2008**, Arkadaş Yayınevi, Ankara, 2009.