WEB PROGRAMLAMA II

Öğr. Gör. M. Mutlu YAPICI

Ankara Üniversitesi
Elmadağ Meslek Yüksekokulu

Öğr. Gör. M. Mutlu YAPICI

Ders İzlencesi

Hafta	Modüller/İçerik/Konular
1. Hafta	Oturum yönetimi
2. Hafta	Cookies kullanımı ve oturum yönetimi
3. Hafta	Session kullanımı ve oturum yönetimi
4. Hafta	Sayfalama ve Arama İşlemleri
5. Hafta	JavaScript, JQuery ve PHP
6. Hafta	AJAX ve PHP
7. Hafta	AJAX ve PHP
8. Hafta	ARA SINAV
9. Hafta	PHP'de Nesne Yönelimli Programlamaya Giriş(347)
10. Hafta	PHP de PDO
11. Hafta	
12. Hafta	
13. Hafta	
14. Hafta	

Bu Ünitede Ele Alınan Konular

- Nesne tabanlı programlama ve tanımı
- Class, Method, Fonction, KeyWord kavramları
- Class ve Method oluşturma
- Access Modifiers (Erişim Düzenleyiciler)
- Kurucu ve yıkıcı metodlar

Öğr. Gör. M. Mutlu YAPICI

OOP

Object Oriented Programming
Nesneye Yönelimli Programlama

Aslında daha önceki derslerde OOP nedir neden kullanılır bahsetmiştik. Eğer biraz olsun programlama ile ilgilendiyseniz nesne tabanlı programlama nedir tam olarak bilmeseniz de bu ismi duymuşsunuzdur. Nesne tabanlı programlama offffffff bunu sürekli yazmak çıldırtıyor bundan sonra kısaca OOP diyeceğim. ☺ OOP aslında programlama işlemi sırasında yazdığımız kodlara bir ahenk ve düzen kazandırır ve bizim kodları daha rahat analiz ederek anlamamızı kolaylaştırır. Aslına bakarsanız OOP 'yi bizim yaşam şeklimize benzetebiliriz. Nasıl Mı?

Düşünecek olursak ilk insanlar her işlerini kendileri yaparmış, yani kendi evlerini, tabaklarını, bıçaklarını ve ihtiyaçları olan her şeyi teker teker kendileri yaparmış. Tabi ki bu oldukça zaman alıcı ve uğraştırıcı bir işlem aynı zamanda hiçte verimli değil. Zamanla yerleşik hayata yani profesyonel hayata geçildikçe, belirli alanlarda uzmanlaşma başlamış ve insanlar sadece uzman oldukları yani iyi bildikleri belirli bir alanda çalışıp uğraşmaya başlamışlar. Yeni bir tabak gerekirse bunu kendileri yapmakla uğraşarak zaman kaybetmek yerine daha kaliteli birşekilde tabak üreten insanlardan almışlar bu şekilde herkes uzmanlaştığı alanda çalışarak daha profesyonel, daha kaliteli ve seri üretimi olan ürünler ortaya çıkarmaya başlamış.

OOP

Object Oriented Programming
Nesneye Yönelimli Programlama

Lafı daha fazla uzatmayalım işte OOP de aynen bu insanlar örneğinde olduğu gibi belirli kod kümelerini belirli görevleri yerine getirmek için ayrıştırmamızı ve özelleştirmemizi sağlar. Aynen insanlarda olduğu gibi belirli bir ürün için kod fabrikaları kurabilmemize olanak sağlıyor. Böylece belirli işler için belirli metodlar (kod fabrikaları) oluşturabiliyoruz. Daha sonra **hangi işleme ihtiyacımız var ise o işleme ait kodları tekrar tekrar yazmak yerine işleme ait tanımladığımız metodu çağırarak kolay ve hızlı bir biçimde kullanabiliriz**.

Hatta tıpkı fabrikalarda olduğu gibi nasıl bir otomobil fabrikası teknolojinin **gelişmesine** bağlı kendi bünyesine yeni özellikler katabiliyor ve kendini geliştirebiliyorsa bizde OOP ile kendi yazdığımız metodlara yeni özellikler ekleyerek sürekli programımızı geliştirebiliriz. Bu da kodlarımıza ayrı bir dinamiklik kazandırmaktadır.

OOP

Sanırım OOP genel anlamıyla nedir? Neden kullanırız anlaşıldı. Şimdi biz bu OOP mantığını kod alanına nasıl taşıyacağız nelere ihtiyacımız var bir bakalım.

Kodlama alanında namespace ler altında oluşturduğumuz **her bir Class (sınıf) gerçek dünyadaki bir fabrikanın temsili olarak düşünülebilir**. Haliyle de bu fabrikamızdan bizler nesne üretip ihtiyacımız olan heryerde kullanabiliriz. Peki ama bir Class ı nasıl tanımlayabiliriz? Hangi fonksiyonları içereceğine nasıl karar verebiliriz?

Oluşturacağımız class fabrikayı temsil ettiğine göre bir fabrika kurma aşamasına gelmişsek biz ne üreteceğimize karar vermişiz demektir. İşte programlamada da bir class oluşturacaksak bu class ı neden oluşturuyoruz nasıl özellikleri olmalı ve ne gibi fonksiyonları bünyesinde barındırmalı karar vermemiz gerekmektedir.

OOP

Temel olarak OOP ne olduğunu anladığımıza göre şimdi programlama dilinde OOP nın temel özelliklerine bakalım. Nesne Yönelimli Programlama Teorisi'nde 4 temel özelliğin gerçekleştirilmesi zorunlu sayılmıştır ve bu temel özelliklerden birini bile sağlamayan bir dil saf (pure) Nesne Yönelimli Programlama Dili sayılmaz.

Bu 4 temel özellik;

1) Sarmalama / Paketleme (Encapsulation)
2) Miras Alma (Inheritance)
3) Soyutlama (Abstraction)
4) Çok Biçimlilik (Polymorphism)

OOP

Bu 4 temel özelliği basitçe açıklayacak olursak;

Soyutlama(Abstraction), problemi çözerken ilgili metodların belli parçalar halinde yazılarak kodların basit ve anlaşılır hale getirilmesidir diyebiliriz.

Çok biçimlilik (Polymorphism) ise, yazdığımız nesnenin yada fonksiyonun farklı veri yada giriş değerleri ile kullanılabilmesidir.

Sarmalama (Encapsulation) ise, yazdığınız sınıf içerisindei metodların dışarıdan erişilebilir olup olmamalarını sağladığımız bir çeşit güvenlik katmanıdır diyebiliriz.

Son olarak **Kalıtım** (Inheritance) ise, var olan sınıfların kalıtım (miras alınarak) yeniden/istenildiği kadar oluşturulabilmesidir.

OOP Nesne

- Nesneler (class) PHP OOP yapısının temel taşıdır. Aslında zaten OOP yapısının temelinde de nesneler vardır. Günümüzde OOP yapısını destekleyen diller için de bu öğreneceklerimiz çoğunlukla geçerli olacaktır.

- Nesneler yani class'lar, sizin tarafınızdan yapmak istediğiniz programın amacına uygun olarak yazılır yada başkası tarafından önceden yazılmış nesneler baz alınarak kullanılabilir. Bu nesneler kalıtım olayı ile oluşturulup kullanılır.

OOP Nesne

Yeryüzü üzerinde bulunan her şey bir nesnedir. Bilgisayar yazılımı açısından nesne, bir varlığın temsil biçimidir. Örneğin; bina bir nesnedir. Binayı bilgisayar yazılımı içinde temsil etmek istersek, öncelikle bu nesnenin yani binanın ayırt edici özelliklerini belirlemeliyiz.

Örneğin;

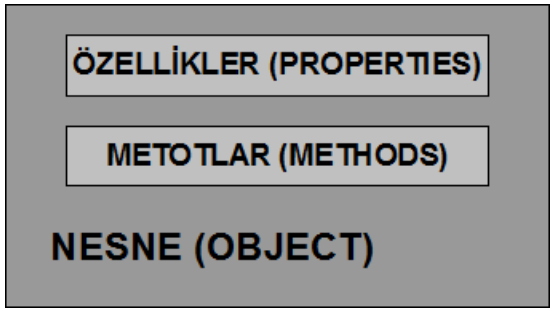
- Bina yüksekliği
- Dış yüzey rengi
- Kat sayısı
- Zemin alanı
- Zemin boyutları

gibi özellikler, binayı temsil etmek için ilk akla gelen birkaç özelliktir. O halde nesnenin bilgisayarda temsilinde, özellikleri kilit rol oynamaktadır.

OOP Nesne

Nesnenin karakteristikleri, özellikler ve gerçekleştirebileceği eylemler de metot adını alır. Bu anlamda, soyutlaştırmanın sonucunda bir nesne;

- Özellikleri (properties, member variables - Objective-C)
- Metotları (methods) ile temsil edilebilir.



Nesne Yönelimli Programlama Dilleri, soyutlamayı sınıf (Class) yapısı ile gerçekleştirirler. Sınıf yapısı içinde o nesneye ait özellikler ve metotlar tanımlanabilir.

Ancak sınıf soyut bir yapıdır ve doğrudan kullanılamaz. O sınıftan üretilen örnekler sınıfa ait tüm özelliklere ve metotlara sahip olurlar. Bunlar program içinde doğrudan kullanılabilirler. Sınıftan üretilen her örnek, aynı özellik ve metotlara sahip olacak ancak özelliklerin değerleri doğal olarak farklı olabilecektir.

Örneğin, insanı bir nesne olarak düşünürsek ve insan sınıfı olarak soyutlarsak, ilk akla gelen özellikleri; boy, kilo, IQ ve EQ değerleri, eğitim düzeyi vb. gibi özellikler olur. Bu özelliklerin değerleri doğal olarak her insan için farklıdır. İnsan için ilk akla gelen eylemler (metotlar) ise; yürüme, okuma-yazma, konuşma, araç kullanma vb. gibi metotlardır.

OOP

Şimdi bir proje düşünelim ve buna ait class tanımlamasını, değişkenleri ve fonksiyonları tanımlayalım.

Bir araba fabrikası kurmak istediğimizi düşünelim. Bunun için ne tür özellikleri tanımlamaya ihtiyacımız vardır?

- Marka
- Tip
- Malzeme
- Motor gücü
- Renk
- Yakıt türü

Gibi bir çok özelliğe ihtiyacımız vardır. Evet fabrikamızın özelliklerine karar verdi hadi şimdi onu inşa edelim ☺.

Sınıf (Class) Ekleme

- Nesne tabanlı programlamayı iyi kavramak için öncellikle sınıflar ve nesneler arasındaki ilişkiyi iyi bilmek gerekir. Classlar kısaca nesneleri oluşturmak için kullanılan kod şablonlarıdır. Hemen bir class tanımlayalım .

```
<?php
    class otomobil
    {
    }
?>
```

Kullanımı ise ;

```
    $ford = new otomobil();
    $ford = new otomobil;
```

Yanlış kullanım şekli:

```
    $ford = new 'otomobil';
```

Buradaki "new" ifadesini sınıf dışından bir nesne yaratmak için kullanıyoruz

Özellik (Attrebute) Ekleme

Sınıflar içerisinde fonksiyonlarımız veya metodlarımız tarafından kullanılabilecek özel değişkenler yani özellikler (**attrebutes**) tanımlayabiliriz. Örnek vermek gerekirse bir ürünün bir ismi ve fiyatı olmalı. Bu özellikleri Erişim belirteçlerine göre ekliyoruz (public,private,protected).Bu Erişim belirteçlerini daha sonra detaylı öğreneceğiz. Örnek bir özellik eklenmiş sınıf ;

```
class Urun {
        public $isim = "Masa";
        public $yapim = "X Firması";
                $fiyat = 77;
 }
```

Yukarıda gördüğünüz gibi sınıfmızda 3 özellik tanımladık. Şimdi bu özelliklere nesnelerimizde nasıl ulaşırız ona bakalım ..

```
$nesne1 = new Urun();
echo $nesne1->isim;
echo $nesne1->fiyat;
```

Ayrıca class içinde özelliğimiz **public** tanımlandığı için nesnelerimizde bu özellikleri okuyabilir veya yerine yeni bir değer verebiliriz.

```
$nesne1 = new Urun();
$nesne1->isim="Sandalye";
$nesne1->fiyat=25;
```

Methods (Metodlar)

- Metotlar bir program içerisinde aynı işi gerçekleştiren satırları belirli düzende sadece birkez oluşturarak gerektiğinde tekrar tekrar kullanabilmemizi sağlayan alt programlardır. Metotlar sayesinde kod tekrarları yapmadan daha anlaşılır bir şekilde kodlama yapılabilmektedir.

- Metotlar direk olarak çalışmazlar ancak program içerisinde çağrılarak çalıştırılabilirler. Bir metot bir amacı gerçekleştirmeye yönelik olarak yazılır. Metot içerisinde bir veya daha fazla ifade bulunabilir.

Methods (Metodlar)

- Programlama dili içerisinde bir veya birden fazla kod bulunduran kod bloklarıdır. Bir nesnenin veya sınıfın programı yürütmesi için kullandığı bir nesnesidir.
- Program içerisinde yürütülen kodlar bu metotlar içerisinde bulunur.
- Program yazarken belirli bir işi yapan kod bloğunu bir kaç kez kullanmak gerekebilir. Bunun için aynı kod bloğunu tekrar yazmak yerine bir metot olarak hazırlanır ve ihtiyaç duyulduğunda kullanılır. Böylece zamandan tasarruf sağlanır

Methods (Metodlar)

Kullanım ve Tanımlama Şekilleri

Metotların yapısal olarak bir takım özellikleri vardır. Bunlar;

- Erişim seviyeleri (Access Modifiers)
- Geriye değer döndürmesi
- Metod adı
- Aldığı parametreler

Bu özellikler bir metodu tanımlamada kullandığımız ve dikkat etmemiz gereken özelliklerdir.

Methods (Metodlar)

Kullanım ve Tanımlama Şekilleri

Her bir metot tanımlanırken metota bir isim verilir. Metot çağrılırken verilen isim kullanılarak çağrılır.

Bazı metotlar dışarıdan değer alabilirler. Dışarıdan gönderilen değerlere **parametre** denir.

Yapılan işlem sonucunda bazı bilgiler metotlar ile metodun çağrıldığı bölüme tekrar gönderilebilir. Bu metotlar geri değer döndüren metotlardır. Bu değerlere **return** yada **geri dönüş değeri** denir.

Tanımlama Şekilleri (Syntax)

```
<ErişimBelirleyicisi >   < Dönüş-tipi>   <MetotAdi>  (  parametre listesi )
{
        //metot içerisinde gerçekleştirilecek işlemler ;
}
```

Methods (Metodlar)

- **DönüşTürü:** Bir tür adıdır ve yöntemin hangi türden bilgi döndüreceğini belirtir. Bu int ya da string gibi herhangi bir tür adı olabilir. Bir değer döndürmeyen yöntem yazılıyorsa, dönüş türü yerine void yazılır.
- **Metod Adı:** Yöntemi çağırmak için kullanılacak addır. Yöntem adları da değişken adlarının uyması gereken tanımlayıcı kurallarına uymalıdır. Örneğin addValues geçerli, add$Values geçersiz birer yöntem adıdır. Yani camelCase yöntem adlarını kullanın; yöntem adlarını açıklayıcı olarak verin ve mümkünse bu yapıya uymak için bir yüklem ile başlayın; örneğin, displayCustomer.
- **ParametreListesi:** Yöntemin kabul edeceği ve seçime bağlı bilgilerin türünü ve adını tanımlar. Parametreleri parantez içinde değişken tanımlaması yapar gibi önce tür adı sonra da parametre adı olacak biçimde bildirebilirsiniz. Yazılan yöntemin iki ya da daha fazla parametresi var ise bunları virgülle ayırmalısınız.
- **Metod gövdesi:** Yöntem çağrıldığında çalıştırılacak satırlardır. Blok (küme) parantezleri arasına yazılır.

Metod (Method) Ekleme

Özellikler nesne verilerimizi değişkenlerde tutmaya izin verirken ayrıca metodlarda sınıf içinde tanımlanan ve verilen görevi gerçekleştirmeye yarayan özel fonksiyonlardır. OOP'un sarmalama (**encapsulation**) özelliği ile erişim yapısı istediğimiz gibi şekillendirilebilir. Özellikle büyük projelerde ve sınıflarda sarmalama özelliğini iyi kullanmak ve planlamak gerekir.

```
class daire {

    public $yaricap; ////özellik

    function yaricapAyarla ($yaricap) {  ///metod
        $this->yaricap = $yaricap;
    }

    function alanHesapla() { ///metod
        return $this→yaricap * $this->yaricap * M_PI;
    }
}

$c = new daire();
$c->yaricapAyarla(5);

echo $c->alanHesapla(), "n";
```

Yukarıdaki örnekte **daire** isminde bir sınıfımız ve içinde 2 metodumuz var. **alanHesapla** isminde bir metodumuz var. Normal bir PHP fonksiyonu farkı yok, ancak sınıflar içerisinde tanımlanan fonksiyonlara method diyoruz. **$this** değişkeni ise özel bir değişkendir. Sınıf içerisindeki değişkenlerimize (member fields) ulaşmak için kullanıyoruz. **alanHesapla** metodumuz gerekli hesaplamaları yaparak **return** ile sonucu döndürmekte.

KAYNAKLAR

- İnternet ortamı
- PHP ve AJAX Haydar TUNA
- A'dan Z'ye PHP Rıza ÇELİK