Nesne Yönelimli Yaklaşım

Nesne(object): verileri ve bu veriler üzerinde işlem yapan üye fonksiyonları birleştiren yapı.

Üye fonksiyonlar o nesnenin verilerine erişmeyi sağlayan tek yoldur.
Veriye doğrudan ulaşılmaz.

Verilerin paketlenmesi(encapsulation) : verilerin ve üye fonksiyonların tek bir çatı altında paketlemesidir.

Veri gizliliği(data hiding) : Nesne içerisindeki veriler üye fonksiyonlarla erişildiğinden, veriye doğrudan erişilmediğinden veriler korunmuş olur buna veri gizliliği denir.

Nesne Yönelimli dillerin özellikleri

Nesneler: nesne yönelimli bir dilde programlama problemini çözmemiz gerektiğinde, problemin fonksiyonlara nasıl bölüneceği değil nesnelere nasıl bölüneceği düşünülmelidir.

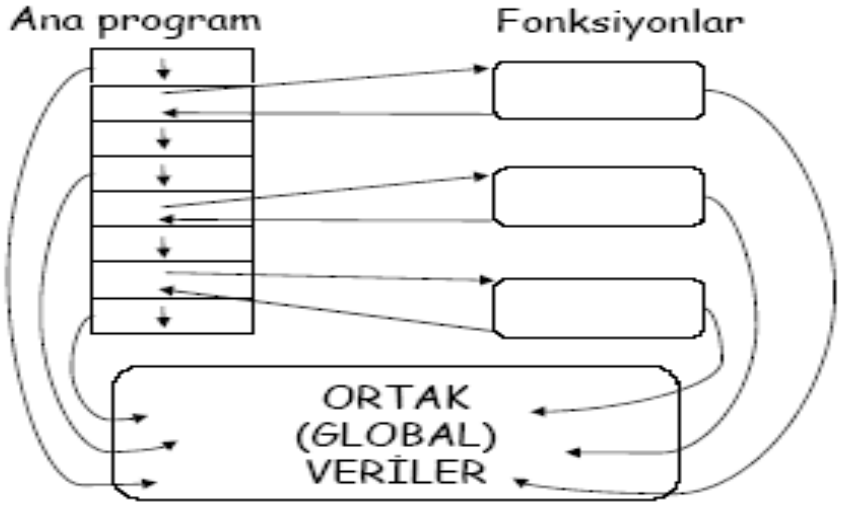
Sınıflar : Bir dizi benzer nesnenin genel tanımıdır. Örneğin taşıt sınıfının nesneleri, kamyon, otobüs, minibüs vs dir.

Kalıtım: Miras alma(inheritance) olayıdır. Örneğin taşıt sınıfının üyeleri araba, kamyon, otobüs vs dir. Bunlarda birer sınıf oluşturmaktadırlar örneğin araba sınıfında, çok çeşitli model ve markada araba nesneleri mevcuttur. Yani bir sınıftan, alt sınıflar türetilebilir. Alt sınıf üyesi olduğu sınıfın özelliklerini miras alır(taşır).
Yeniden kullanılabilirlik: bir sınıfı hatasız olarak oluşturduktan sonra, diğer programcılara kendi programlarında kullanmaları için dağıtılabilir.

Çok biçimlilik(polymorphism): operatörler ve fonksiyonlar işlevlerine bağlı olarak farklı şekillerde kullanılabilir.

Emir Esaslı(Procedural) Programlama Yöntemi

- Basic, Fortran, Pascal, C gibi programlama dillerinin desteklediği bu yöntemde öncelikle gerçeklenmek istenen sistemin yapması gereken iş belirlenir.
- Büyük boyutlu ve karmaşık işler, daha küçük ve basit işlevlere (fonksiyon) bölünerek gerçeklenirler.



Emir Esaslı Programlama Yönteminin Değerlendirmesi

- “Böl ve yönet” prensibine dayanır. Amaç büyük programları küçük parçalara bölerek yazılım geliştirme işini kolaylaştırmaktır.
- Ancak yazılımların karmaşıklıkları sadece boyutlarından kaynaklanmaz. Küçük

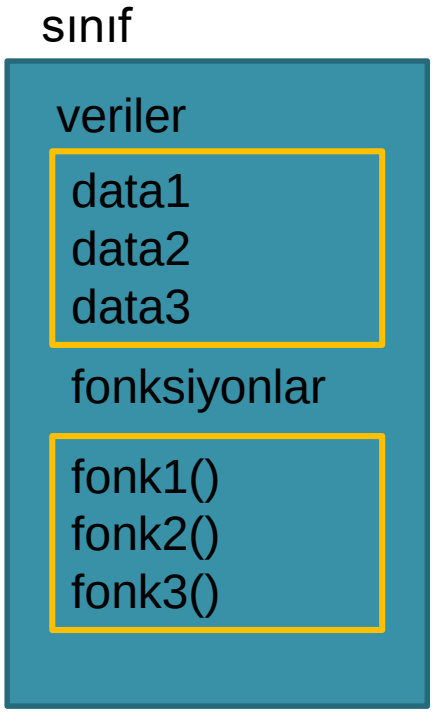
  problemler de karmaşık olabilir.
- Gerçek dünyadaki sistemler sadece fonksiyonlardan oluşmaz. Dolayısıyla emir esaslı yaklaşımda karmaşık bir problemin gerçeğe yakın bir modelini bilgisayarda oluşturmak zordur.
- Tasarım aşamasında verilerden çok fonksiyonlara odaklanıldığından hatalar nedeniyle veri güvenliği tehlikeye girebilmektedir.
- Kullanıcılar kendi veri tiplerini çok güçlü biçimde tanımlayamazlar.
- Programın güncellenmesi zordur.
- İşleve dayalı yöntemi de kullanarak kaliteli programlar yazmak mümkündür.

Ancak nesneye dayalı yöntem kaliteli programların oluşturulması için programcılara daha çok olanak sağlamaktadır ve yukarıda açıklanan sakıncaları önleyecek yapılara sahiptir.

Nesneye Dayalı (Object-Oriented) Programlama Yöntemi

- Gerçek dünya nesnelerden oluşmaktadır.
- Çözülmek istenen problemi oluşturan nesneler, gerçek dünyadaki yapılarına benzer bir şekilde bilgisayarda modellenmelidir.
- Nesnelerin yapıları iki bölümden oluşmaktadır: **1.** Nitelikler(özellikler ya da durum bilgileri), **2.** Davranışlar(yetenekler)
- Tasarım yapılırken sistemin işlevi değil, sistemi oluşturan **veriler** esas alınır.
- Aşağıdaki elemanlar nesne olarak modellenebilir:
- İnsan kaynakları ile ilgili bir programda; memur, işveren, işçi, müdür, genel müdür.
- Grafik programında; nokta, çizgi, çember, silindir.
- Matematiksel işlemler yapan programda; karmaşık sayılar, matris.
- Kullanıcı arayüzü programında; pencere, menü, çerçeve.

Veri ve fonksiyonları tek bir bütün haline getirmek nesne yönelimli programlamanın temel yaklaşımıdır.



Sınıf ve nesneler

Bir nesne ile sınıf arasındaki ilişki tıpkı bir değişken ile veri tipi arasındaki ilişki gibidir. Bir nesne kendi sınıfının bir örneğidir.

Birprogramda sınıf (basitnesne)önce tanımlanacak , main() fonksiyonu içerisinde sınıfa ait nesneler (b1,b2) üretilecektir.

Sınıf oluşturma class anahtar kelimesi ile başlar, ardından sınıfın adı gelir. Yapı tanımlamasında olduğu gibi sınıfın gövdesi küme parantezleri ile ayrılır ve sonuna (;) konur.

Private(özel): private olarak tanımlanan veri ve fonksiyonlara sadece, tanımlı oldukları sınıf içerisinden erişilebilir.

Public(genel): public olan veri ve fonksiyonlara tanımlı oldukları sınıfların dışındanda erişmek mümkündür.

Genellikle bir sınıfdaki fonksiyonlar public, veriler private tır. Veriler üzerlerinde kazara oynanmasın diye private olarak tanımlanırlar. Üzerinde işlem yapılan fonksiyonlar ise sınıfın dışından erişilebilsin diye public olarak tanımlanırlar.
Bu bir kural değildir.

Yöntemin Değerlendirmesi:

- Gerçek dünya nesnelerden oluştuğundan bu yöntem ile sistemin daha gerçekçi bir modeli oluşturulabilir. Programın okunabilirliği güçlenir.
- Nesne modellerinin içindeki veriler sadece üye fonksiyonların erişebileceği şekilde düzenlenebilirler. Veri saklama (data hiding) adı verilen bu özellik sayesinde verilerin herhangi bir fonksiyon tarafından değiştirilmesi önlenir.
- Programcılar kendi veri tiplerini yaratabilirler.
- Bir nesne modeli oluşturduktan sonra bu modeli çeşitli şekillerde defalarca kullanmak mümkündür (reusability).
- Programları güncellemek daha kolaydır.
- Nesneye dayalı yöntem takım çalışmaları için uygundur.

NESNEYE YÖNELİK PROGRAMLAMA DİLLERİ

- Programlama dilleri literatüründe sınıf ve alt sınıf kavramını ilk olarak SIMULA67dili tanıtmıştır.
- Smalltalk, Eiffel, ADA95 ve Java dillerinde tüm veriler nesne şeklindedir. Yani bu diller tam nesne yönelimlidir.
- C++ ise hem emir esaslı paradigmayı hem de nesneye yönelimli k paradigmayı destekler.

Smalltalk

- SIMULA67'de tanıtılan fikirler, Smalltalk ile güçlenmiş ve Smalltalk ile nesne yönelimli dil popüler hale gelmiştir
- Smalltalk'ta bir program sadece kalıtım hiyerarşisi içinde düzenlenmiş birbirleriyle mesajlar ile etkileşen nesne sınıflarından oluşabilir. Smalltalk'ta tüm veriler nesnelerle gösterilmek zorunda olduğu için tam nesneye yönelik bir programlama dili olarak nitelendirilir.
- Smalltalk dilinde bütün bağlamlar dinamik olarak gerçekleşir.
- Smalltalk'ta sınıfların sadece tek üst sınıfı bulunabilir (tekli kalıtım modeli).

C++

- C diline sınıf tanımlama, sınıf türetme, *public/private* erişim kontrolü, *constructor/deconstructor* ve metod yükleme özelliklerinin ve çoklu kalıtım, soyut sınıflar, sanal metodlar eklenesiyle elde edilmiş halidir.
- *Constructor*, bir nesne yaratıldığında bir kez çalıştırılan özel bir metod olmaktadır. Benzer şekilde bir nesne yok edildiği zaman bir *destructor* metodu varsayılan olarak çağrılır.
- Bir sınıf için bir veya daha fazla *Constructor* metodu tanımlanabilir.
- C++'da bağlama genellikle durağan olarak gerçekleşir.
- Ancak, *virtual* metodlar ve göstergeler, dinamik bağlama etkisi vermek için kullanılır. Metodların dinamik olarak bağlanabilmesi için metod, üst sınıfta *virtual* olarak tanımlanmalı ve daha sonra türetilmiş sınıflarda yeniden tanımlanmalıdır.

Java

- Java, çeşitli elektronik aygıtlara yazılım geliştirmek için geliştirilmiş bir programlama dilidir.
  - Java'da gösterge tipi yoktur.
- Yorumlayıcıya dayalı gerçekleştiriminden dolayı Java taşınabilirdir. Bir Java kaynak kodundan bytecode adı verilen bir ara kod üretilir ve bytecode yorumlayıcısının bulunduğu her makina bu programı çalıştırabilir.
- Birçok Internet tarayıcısı (browser) Java programlarını doğrudan yükleyebilir ve çalıştırabilir. Bu özelliği nedeniyle Java ağ programlama dili olarak nitelenmektedir.
- Java'da sınıflar arasında tekli kalıtıma izin verilmiştir. Ancak çoklu kalıtımı desteklemek için ayrı arayüz modülleri sağlanmıştır.
- Java'da programlamada eşzamanlılık (concurrency) önceden tanımlı olan thread sınıfı ile desteklenir.
- Dinamik bellek yönetimi için, otomatik bellek düzenleme (garbage collection) gerçekleştirilmektedir.

Java ve C++

- *public*, *protected* ve *private* tanımlayıcılar Java'da da geçerlidir. Java'da yeni nesne tiplerinin tanımlanması için sınıf yapısı vardır. Bir sınıfta, veri sahaları ve metodlar vardır. Bir sınıfın bir örneği, o nesneyi oluşturan sahaların kendine ilişkin kopyalarını içerir.

- Java'da da *constructor* ve *destructor* metodları her sınıf için tanımlanabilir. Metod yükleme constructor metodlarına da uygulanabilir.
- Java'da tüm sınıflar, *object* adlı kökten türetilmiş bir hiyerarşi ağacının düğümleridir. Her sınıf, *object* sınıfından türetilmiştir.
- Java, soyut sınıfların ve metodların tanımlanmasını destekler. Soyut bir metod, bir alt sınıf tarafından gerçekleştirilmelidir.
- Java'da tekli kalıtıma izin verilir. Ama aynı zamanda alt sınıflamanın kısıtlanması da olasıdır. Java'da *public, private, protected* tanımlarına ek olarak, kalıtım süreci değiştirilebilmesi için *static*, *abstract* ve *final* tanımlayıcıları bulunur.

static, *abstract* ve *final* tanımlayıcıları

- *static* özelliğindeki bir veri sahası, bir sınıfın tüm örneklerinde paylaşılır. *static* bir metod, sınıfın bir örneği yaratılmadan da çağrılabilir ve *static* bir metod, bir alt sınıfta yeniden tanımlanamaz.
- *abstract* olarak tanımlanmış bir sınıf örneklenemez ve sadece üst sınıf olabilir. Bir *abstract* metod ise bir alt sınıf tarafından gerçekleştirilmek zorundadır.
- *final* olarak tanımlanmış bir sınıfın alt sınıfları tanımlanamaz ve *final* olarak tanımlanmış bir metod, herhangi bir alt sınıfta yeniden tanımlanamaz.

Özet

- Bu bölümde emir esaslı paradigma ile nesneye yönelik programlamanın temel özellikleri karşılaştırılmış ve
- Sınıf, Nesne, Kalıtım, Çokyapılılık gibi nesne yönelimli kavramlar tanıtılmıştır.
- Smalltalk, C++, Java gibi nesneye yönelik dillerin genel özellikleri anlatılmıştır.