

NESNE TABANLI PROGRAMILAMA

- Klasik programlamada veriler ve fonksiyonlar vardı. Her şey veri ve metotlarla yapılıyordu.
- Bu ders ile Algoritmik düşünmeyi geliştirmenizin yanı sıra nesne merkezli düşünme ile algoritmaların ve problemlerin modellenmesi gerçekleştirilecektir.

- İnsanlar açısından nesne, bir varlığın temsil biçimidir. Örneğin insan bir nesnedir. İnsanı bilgisayar yazılımı içinde temsil etmek istersek, insanın ayırt edici özelliklerini belirlemeliyiz;
 - Boy
 - Göz rengi
 - Kilo
 - IQ
 - Konuşma şekli

- Nesnenin karakteristikleri, özellikler ve gerçekleştirebileceği eylemler de metot adını alır. Böylece soyutlaştırma sonucunda nesne özellikleri ve metotlar ile temsil edilir.
- NTP de soyutlamayı sınıf yapısı ile gerçekleştiririz. Sınıf yapısı içinde o sınıfa ait özellikler ve metotlar tanımlanır.
- Sınıftan üretilen her örnek, aynı özellik ve metotlara sahip olacak ancak özelliklerin değerleri her örnek için birbirinden farklı olacaktır.

- NTP'nin 3 temel kavramı;
 - Kapsülleme (encapsulation)
 - Kalıtım (Inheritance)
 - Çok biçimlilik (Polimorphism)

Türler

- Türler ikiye ayrılır.
 - Değer türleri (Stack) (Sbyte, Byte, Char, Float, Decimal, Double.....)
 - Referans türleri (Heap) (Class, String, Array.....)

Sınıflar

- Her nesne, mutlaka sınıflardan türetilmektedir.

Class sınıf_ismi

{

veri_tipi değişken;

donus_degeri

metot_ismi(parametreler)

{

}

}

Örnek

- Öğrenci isminde bir sınıfımız olsun. Sınıfın isim adında bir özelliği ve isim_ne, ekrana_yaz isimli iki metodu olsun. Main kısmında bu sınıftan bir öğrenci türetilsin ve öğrenciye isim atansın.

- Nesne tanımlayarak sınıfın özelliklerini nesneye yüklemiş oluruz. Sonra bu nesne ile sınıf içindeki metot ve değişkenlere ulaşabiliriz.

Sınıf Üyeleri

- Veri üyeleri ve fonksiyon üyeleri olarak ikiye ayrılır.

Veri Üyeleri

- Sınıfın veri üyeleri değer türünde olabileceği gibi referans türünde de olabilir. Her nesnenin datası kendine aittir, hiçbir şekilde karışmaz. Veri üyeleri şu 3 öğeden oluşur;
 - Alanlar; nesneye ait bilgilerin tutulduğu üye değişkenler.
 - Sabitler; nesnenin değiştirilemeyen alanları.
 - Olaylar; belirli bir tetiklemenin meydana gelmesiyle çağrılacak olan fonksiyonların adres bilgilerini tutar.

Fonksiyon Üyeleri

- Şunlardan oluşur;
 - Yöntemler (metotlar); nesnenin davranış şekilleridir.
 - Özellikler; nesnenin üye verilerine değer atamasının ve değer çekilmesinin kontrollü olarak yapılmasını sağlar.
 - Yapıcılar (Constructors); nesne ilk oluşturulduğunda çağrılır.
 - Yıkıcılar (Destructor); nesnenin ömrü tamamlandığında çağrılır.
 - Operatörler; nesnenin daha esnek kullanılmasına izin verirler.
 - Dizinleyiciler; nesnenin bir dizi gibi kullanılmasına izin verirler.

Metotlar ve Dönüş Değer Tipleri

- Tüm metotlar bir sınıfın üyesi olmak zorundadır.
[Erişim] <dönüş değeri> metot_ismi (parametre listesi)
{
 metot gövde metni
}

- Bir metot kullanılırken bulunduğu sınıf türünden bir nesne tanımlanır ve “.” operatörü ile metoda ulaşılır.
- Geri dönüş değeri olmayan metotlar “void” olarak tanımlanır. “void” tanımlı bir metot için “return” kullanılamaz.
- “static” olarak tanımlanan metotları çağırmak için nesne tanımlanmasına gerek yoktur. Eğer bulunduğu sınıfta çağırılacaksa ismini yazmak yeterlidir, başka sınıftan çağırılacaksa “sinfadi.metot();” şeklinde çağırılır.

Özyinelemeli (Recursive) Metotlar

- Kendi kendini çağıran fonksiyonlara denir. Bu fonksiyonlara durabilmesi için bitiş koşulu verilmek zorundadır.
- Bazı ilginç problemlerin çözümü özyinelemeli fonksiyonlarla daha anlaşılır ve çabuk çözülebildiği için tercih edilmektedir.
- Matematiksel işlemlerin çoğu özyinelemeli olarak tanımlanmaktadır.

Fibonacci Dizisinin C Dilinde Recursive Örneği

```
#include <stdio.h>
#include <conio.h>
double fibonacci (double N)
{
    if (N<=2)
        return 1;
    else
        return (fibonacci(N-1)+fibonacci(N-2));
}
int main(int argc, char *argv[]) {
    int i;
    for (i=1;i<24;i++)
        printf("%d \n",fibonacci(i));
    return 0;
}
```

Erişim Belirleyiciler

- Erişim belirleyiciler ile nesne sahip olduğu üyelere kimin ve hangi şartla erişebileceğini belirler.
- Erişim Türleri:
 - Public; Bütün sınıflar erişebilir
 - Private; Alt sınıf dahil başka hiçbir sınıf erişemez
 - Protected; Alt sınıflar ve aynı pakettekiler erişebilir
 - Private Protected; Yalnızca alt sınıflar erişebilir
 - <default>; Aynı pakettekiler erişebilir