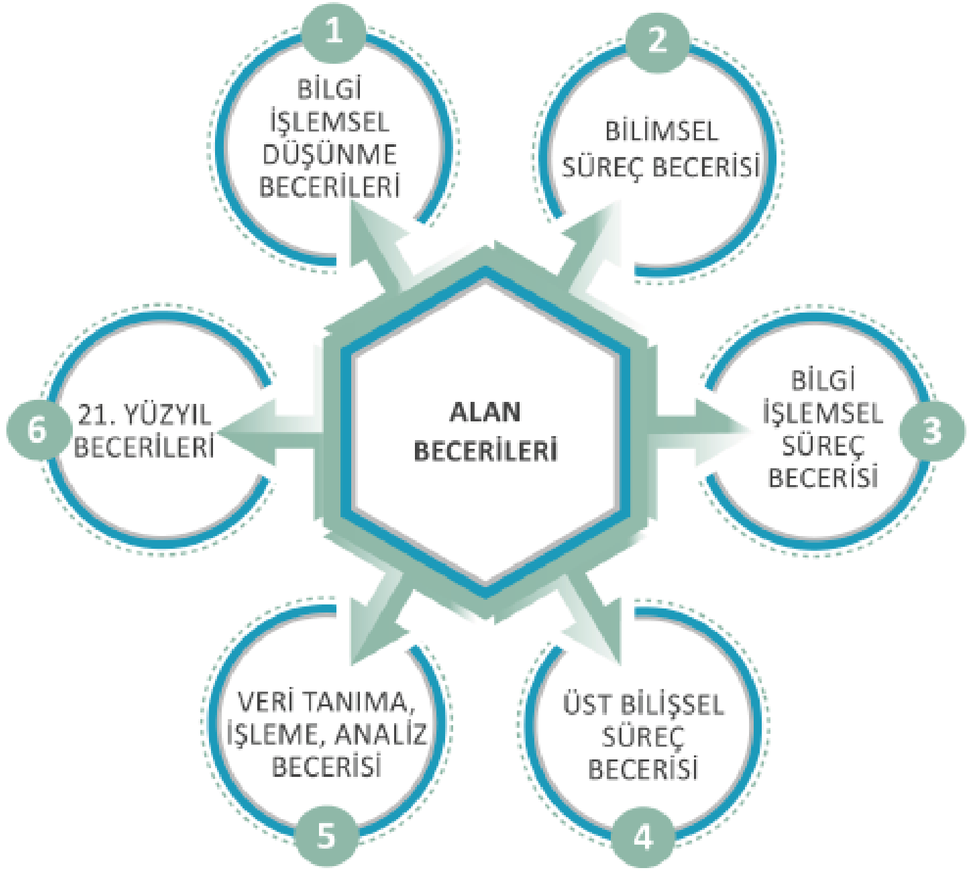
DİLEK ÖLMEZ TOKLUCA

BİLİŞİM TEKNOLOJİLERİ VE YAZILIM DERSİ ÖĞRETİM PROGRAMI

Programlama Dilleri(C# ve Python Programlama)

BİLİŞİM TEKNOLOJİLERİ VE YAZILIM DERSİ ÖĞRETİM PROGRAMI'NIN UYGULANMASI



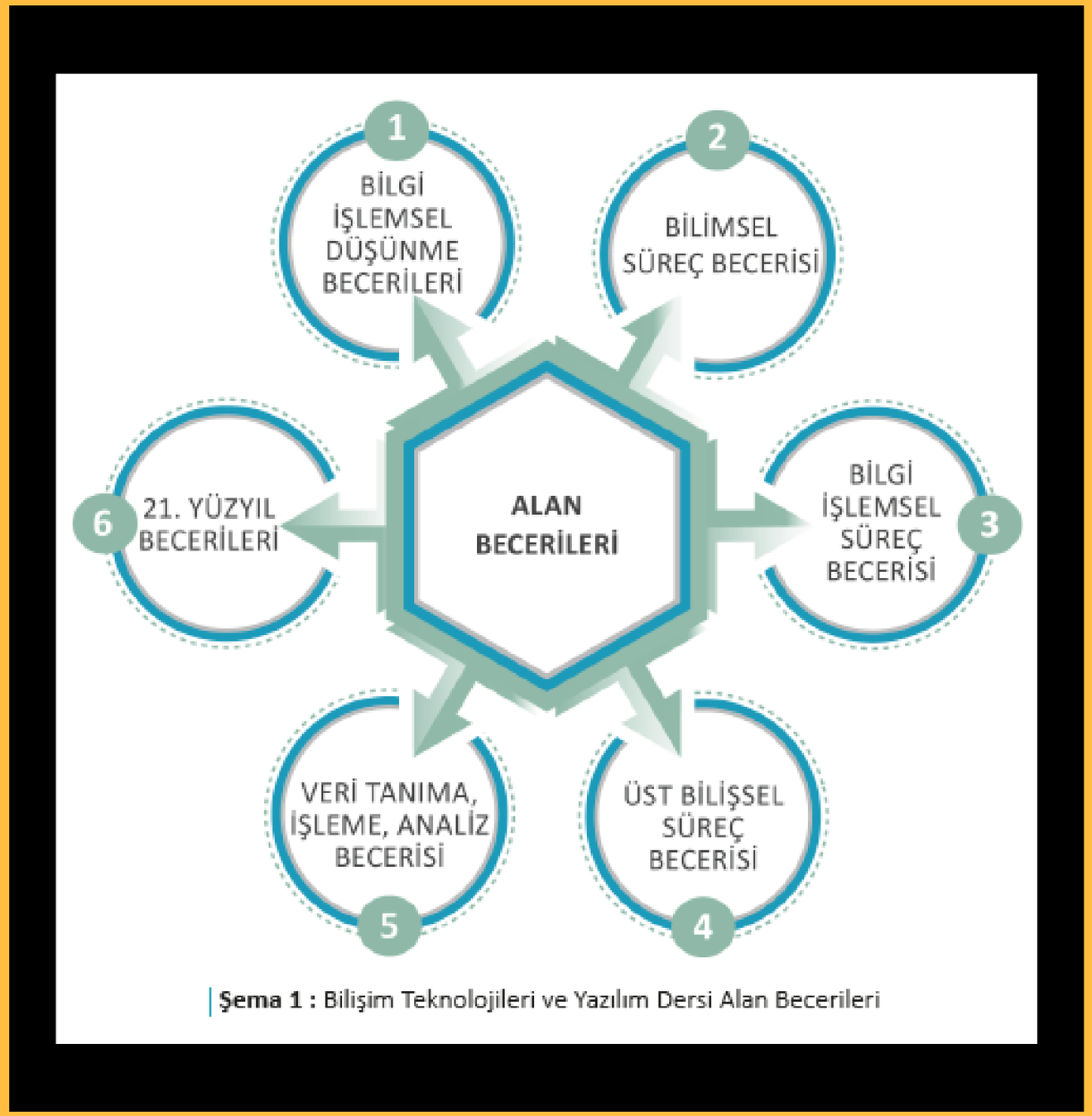
Şema 1 : Bilişim Teknolojileri ve Yazılım Dersi Alan Becerileri

Bilişim Teknolojileri ve Yazılım Dersi Öğretim Programı, bireylerin toplumsal ve mesleki hayatlarında başarılı olabilmeleri için gereken temel becerilere odaklanmaktadır.

Bu beceriler,

- bilgiye erişim ve günlük yaşam problemlerinin çözümünde kullanma,
- karmaşık görevleri yerine getirme,
- iş birliği yapma,
- teknolojiyi etkili bir şekilde kullanma ve toplumsal olaylardan haberdar olma

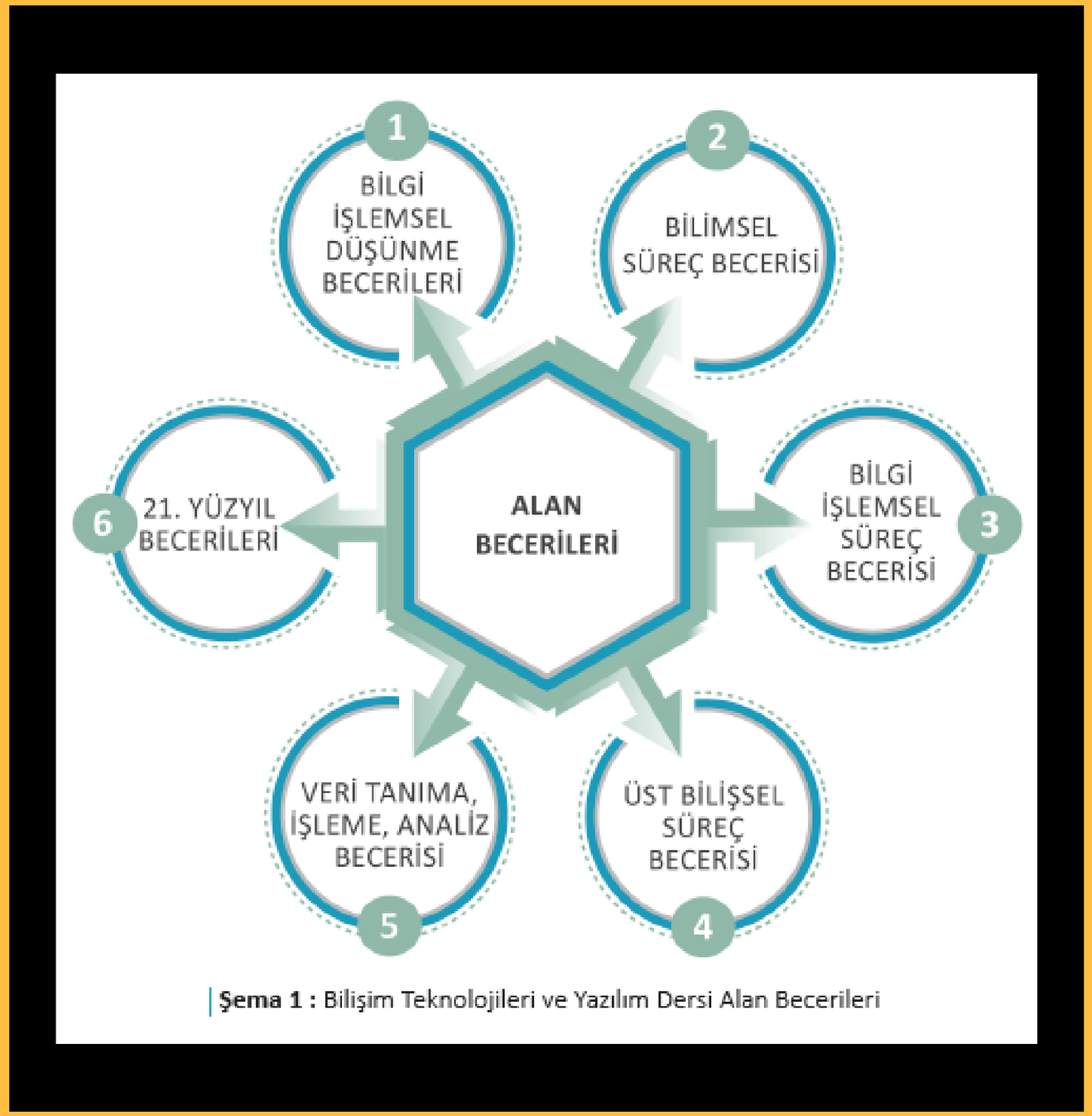
gibi çok yönlü yetenekleri kapsamaktadır.



Şema 1 : Bilişim Teknolojileri ve Yazılım Dersi Alan Becerileri

1. BİLGİ İŞLEMSEL DÜŞÜNME BECERİLERİ

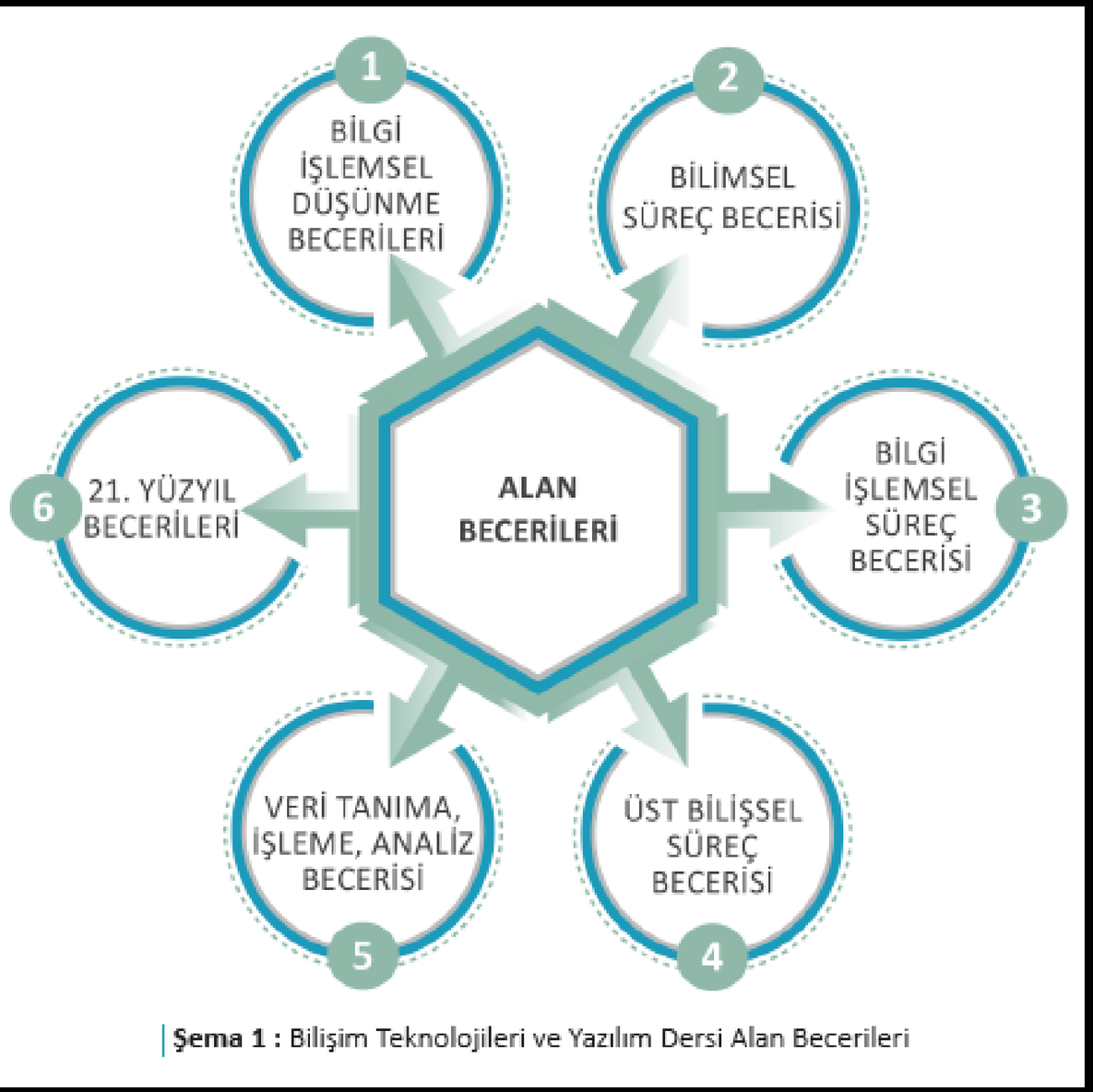
- Soyutlama Becerisi
- Mantıksal Düşünme Becerisi
- Test Etme ve Hata Ayıklama Becerisi
- Algoritmik Düşünme Becerisi
- Genelleme Becerisi



Şema 1 : Bilişim Teknolojileri ve Yazılım Dersi Alan Becerileri

2. BİLİMSEL SÜREÇ BECERİSİ

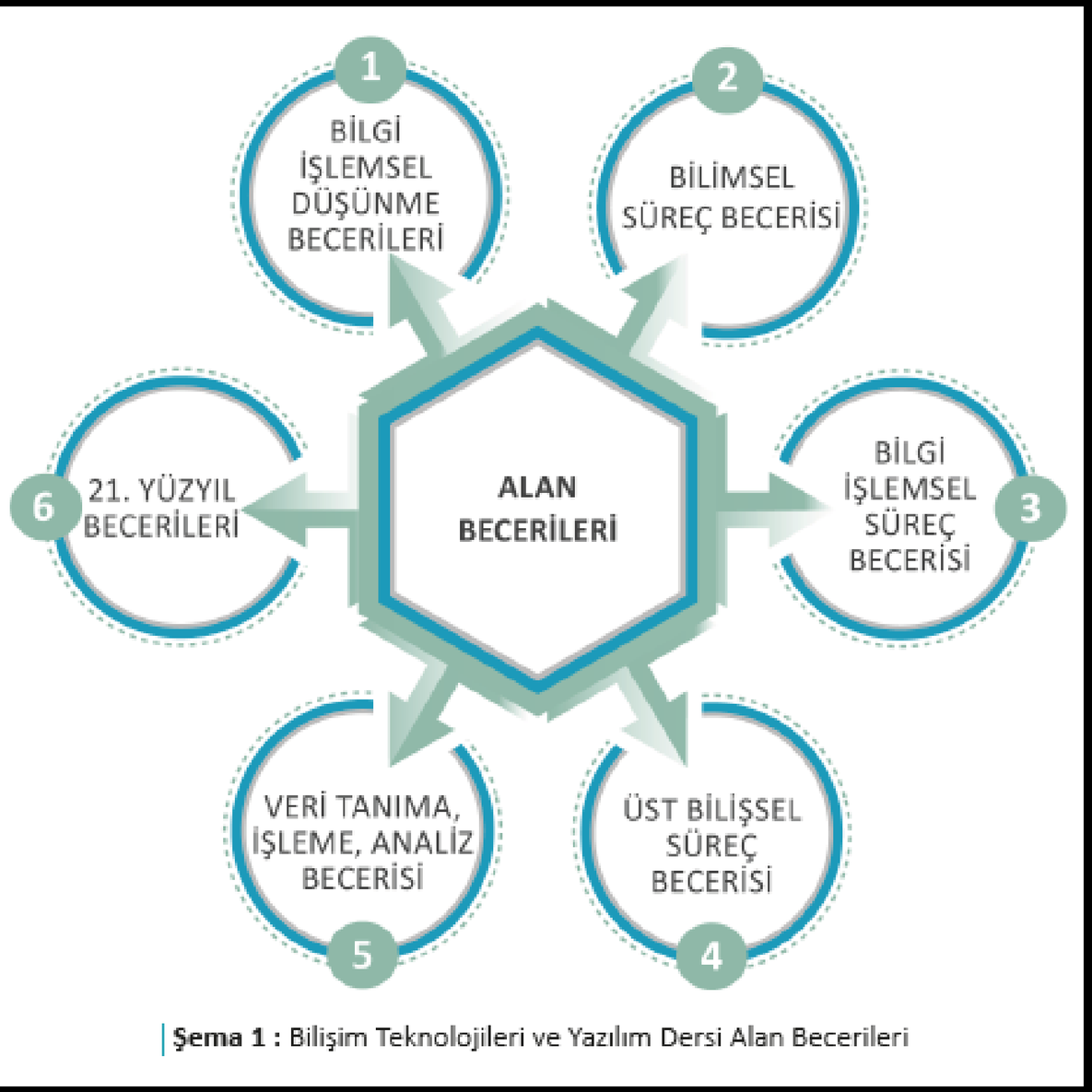
- Gözlem Yapma Becerisi
- Tahmin Etme Becerisi
- Ölçüm Yapma Becerisi



Şema 1 : Bilişim Teknolojileri ve Yazılım Dersi Alan Becerileri

3. BİLGİ İŞLEMSEL SÜREÇ BECERİSİ

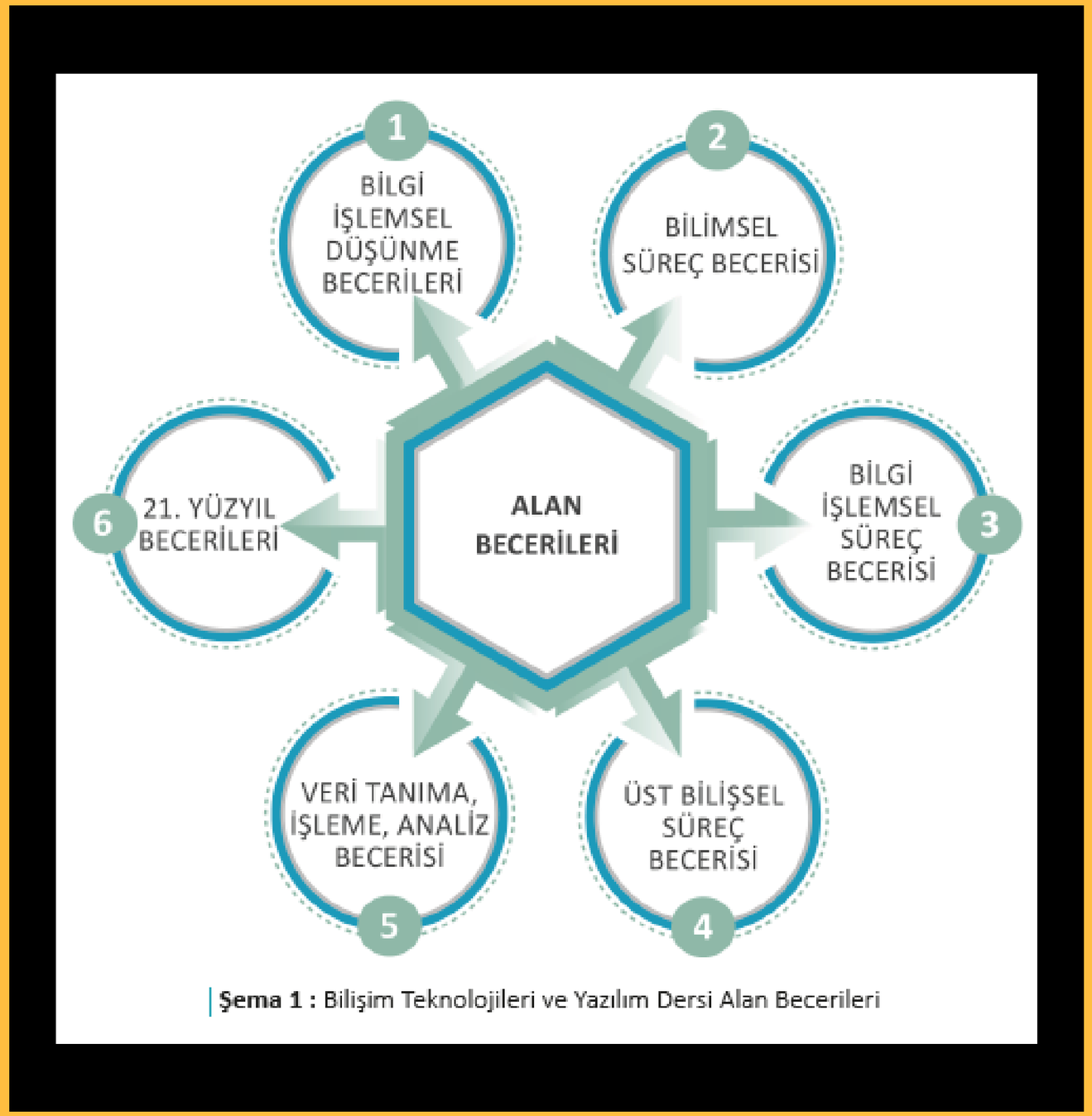
- Soyutlama Becerisi
- Algoritmik Düşünme Becerisi
- Kodlama, Programlama Becerisi
- Veri Toplama, İşleme, Dönüştürme Becerisi



Şema 1 : Bilişim Teknolojileri ve Yazılım Dersi Alan Becerileri

4. ÜST BİLİŞSEL SÜREÇ BECERİSİ

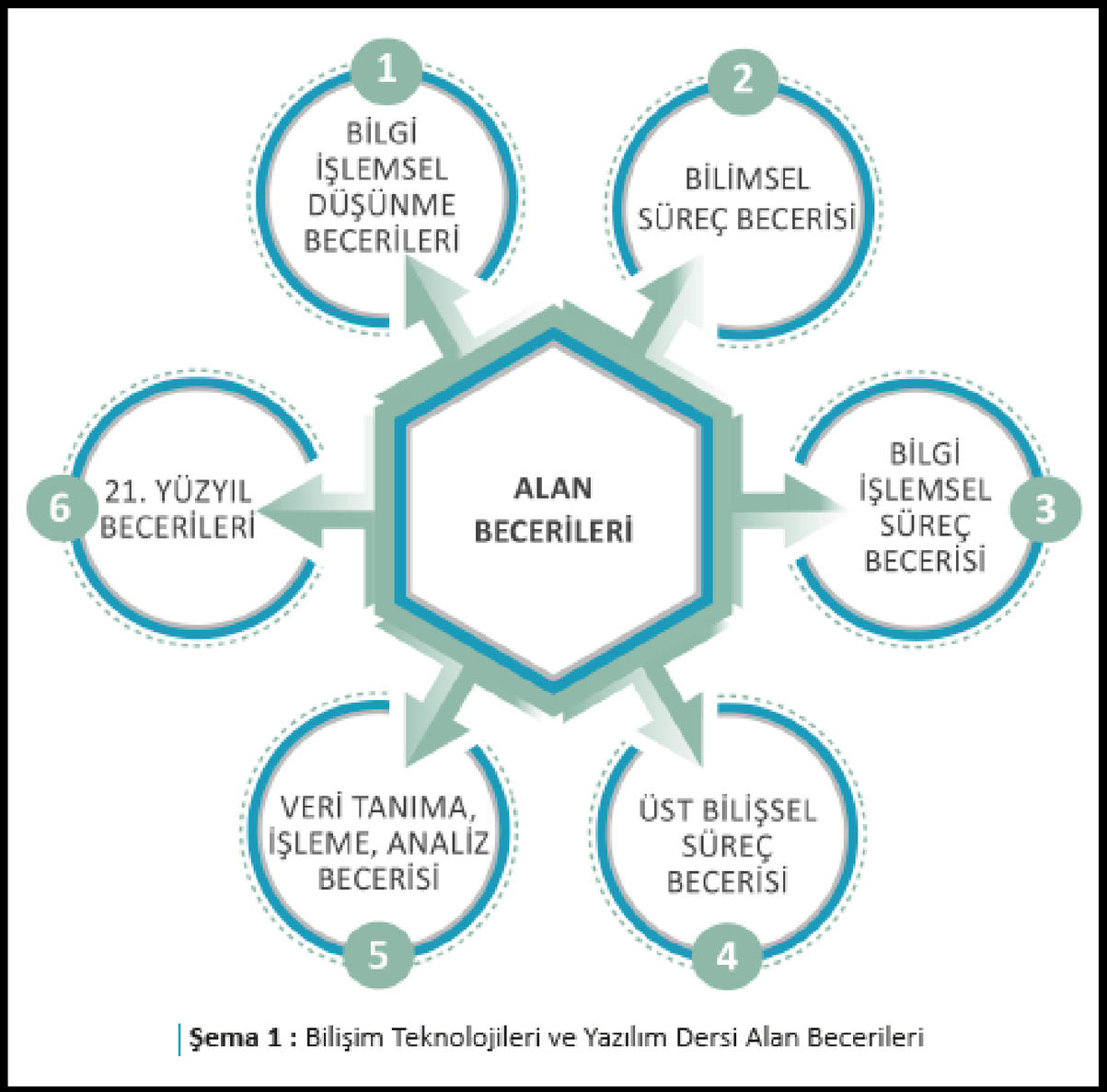
- Sistem Tasarlama Becerisi
- Sistem Kurma (İnşaa Etme) Becerisi
- Test Etme ve Hata Ayıklama Becerisi
- Teknik İşlem Becerisi



Şema 1 : Bilişim Teknolojileri ve Yazılım Dersi Alan Becerileri

5. VERİ TANIMA, İŞLEME, ANALİZ BECERİSİ

- Veri Toplama
- Veri Gösterimi ve Analizi
- Model Doğrulama
- Test ve Doğrulama
- Algoritma ve Süreçler
- Kontrol Yapıları

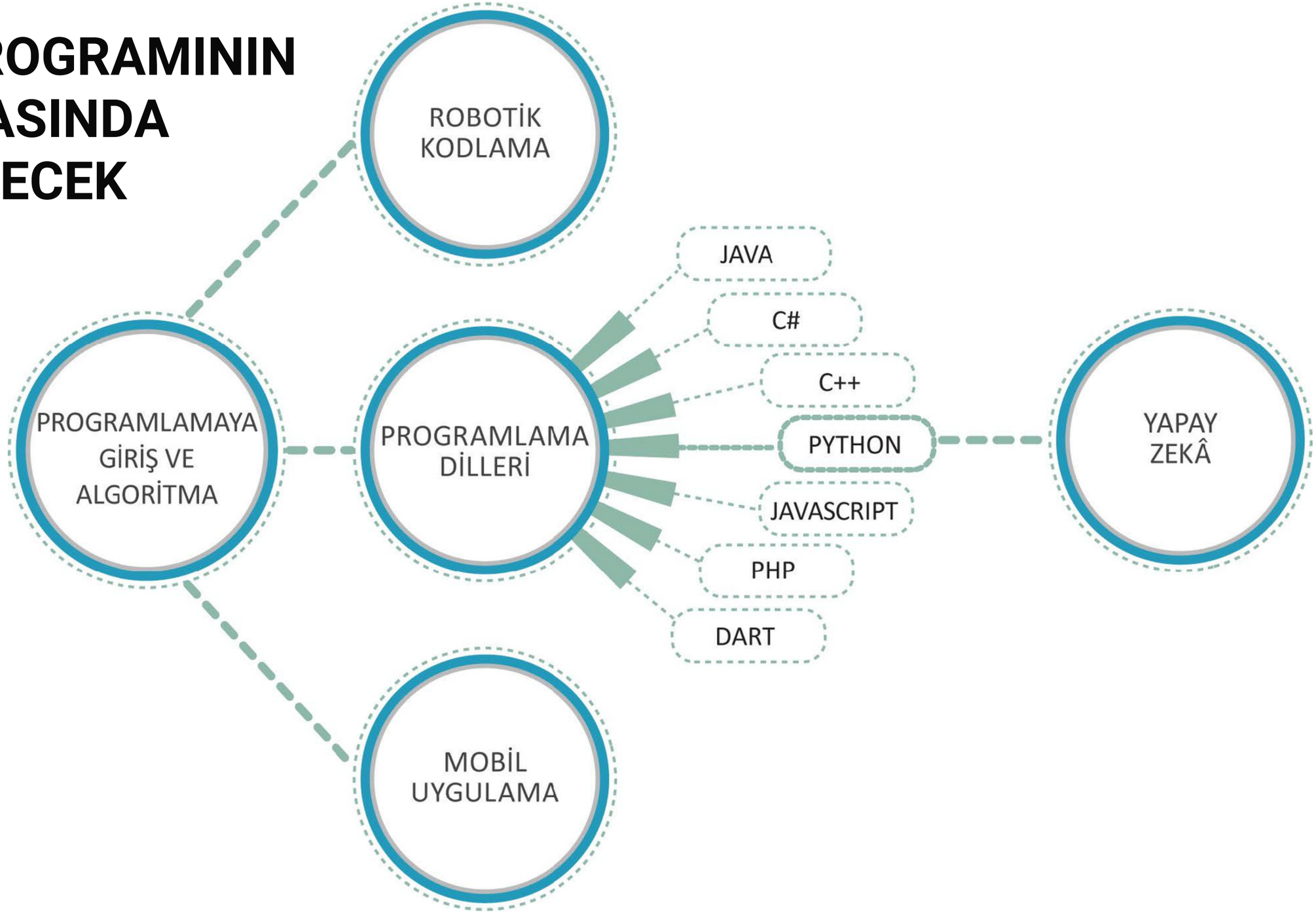


Şema 1 : Bilişim Teknolojileri ve Yazılım Dersi Alan Becerileri

6. 21. YÜZYIL BECERİLERİ

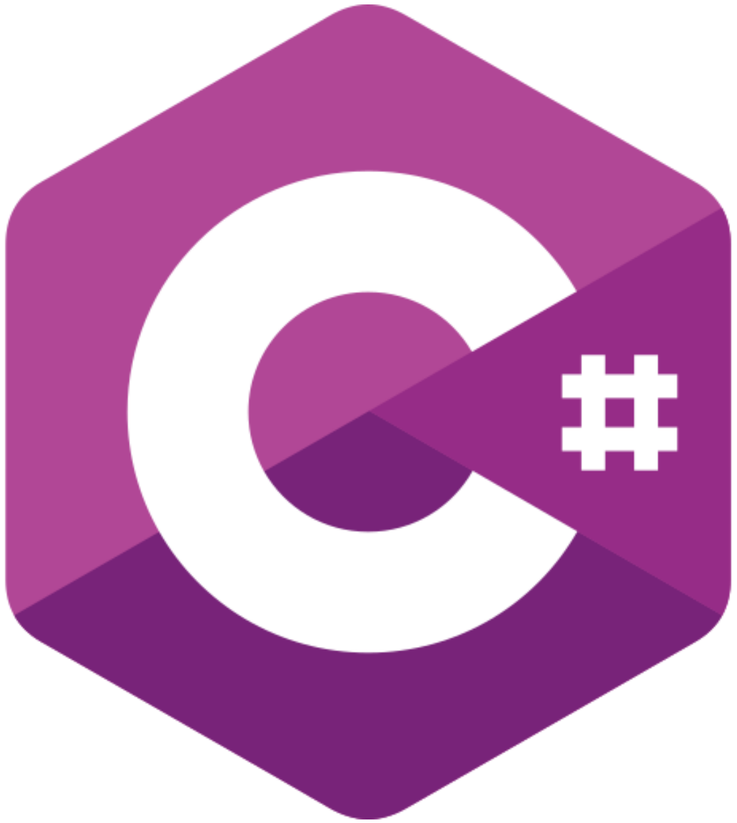
- 21. yüzyıl becerileri arasında bilgisayar programlama ve donanım konularının yanı sıra eleştirel düşünme, problem çözme, iş birliği, iletişim, yaratıcılık ve dijital yurttaşlık gibi yetenekler bulunmaktadır. Öğrenciler, karmaşık problemleri analiz etme ve çözme yeteneklerini kazanırken dijital araçlarla etkili bir şekilde iletişim kurmayı öğrenirler.

ÖĞRETİM PROGRAMININ UYGULANMASINDA DİKKAT EDİLECEK HUSUSLAR



PROGRAMLAMA DİLLERİ (C#)

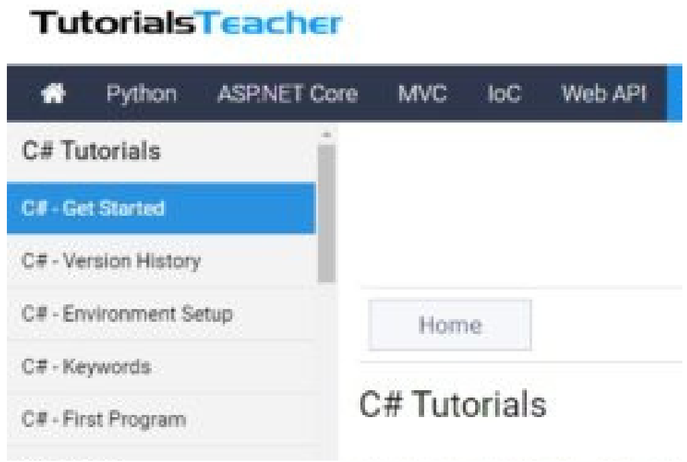
ÜNİTE		KAZANIM SAYISI	DERS SAATİ	YÜZDE ORANI (%)
1. Ünite:	C# Çalışma Ortamı	5	12	18
2. Ünite:	Karar ve Döngü Yapıları	3	12	16
3. Ünite:	Sınıflar	7	12	18
4. Ünite:	Diziler ve Koleksiyonlar	3	12	16
5. Ünite:	Formlar	4	12	16
6. Ünite:	Veri Tabanı İşlemleri	6	12	16



PROGRAMLAMA DİLLERİ (C#)

C# programlama dili modülünde temel çalışma ortamı, karar - döngü yapıları, sınıflar, diziler ve koleksiyonlar ile form oluşturma gibi konuları içeren becerilerin kazandırılması hedeflenmektedir. Ayrıca, veri tabanı işlemlerinin ayrıntılı biçimde ele alınması amaçlanmaktadır.

Ücretsiz Eğitim Platformları

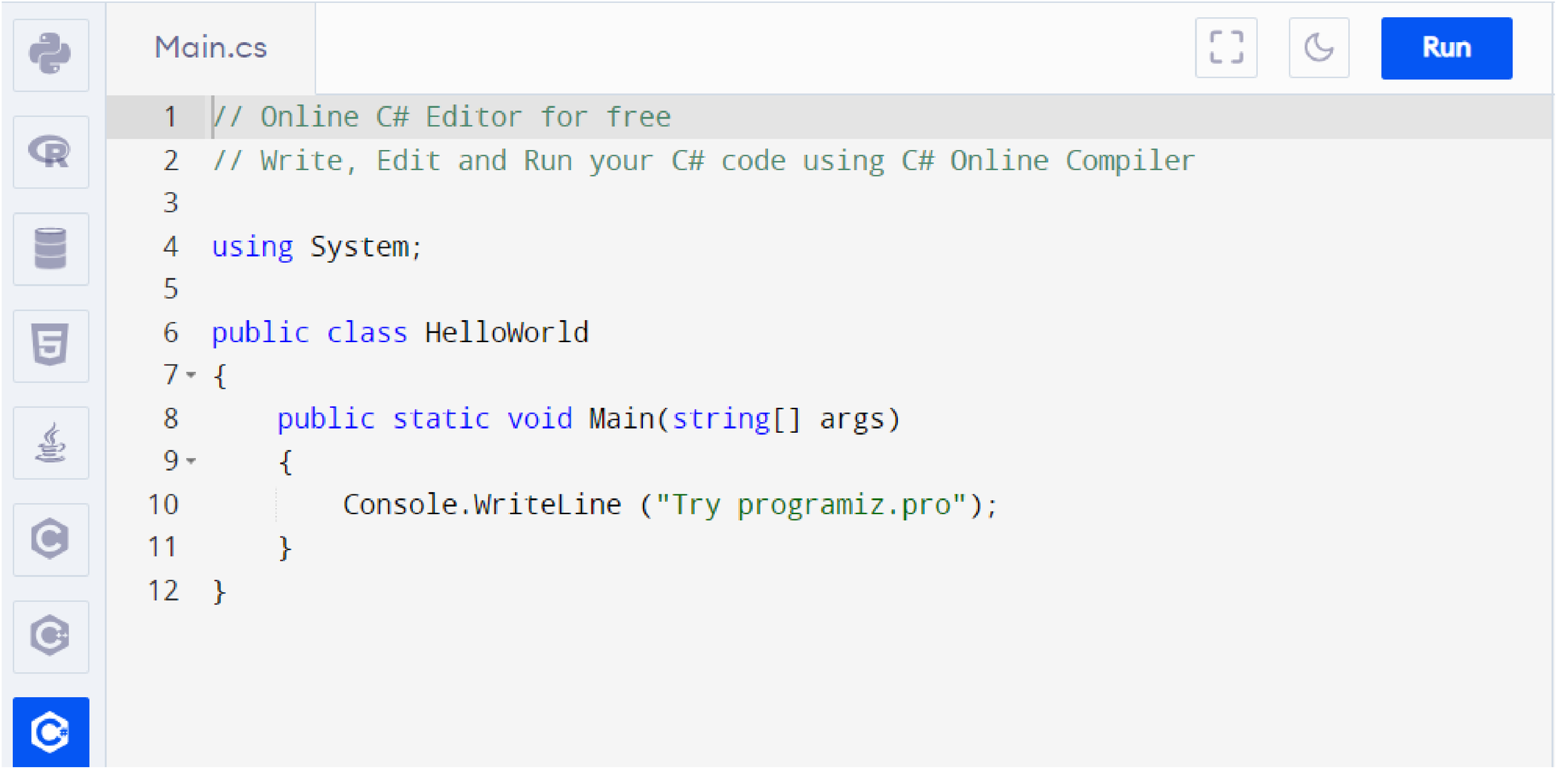


Online kod yazma özelliği ile bir program yüklemeden istediğiniz dilde kodları değiştirip pratik yapabilirsiniz. W3School web sitesinin en güzel özelliği sitede gördüğünüz kodları hemen deneme imkanı sunan kendin dene özelliğidir.

Tutorials Teacher, çeşitli programlama dillerinde ücretsiz eğitimler içeren popüler bir öğrenme merkezidir. Hem yeni başlayanlar hem de profesyoneller için eğitim sunar.

Programiz

C# Online Compiler



1. ÜNİTE: C# ÇALIŞMA ORTAMI

Öğretme yaklaşımları

C# programlama dili .NET çerçevesi ile entegre bir programlama dilidir. .NET çerçevesi, farklı dillerde yazılmış uygulamaların birlikte çalışabilmesini sağlamaktadır. C# programlama dili bu çerçevenin önemli bileşenlerinden biridir. C# programlama dilini kullanmak için .NET çerçevesinin yüklü olması gerekir. Farklı C# geliştirme ortamları kullanılarak konsol uygulamaları, Windows formları, web uygulamaları geliştirmek için C# kodu yazılabilir ve derlenebilir. Bu ünitede basit bir "Merhaba Dünya" uygulaması yazılarak C# diline giriş yapılabilir. C# programlama dilinde temel veri türleri int, float, double, bool, char ve string gibi yapılar değişken tanımlamak için kullanılabilir. Aritmetik, karşılaştırma ve mantıksal operatörler gibi ifadeler yazılabilir. Kod blokları süslü parantezler ile belirtilir.

1. ÜNİTE: C# ÇALIŞMA ORTAMI

KAZANIM 1.1. C# PROGRAMLAMA DİLİNİ TANIR.

KAZANIM 1.2. .NET FRAMEWORK İLE C# PROGRAMLAMA DİLİ İLİŞKİSİNİ TANIMLAR.

KAZANIM 1.3. KOD DÜZENLEYİCİ ARAYÜZÜNDE TEMEL İŞLEMLERİ GERÇEKLEŞTİRİR.

KAZANIM 1.4. NAMESPACE KAVRAMINI BİLİR.

KAZANIM 1.5. TEMEL VERİ TÜRLERİNİ VE DEĞİŞKEN İŞLEMLERİNİ AÇIKLAR.

KAZANIM 1.1. C# programlama dilini tanır.

- C# programlama dilinin doğuş süreci ve neden ihtiyaca hâsıl olduğu söylenir.
- Genel C# programlama dili kavramları ve söz dizimi anlatılır.
- Visual studio community (parasız) kurulumu anlatılabilir.

KAZANIM 1.2. .NET Framework ile C# programlama dili ilişkisini tanımlar.

- .NET Framework ve temel bileşenleri (CLR, .NET sınıf kütüphanesi, CTS, CLS, managed code, Visual Studio IDE vb.) açıklanır.
- .NET Framework ile C# programlama dilinin .NET Framework tarafından sağlanan CLR üzerinde nasıl etkileşimde bulunduğu örneklendirilir.
- .NET Framework'ün C# programlama dili ile kullanılarak web tabanlı sitelerden büyük ölçekli iş uygulamalarına kadar geniş bir yelpazede uygulamalar geliştirildiği vurgulanır.

.NET Framework C# programlama dili ilişkisi

Microsoft C# dilini .NET Framework ile kod geliştirmek için oluşturmuştur. C# kütüphaneleri .NET Framework kütüphaneleridir.

NET Framework çatısı farklı dillerin aynı ortamda çalışmasını sağlar. Geliştirilen projenin bir bölümü VB.NET ile diğer bölümü C# dili ile kodlanabilir.

Linkedin büyük ölçekli iş ağı olarak tasarlanırken C# kütüphanelerinden faydalanılmıştır.

DERLEME SÜRECİ

C# Kodu -> C# Derleyicisi -> MSIL(Microsoft Intermediate Language) Derleyicisi -> MSIL Kod (Taşınabilir Assembly kod)

ÇALIŞTIRMA SÜRECİ

MSIL Kod -> CLR(Common Language Runtime) -> Makine Dili

KAZANIM 1.3. Kod düzenleyici arayüzünde temel işlemleri gerçekleştirir.

- C# programlama dili arayüzünde kod yazma sürecinde hata ayıklama, otomatik tamamlama ve belge yönetimi araçları incelenir.
- Yeni bir C# programlama dili arayüzü projesi açma, proje dosyasını düzenleme, projeyi yapılandırma, dosya ekleme ve düzenleme işlemleri uygulatılır.
- Yazdığı kodları düzenli bir şekilde yönetmesi, değişiklikleri kaydetmesi ve ardından derleyerek çalışan bir uygulama yapması anlatılır.

KAZANIM 1.4. Namespace kavramını bilir.

- Namespace kavramı ve kullanım alanları vurgulanır.
- Farklı namespace türlerini içe aktararak kullanma öğretilir.
- Namespace türleri kod organizasyonunda uygulatılır.

Namespace Kullanım Alanları

Proje ve Çözüm Düzeyinde Organizasyon

- Projelerde, namespace'ler proje düzeyinde organizasyon sağlar. Bir proje içindeki sınıfları, modülleri ve diğer bileşenleri düzenlemek ve gruplamak için kullanılır.

Farklı Kütüphaneler ve Modüller Arasında Ayırma

- Farklı kütüphaneler veya modüller arasında çakışmaları önlemek için kullanılır. Bu şekilde, farklı bileşenlerin birbirleriyle çakışma olasılığı en aza indirilir.

İsim Çakışmalarını Önleme

- Aynı isme sahip iki farklı sınıf veya metot olasılığını önler. Örneğin, iki farklı kütüphanede aynı ismi taşıyan bir sınıf varsa, bu sınıfları ayırmak için namespace'ler kullanılır.

Framework ve Kütüphane Tasarımı

- Framework veya kütüphane tasarlarken, bu bileşenleri mantıklı gruplara ayırmak ve namespace'ler aracılığıyla kullanıcıya sunmak yaygındır. Bu, kullanıcıların bileşenleri daha iyi anlamasını ve kullanmasını sağlar.

```csharp
namespace Sirket.Proje.Modul
{
    public class Calisan
    {
        public string Ad { get; set; }
        public string Soyad { get; set; }
    }

    public class Departman
    {
        public string Ad { get; set; }
    }
}
```

Örnek bir C# namespace kullanımı

Yandaki örnekte, Sirket.Proje.Modul namespace'i altında Calisan ve Departman sınıfları bulunmaktadır.

Kodu düzenlemek, çakışmaları önlemek ve daha iyi organizasyon sağlamak için kullanılan tipik bir namespace yapısı kullanılmıştır.

KAZANIM 1.5. Temel veri türlerini ve değişken işlemlerini açıklar.

- Temel veri türleri (int, string, double vb.) ve farklı tipteki veriler (tam sayılar, metin, ondalık sayılar vb.) incelenir.
- Program arayüzünde değişkenler oluşturulur, değerlerle ilişkilendirilir ve bu değerler güncellenir.
- Matematiksel operatörler (+, -, *, / vb.) ile mantıksal operatörler (==, !=, <, > vb.) kullanılarak temel aritmetik ve karşılaştırma işlemleri gerçekleştirilir.
- Bir değişkenin değeri diğer bir değişkene atanarak veya aralarında matematiksel işlemler yapılarak veri aktarımı ve dönüşümü gerçekleştirilir.

Değerler

- Sabır (Kazanım 1.1., 1.2., 1.5.)
- Öz denetim (Kazanım 1.3., 1.4.)
- Saygı (Kazanım 1.1., 1.2., 1.3., 1.4., 1.5.)

Alan Becerileri

- Soyutlama Becerisi (Kazanım 1.1., 1.3.)
- Algoritmik Düşünme Becerisi (Kazanım 1.2., 1.5.)
- Veri Toplama, İşleme, Dönüştürme Becerisi (Kazanım 1.2., 1.4.)

2. ÜNİTE: KARAR VE DÖNGÜ YAPILARI

Öğretme yaklaşımları

Belirli bir talimat bloğunu yalnızca belirli bir koşul doğru olduğunda yürütmek için kontrol ifadeleri; bir bloğu tekrar tekrar çalıştırmak için ise döngüler kullanışlıdır. Karar ifadelerinin programın belirli koşullara göre farklı davranmasını sağladığını ve programları belirli şartlara göre yönlendirmeyi öğrenir. Karar ifadelerinde karşılaştırmayı mantıksal operatörlerle (eşitlik, büyüklük-küçüklük kontrolü vb.) tanımlamayı ve karşılaştırmayı örnekler üzerinde uygular. Kullanıcıdan alınan girdilere veya belirli şartlara bağlı olarak karar ifadelerini kullanarak programlama yapmayı öğrenir. Döngü yapıları ile belirli işlemlerin tekrarlanmasını sağlamayı ve program akışını kontrol etmeyi öğrenir. Farklı türdeki döngü yapılarının (for, while, do-while) çalışma prensiplerini anlayarak veri koleksiyonları üzerinde dolaşma, belirli bir aşamaya kadar işlem yapma ve tekrar eden görevleri otomatikleştirme gibi işlemleri gerçekleştirir.

2. ÜNİTE: KARAR VE DÖNGÜ YAPILARI

KAZANIM 2.1. KARAR İFADELERİNİ ÖRNEKLERLE AÇIKLAR.

KAZANIM 2.2. MANTIKSAL OPERATÖRLERİ KULLANARAK KOŞULLAR OLUŞTURUR.

KAZANIM 2.3. DÖNGÜ YAPILARINI KULLANARAK TEKRARLAYAN İŞLEMLER GERÇEKLEŞTİRİR.

KAZANIM 2.1. Karar ifadelerini örneklerle açıklar.

- Programın belirli koşullara göre farklı davranışlar sergilemesini sağlayan karar ifadelerini farklı durumlar altında kontrol edip programlarında belirli şartlara göre yönlendirmesi anlatılır.
- Karar ifadeleriyle ilgili karşılaştırmanın mantıksal operatörlerle (eşitlik, büyüklük-küçüklük kontrolü vb.) tanımlanması ve karşılaştırılması örneklendirilir.
- Programlamanın temel yapı taşı olan karar ifadelerini, kullanıcıdan alınan girdilere veya belirli şartlara bağlı olarak kullanacağı örnek uygulamalar yaptırılır.

KAZANIM 2.2. Mantıksal operatörleri kullanarak koşullar oluşturur.

- Mantıksal operatörler (AND, OR, NOT) anlatılır.
- İki veya daha fazla koşulun aynı anda doğru olması gerektiği durumlarda mantıksal operatörleri (AND, OR, NOT) birleştirerek kullanması sağlanır.
- Mantıksal operatörler karmaşık koşulların analizi için farklı örneklerde çalışılır.

KAZANIM 2.3. Döngü yapılarını kullanarak tekrarlayan işlemler gerçekleştirir.

- Döngü yapıları ile belirli işlemlerin tekrarlanması öğretilir.
- Farklı türdeki döngü yapılarının (for, while, do-while) çalışma prensibi açıklanır.
- Veri koleksiyonları üzerinde dolaşma, belirli bir aşamaya kadar işlem yapma, işlemi devam ettirme ve tekrar eden görevleri otomatikleştirme işlemlerinde döngüleri kullanma durumları örneklendirilir.

Değerler

- Sabır (Kazanım 2.1.)
- Öz denetim (Kazanım 2.2., 2.3.)
- Yardımseverlik (Kazanım 2.3.)

Alan Becerileri

- Soyutlama Becerisi (Kazanım 2.1., 2.3.)
- Algoritmik Düşünme Becerisi (Kazanım 2.2.)
- Veri Toplama, İşleme, Dönüştürme Becerisi (Kazanım 2.2., 2.3.)

3. ÜNİTE: SINIFLAR (CLASS)

Öğretme yaklaşımları

C# programlama dilinde nesne tabanlı programlamayı nasıl kullanabileceğine yönelik uygulamalar yapılır. Nesne yönelimli programlamanın temel kavramları üzerinde durulur. Bir sınıf oluşturarak oluşturduğu sınıftan nesneler üretmesi sağlanır. Kapsülleme, alanlar ve özellikler, bir sınıf içinde verilerin nasıl saklandığını kontrol etmemizi, dışarıdan nasıl erişileceğini belirlememizi ve güvenliği artırmamızı sağlayan önemli kavramlardır. Gerçek dünya örnekleri ve öğrencilere bu kavramları uygulamalarını sağlayacak basit projeler, konuyu daha iyi anlamalarına yardımcı olabilir.

3. ÜNİTE: SINIFLAR (CLASS)

KAZANIM 3.1. NESNE TABANLI PROGRAMLAMANIN C# PROGRAMLAMA DİLİ İÇERİSİNDE NASIL UYGULANDIĞINI AÇIKLAR.

KAZANIM 3.2. KAPSÜLLEME, ALANLAR VE ÖZELLİKLER KAVRAMLARINI AÇIKLAR.

KAZANIM 3.3. METOTLARI KULLANIR.

KAZANIM 3.4. YAPICI VE YIKICI METOTLARIN ROLÜNÜ AÇIKLAR.

KAZANIM 3.5. DEĞER VE REFERANS TİPLERİ ARASINDAKİ FARKI KAVRAR.

KAZANIM 3.6. KALITIM KAVRAMINI KAVRAR.

KAZANIM 3.7. NESNEYE YÖNELİK C# PROGRAMLAMA DİLİ TEMEL YAPILARINI KULLANIR.

League of Legends: C++, C#

League of Legends (LoL), Riot Games tarafından geliştirilen ve dünya çapında büyük bir oyuncu kitlesi bulunan bir MOBA (Çok Oyunculu Çevrimiçi Savaş Arenası) oyunudur. Oyunun çekirdek bileşenleri ve performans kritik bölümleri C++ diliyle yazılmıştır. Oyunun kullanıcı arayüzü ve istemci tarafı ise C# programlama dili kullanılarak geliştirilmiştir.

KAZANIM 3.1. Nesne tabanlı programlamanın C# programlama dili içerisinde nasıl uygulandığını açıklar.

- C# programlama dili içinde sınıflar, nesneler, kalıtım ve çok biçimlilik olarak ifade edilen nesne tabanlı yapıların gerçek dünya entegrasyonu sağlanarak kodun daha organize ve sürdürülebilir hâle gelmesini sağlayan avantajlar öğretilir.
- Nesne tabanlı programlama yapılarını kullanarak gerçek dünya sorunları incelenir.
- Nesne tabanlı programlama yapılarının prensipleri ve uygulamaların arka planındaki mantık açıklanarak örnek uygulamalar gerçekleştirilir.

Banka Hesap Yönetimi

GERÇEK DÜNYA PROBLEMİ: BANKA MÜŞTERİLERİNİN HESAPLARININ YÖNETİLMESİ.

Nesne Tabanlı Yapı:

- Sınıflar: Müşteri, Hesap, İşlem
- Özellikler: Müşteri adı, hesap numarası; Hesap bakiyesi, hesap türü; İşlem tarihi, tutarı
- Metodlar: Para yatırma, çekme; Hesap hareketlerini görüntüleme

E-ticaret Uygulaması

GERÇEK DÜNYA PROBLEMİ: ONLİNE ALIŞVERİŞ SİTESİNİN ÜRÜN YÖNETİMİ VE MÜŞTERİ İŞLEMLERİ.

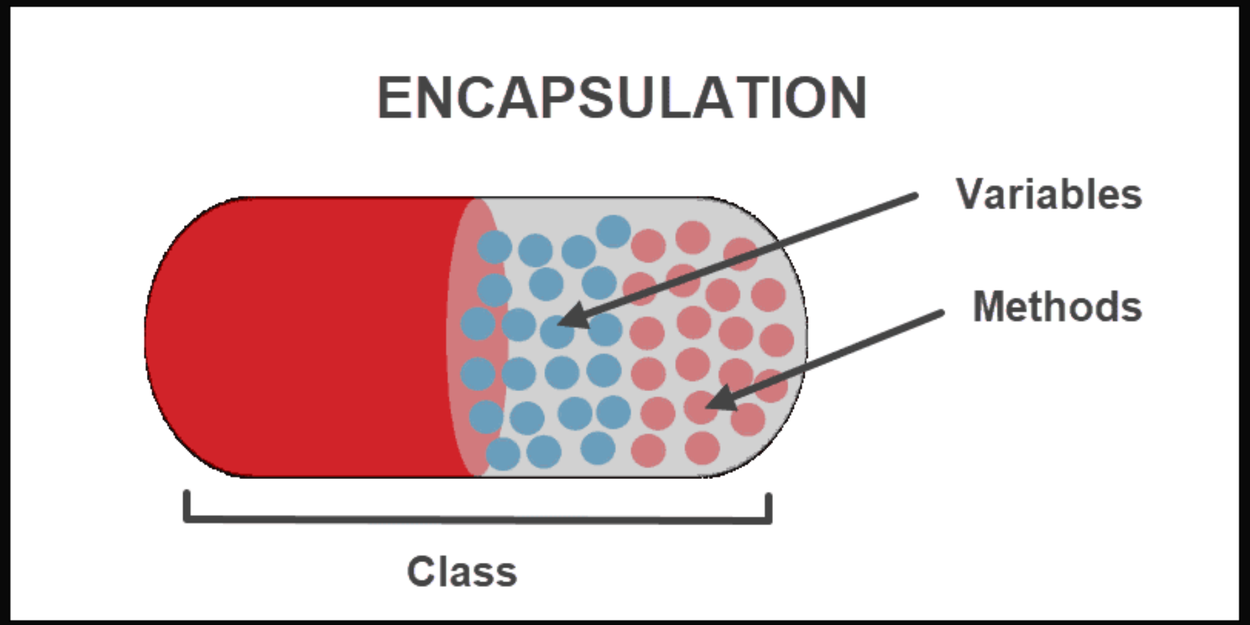
Nesne Tabanlı Yapı:

- Sınıflar: Ürün, Müşteri, Sipariş
- Özellikler: Ürün adı, fiyatı, stok durumu; Müşteri adı, adresi, sipariş geçmişi; Sipariş numarası, tarih
- Metodlar: Ürün ekleyip kaldırma, müşteri kaydı, sipariş oluşturma

KAZANIM 3.2. Kapsülleme, alanlar ve özellikler kavramlarını açıklar.

- Kapsülleme kavramı ve neden önemli olduğu anlatılır.
- Alanların özellikleri vurgulanır.
- Kapsüllemenin sınıf tasarımında kullanımı ve güvenli erişim yöntemleri açıklanır.

Neden Kapsülleme?



- GÜVENLİK VE VERİ BÜTÜNLÜĞÜ
- MANTIKSAL DÜZEN VE BAKIM KOLAYLIĞI
- YENİDEN KULLANILABİLİRLİK
- ESNEKLİK VE DEĞİŞTİRİLEBİLİRLİK
- SİSTEM TASARIMINDA MODÜLERLİK

KAPSÜLLEME ARABAYI BİR NESNE OLARAK ELE ALARAK AÇIKLANABİLİR.

Motor Kapsülleme:

- Arabanın motoru, karmaşık bir yapıya sahip ve birçok parçadan oluşan bir sistemdir. Ancak, sürücü veya diğer sistemler motorun iç detaylarına tam olarak erişemez. Araba sahibi sadece aracın çalıştığından emin olabilir ve motorun iç detaylarına gerek duymaz.

Kaporta ve Dış Görünüm:

- Arabanın dış görünümü, yani kaporta, aracın iç detaylarını gizleyen bir kapsülleme örneğidir. Bu, aracın estetik ve aerodinamik özelliklerini korur ve dışarıdan bakıldığında aracın içindeki karmaşık sistemlerin detaylarına gerek duyulmaz.

Servis Arayüzü:

- Araba servisi, aracın bakımını yapmak ve onarımları gerçekleştirmek için bir arayüzdür. Ancak, servis arayüzü sadece belirli bir metot ve prosedürleri sunar. Servis personeli, motorun iç detaylarına tam erişim sağlar, ancak bu detaylar dışarıya açık değildir.

KAZANIM 3.3. Metotları kullanır.

- Belirli bir işlevi yerine getiren kod bloklarını tanımlamak için metotlar ve kod blokları anlatılır.
- Metotların parametre alması ve bu parametreleri kullanması vurgulanır.
- Metotların geri dönüş değerlerini ve sonuçları nasıl ilettiği incelenir.
- Instance metotlar, statik metotlar ve sanal metotlar arasındaki farklar açıklanır.

Üçgen Alanı Hesaplama

csharp

```csharp
using System;

class Program
{
    static void Main()
    {
        double tabanUzunlugu = 6.0;
        double yukseklik = 4.5;

        double alan = UcgenAlaniHesapla(tabanUzunlugu, yukseklik);

        Console.WriteLine($"Üçgenin alanı: {alan}");
    }

    static double UcgenAlaniHesapla(double taban, double yukseklik)
    {
        return 0.5 * taban * yukseklik;
    }
}
```

Yandaki örnek, üçgenin taban uzunluğu ve yüksekliği parametre olarak alan ve üçgenin alanını hesaplayan metodu içerir.

KAZANIM 3.4. Yapıcı ve yıkıcı metotların rolünü açıklar.

- Yapıcı ve yıkıcı metotların sınıfların süresini nasıl etkilediği incelenir.
- Yapıcı metotların parametreleri ve farklı aşırı yüklenmiş yapıcı metotların nasıl tanımlandığı anlatılır.
- Yıkıcı metotlarda kaynağın serbest bırakılması ve sınıfın sonlandırılması süreçlerindeki rolü açıklanır.

csharp

```csharp
public class Araba
{
    public string Marka { get; set; }
    public string Model { get; set; }

    // Parametre alan yapıcı metot
    public Araba(string marka, string model)
    {
        Marka = marka;
        Model = model;
    }
}
```

csharp

```csharp
public class DosyaIslemleri
{
    // Yıkıcı metot
    ~DosyaIslemleri()
    {
        // Nesne imha edilmeden önce yapılacak işlemler
    }
}
```

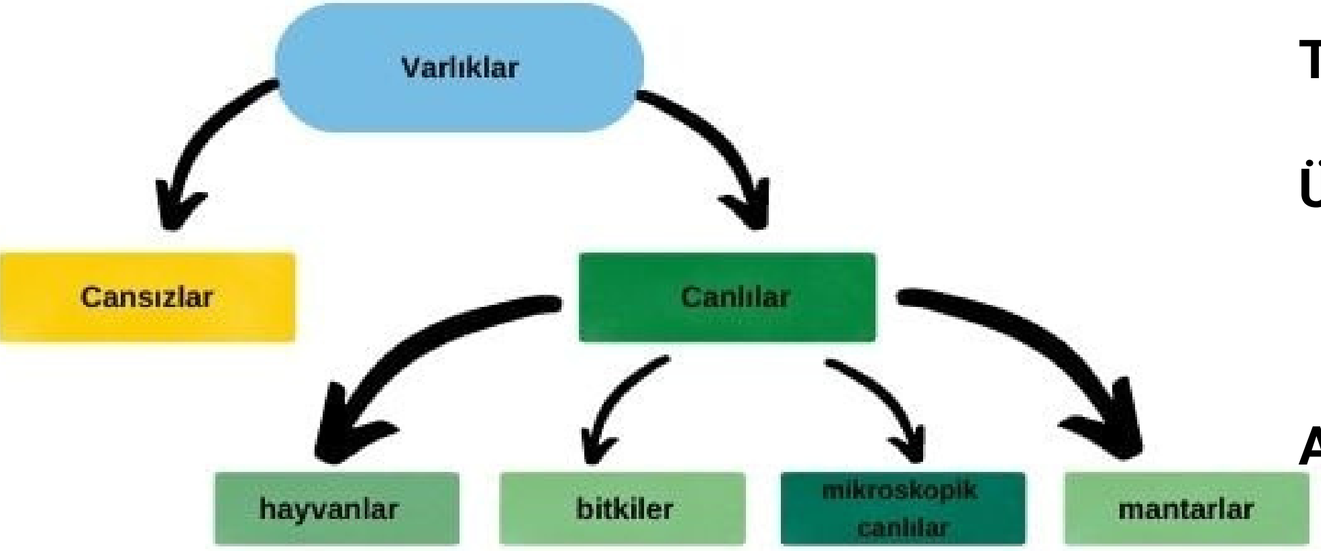
Yapıcı ve yıkıcı metotlar, bir sınıfın başlatılması ve sonlandırılması süreçlerini kontrol etmek için kullanılır. Bu metotlar, nesne tabanlı programlamada önemli bir rol oynar ve kodun düzenli ve etkili bir şekilde çalışmasını sağlar.

KAZANIM 3.5. Değer ve referans tipleri arasındaki farkı kavrar.

- Değer tipleri (int, float, bool) ve referans tipleri (sınıflar, diziler) arasındaki temel farklar anlatılır.
- Değer tiplerinin değer kopyalama, referans tiplerinin referans kopyalama üzerinde çalışması açıklanır.
- Değer-referans tiplerinin farklı kullanım senaryoları ve avantajları vurgulanır.

KAZANIM 3.6. Kalıtım kavramını kavrar.

- Bir sınıfın diğer bir sınıftan kalıtım alması anlatılır.
- Alt sınıfların üst sınıflardan türetildiği ve temel sınıf ile türetilmiş sınıf arasındaki ilişki açıklanır.
- Kalıtım kavramının sınıf tasarımında kullanımı ve kodun yeniden kullanılabilirliği uygulamalarla incelenir.



Temel kavramlar

Üst Sınıf (Base Class veya Parent Class)

Kalıtım verisi sağlayan sınıfa "üst sınıf" veya "temel sınıf" denir. Bu sınıf, ortak özellikleri ve davranışları içerir.

Alt Sınıf (Derived Class veya Child Class)

Kalıtım alacak sınıfa "alt sınıf" veya "türetilmiş sınıf" denir. Bu sınıf, üst sınıfın özelliklerini miras alır ve bu özelliklere ek özellikler ekleyebilir veya mevcut özellikleri değiştirebilir.

Miras Alınan Özellikler (Inherited Members)

Üst sınıftan alt sınıfa aktarılan özellikler, alanlar ve metotlar gibi unsurlardır.

Miras Alan (Derived)

Alt sınıfı ifade eder. Bir sınıfın, diğer bir sınıfın özelliklerini miras aldığı durumu ifade eder.

```csharp
// Üst Sınıf
public class Arac
{
    public string Marka { get; set; }
    public string Model { get; set; }

    public void Sur()
    {
        Console.WriteLine("Araç sürülüyor.");
    }
}

// Alt Sınıf
public class Araba : Arac
{
    public int MotorGucu { get; set; }

    public void ArabaMetodu()
    {
        Console.WriteLine("Araba özel metodu.");
    }
}
```

Yandaki örnekte, Araba sınıfı, Arac sınıfından özelliklerini (Marka, Model, Sur metodu) miras alıyor. Bu sayede Araba sınıfı, kendi özel özelliklerini (MotorGucu, ArabaMetodu) ekleyebilir. Kalıtımı sınıf içerisinde kullanmanın temel avantajları aşağıdaki gibidir.

Yeniden Kullanılabilirlik

- Bir sınıfın özelliklerini miras alarak, kodun tekrar kullanılabilirliği artar. Ortak özelliklere sahip sınıfların tekrar tekrar yazılması gerekmez.

Organizasyon ve Hiyerarşi

- Kalıtım, sınıfları bir hiyerarşi içinde düzenler. Genel özelliklere sahip bir üst sınıf ve bu sınıftan türetilen özel sınıflar arasında bir düzen oluşturur.

Değişiklikleri Yönetme

- Bir üst sınıfta yapılan değişiklikler, alt sınıfları da etkiler. Bu sayede, kodun bakımı ve değişikliklere uyum sağlama kolaylaşır.

Polimorfizm

- Kalıtım, polimorfizm konseptini destekler. Bu, bir nesnenin farklı biçimlere sahip olabilmesi anlamına gelir. Aynı türden farklı sınıfların aynı arayüzü paylaşabilmesini sağlar.

KAZANIM 3.7. Nesneye yönelik C# programlama dili temel yapılarını kullanır.

- Soyut sınıflar ve arayüzlerin kullanımı açıklanır.
- Çok biçimlilik ile farklı nesnelerin aynı metodları farklı şekillerde uygulayabilmesi incelenir.
- Statik sınıflar, isimsiz sınıflar, mühürlü sınıflar, parçalı sınıflar ve numaralandırmalar C# programlama arayüzünde çalıştırılır.

Değerler

- Sabır (Kazanım 3.1., 3.6., 3.7.)
- Öz denetim (Kazanım 3.2., 3.3.)
- Saygı (Kazanım 3.1., 3.3., 3.4., 3.5., 3.6., 3.7.)

Alan Becerileri

- Soyutlama Becerisi (Kazanım 3.1., 3.3., 3.4., 3.5., 3.6., 3.7.)
- Mantıksal Düşünme Becerisi (Kazanım 3.2.)
- Tahmin Etme Becerisi (Kazanım 3.3.)
- Algoritmik Düşünme Becerisi (Kazanım 3.2., 3.3., 3.7.)
- Kodlama, Programlama Becerisi (Kazanım 3.3.)
- Genelleme Becerisi (Kazanım 3.3., 3.7.)

4. ÜNİTE: DİZİLER VE KOLEKSİYONLAR

Öğretme yaklaşımları

Diziler aynı türdeki verilerin bir araya getirilmesi ve bir indeksleme sistemi kullanarak erişim sağlanmasıdır. C# programlama dilinde tek boyutlu ve çok boyutlu diziler bulunmaktadır. C# koleksiyonlarında değer türlerini (value types) ve referans türlerini (reference types) birbirine dönüştürmek için boxing ve unboxing kavramlarının kullanımı açıklanır. C# dilinde yaygın olarak kullanılan bazı koleksiyon türleri olan List, Queue, Stack, Dictionary, Hashtable ve SortedList kullanımları örneklendirilir.

4. ÜNİTE: DİZİLER VE KOLEKSİYONLAR

KAZANIM 4.1. TEK BOYUTLU VE ÇOK BOYUTLU DİZİLERİ KAVRAR.

KAZANIM 4.2. KOLEKSİYONLARDA BOXİNG VE UNBOXİNG KAVRAMLARINI AÇIKLAR.

KAZANIM 4.3. KOLEKSİYON TÜRLERİNİ KAVRAR.

KAZANIM 4.1. Tek boyutlu ve çok boyutlu dizileri kavrar.

- Tek boyutlu ve çok boyutlu dizilere değişkenleri doğru bir şekilde tanımlayarak değer atama uygulamaları C# programlama arayüzünde kullanılır.
- Verileri sıralama, arama ve değiştirme işlemleri için indeks numarası kullanarak dizi elemanlarına erişim uygulamaları gerçekleştirilir.
- Dizilerde sayıların, metinlerin ve diğer veri türlerinin kullanımı açıklanır.

Tek Boyutlu Diziler

```csharp
// Tek boyutlu tam sayı dizisi tanımı
int[] tekBoyutluDizi = new int[5]; // 5 elemanlı bir dizi
```

- Yukarıdaki örnekte, int türünde elemanlar içeren ve 5 elemana sahip bir tek boyutlu dizi tanımlanmıştır. Dizinin elemanları sıfırdan başlayarak indekslenir. Örneğin, tekBoyutluDizi[0] ilk elemanı temsil eder.

Çok Boyutlu Diziler

- Çok boyutlu diziler, iki veya daha fazla boyutta veri depolayan yapıları ifade eder. C# dilinde en yaygın olarak kullanılan iki boyutlu dizilerdir, ancak teorik olarak daha fazla boyutta da oluşturulabilirler.

```csharp
// İki boyutlu tam sayı dizisi tanımı
int[,] ikiBoyutluDizi = new int[3, 4]; // 3×4 matris
```

- Yukarıdaki örnekte 3 satır, 4 sütundan oluşan bir iki boyutlu dizi tanımlanmıştır. Elemanlara erişim için iki indeks kullanılır, örneğin ikiBoyutluDizi[0, 0] ilk elemanı temsil eder.

KAZANIM 4.2. Koleksiyonlarda boxing ve unboxing kavramlarını açıklar.

- Değer tiplerinin (int, char, vb.) ve referans tiplerinin (object, interface, vb.) değişimi anlamına gelen boxing ve unboxing kavramları tanıtılır.
- Farklı tipteki ögeleri dinamik bir şekilde saklayabilen ArrayList koleksiyonunun oluşturulması ve dinamik verilerin depolanması anlatılır.
- Farklı tipteki verileri ArrayList koleksiyonu ile saklama açıklanır.
- Koleksiyon elemanlarına erişme örnekleri çalışılır.
- Koleksiyonların verileri düzenlemek, filtrelemek ve hızlı erişim sağlamak için kullanılması anlatılır.
- Dinamik boyutları, veri yönetimi kolaylığı ve performans optimizasyonu özelliklerine sahip koleksiyonların avantajları vurgulanır.

```csharp
List<object> objectList = new List<object>();

int sayi1 = 10;
double sayi2 = 3.14;

// Boxing işlemi
objectList.Add(sayi1);
objectList.Add(sayi2);

// Unboxing işlemi
int geriAlinanSayi = (int)objectList[0];
double geriAlinanSayi2 = (double)objectList[1];
```

Boxing ve Unboxing Kullanımı

Bu örnekte, List<object> türünde bir koleksiyon oluşturulmuş ve bu koleksiyona değer tipleri boxing işlemi ile eklenmiştir. Daha sonra, koleksiyondan elemanlar geri alınarak unboxing işlemi gerçekleştirilmiştir.

```csharp
using System;
using System.Collections;

class Program
{
    static void Main()
    {
        // ArrayList oluşturuluyor
        ArrayList myArrayList = new ArrayList();

        // Farklı tiplerde veriler ekleniyor
        myArrayList.Add(42);       // int
        myArrayList.Add("Merhaba"); // string
        myArrayList.Add(3.14);     // double
        myArrayList.Add(true);     // bool

        // Verilere erişim ve kullanım
        Console.WriteLine("ArrayList Elemanları:");

        foreach (object item in myArrayList)
        {
            Console.WriteLine(item);
        }

        // Örnek bir unboxing işlemi
        int intValue = (int)myArrayList[0];
        Console.WriteLine("Unboxing: " + intValue);
    }
}
```

Farklı tipteki verileri ArrayList koleksiyonu ile saklama

Bu örnekte, ArrayList koleksiyonu içine int, string, double ve bool türlerinde farklı veri tipleri eklenmiştir.

foreach döngüsü kullanılarak koleksiyon içindeki verilere erişilmiştir.

KAZANIM 4.3. Koleksiyon türlerini kavrar.

- List, queue-stack, dictionary, hashtable ve sortedlist koleksiyon türleri anlatılır.
- Her bir koleksiyon türünün kullanımı ve tercih edilme durumları incelenir.
- Veri saklama ve işleme gibi uygulama örnekleriyle koleksiyonların işlevselliği fark ettirilir.

Değerler

- Öz denetim (Kazanım 4.2)

Alan Becerileri

- Sorgulama Becerisi (Kazanım 4.2, 4.3)
- Mantıksal Düşünme Becerisi (Kazanım 4.1)

5. ÜNİTE: FORMLAR

Öğretme yaklaşımları

WinForms teknolojisi, karmaşık uygulamalarda kullanabilen temel arayüz bileşenlerini sunar. Konteyner sınıfları, içindeki bileşenleri düzenlemek, gruplamak ve organize etmek için çok kullanışlıdır. Kullanıcı ile etkileşimi sağlarken menüler ve sağ tıklama menüleri örneklendirilir. MessageBox, OpenFileDialog, SaveFileDialog iletişim kutularını tasarlanır. Kullanıcı girişi doğrulama işlemleri (validasyon), kullanıcı tarafından sağlanan verilerin doğruluğunu kontrol etmek ve uygulamanın güvenilirliğini artırmak için önemlidir. C# Windows Forms uygulamasında kullanıcı girişi doğrulama işlemlerini gerçekleştirmek için birkaç örnek yapılabilir.

5. ÜNİTE: FORMLAR

KAZANIM 5.1. FORMLARI, FORM SINIFLARINI, KONTROL SINIFLARINI VE KONTEYNER SINIFLARINI TANIR.

KAZANIM 5.2. MENÜLER, MENUSTRİP KONTROLÜ VE CONTEXTMENUSTRİP KONTROLLERİNİ TANIR.

KAZANIM 5.3. İLETİŞİM KUTULARINI KULLANIR.

KAZANIM 5.4. DOĞRULAMA VE VERİ BAĞLAMA İŞLEMLERİNİ YAPAR.

KAZANIM 5.1. Formları, form sınıflarını, kontrol sınıflarını ve konteyner sınıflarını tanır.

- Formları, işletim sistemi arayüzü uygulamalarında kullanılan düğmeler ve metin kutuları temel arayüz bileşenlerini kullanma anlatılır.
- Pencere oluşturma, diğer giriş kontrollerini form üzerine yerleştirme ve form sınıfları açıklanır.
- Kontrol sınıfları (butonlar, metin kutuları, etiketler vb.) ve bu kontrolleri bir form içine yerleştirme anlatılır.
- Formdaki bileşenleri gruplamak amacıyla konteyner sınıflarının kullanılması, formdaki bileşenlerin düzenli bir şekilde yerleştirilmesi ve kullanıcı arayüzünün organize edilmesi çalışılır.

```csharp
using System;
using System.Windows.Forms;

namespace TemelArayuzUygulamasi
{
    public partial class AnaForm : Form
    {
        // Form başlatıldığında çalışacak kodlar
        public AnaForm()
        {
            InitializeComponent();
        }

        // "Merhaba" düğmesine tıklandığında çalışacak kodlar
        private void merhabaDugmesi_Click(object sender, EventArgs e)
        {
            // Kullanıcının adını al
            string kullaniciAdi = adTextBox.Text;

            // Merhaba düğmesine tıklandığında bir iletişim kutusu göster
            MessageBox.Show("Merhaba, " + kullaniciAdi + "!");
        }
    }
}
```

Formdaki temel arayüz bileşenleri örnek

Yandaki örnekte, bir form **(AnaForm)** oluşturulmuş ve bu form üzerine bir düğme **(merhabaDugmesi)** ve bir metin kutusu **(adTextBox)** eklenmiştir. Form başlatıldığında çalışacak kodlar, **AnaForm** sınıfının kurucu metodunda bulunmaktadır. Düğmeye tıklandığında çalışacak kodlar ise **merhabaDugmesi_Click** metodunda bulunmaktadır.

Bu kod örneği, kullanıcıya bir isim girmesi için bir metin kutusu ve "Merhaba" düğmesine tıkladığında bir iletişim kutusu ile karşılamak için bir düğme içermektedir. WinForms teknolojisi, daha karmaşık uygulamalarda kullanılabilen birçok farklı araç ve bileşen sunar.

KAZANIM 5.2. Menüler, MenuStrip kontrolü ve ContextMenuStrip kontrollerini tanır.

- Kullanıcı arayüzünü özelleştirmek için MenuStrip kontrolü ve ContextMenuStrip kontrolünün eklendiği örnekler uygulatılır.
- Ana menüler ve alt menüler oluşturarak uygulamanın farklı bölümlerine hızlı erişim işlemi incelenir.
- Etkileşimi artırmak için kullanıcıların belirli nesnelere sağ tıklandığında açılan dinamik menülerin oluşturulması anlatılır.
- Kontrol seçeneklerini özelleştirme ve kullanıcılar için etkileşimli menülerin entegre edilmesi C# programlama arayüzünde gerçekleştirilir.

- Windows Forms uygulamalarında menüler, kullanıcı arayüzünü düzenlemenin ve kullanıcının uygulamadaki işlevlere erişmesini sağlamanın önemli bir yolu olarak kullanılır.
  C# dilinde MenuStrip kontrolü ve ContextMenuStrip kontrolü, menüler oluşturmak için kullanılan temel kontrollerdir.

- Bir örnekle, bir MenuStrip içinde "Dosya" adlı bir menü ve bu menünün altında "Yeni", "Aç", "Çıkış" öğelerini içeren bir menü örneği oluşturulabilir. Ayrıca etkileşimi artırmak için bir ContextMenuStrip oluşturularak sağ tıklama menüsü için "Kopyala" ve "Yapıştır" öğeleri eklenebilir.

KAZANIM 5.3. İletişim kutularını kullanır.

- MessageBox, OpenFileDialog, SaveFileDialog iletişim kutularını kullanıcılarla iletişim kurmak için eklemesi anlatılır.
- Kullanıcıya bilgi vermek, doğrulama yapmak veya hataları bildirmek için iletişim kutularını kullanarak kullanıcıdan veri alma ve uyarı mesajları ekleme örnekleri incelenir.
- Farklı senaryolar için iletişim kutularının mesaj içeriklerinin, başlıklarının ve düğme metinlerinin kişiselleştirilmesi uygulamaları ile dinamik ve kullanıcı dostu arayüzler oluşturulması durumları fark ettirilir.

KAZANIM 5.4. Doğrulama ve veri bağlama işlemlerini yapar.

- Kullanıcı tarafından sağlanan verilerin doğruluğunu kontrol etmek için kullanıcı girişi doğrulama işlemleri (validasyon) icra ettirilir.
- Veri bağlama ve verileri kullanıcı arayüzüne bağlama işlemleri anlatılır.
- Kullanıcı arayüzü ile etkileşimli uygulamalar geliştirmek için doğrulama ve veri bağlama örnekleri uygulatılır.

Değerler

- Sabır (Kazanım 5.2., 5.4.)
- Öz denetim (Kazanım 5.3.)
- Sorumluluk (Kazanım 5.1., 5.2., 5.3., 5.4.)
- Saygı (Kazanım 5.1., 5.2., 5.3., 5.4.)

Alan Becerileri

- Soyutlama Becerisi (Kazanım 5.1., 5.3.)
- Algoritmik Düşünme Becerisi (Kazanım 5.2.)
- Mantıksal Düşünme Becerisi (Kazanım 5.4.)
- Veri Toplama, İşleme, Dönüştürme Becerisi (Kazanım 5.3., 5.4.)

6. ÜNİTE: VERİ TABANI İŞLEMLERİ

Öğretme yaklaşımları

Veri tabanları, C# uygulamaları ile entegre edilerek verilerin depolanması, güncellenmesi ve alınması gibi işlemleri gerçekleştirmek için yaygın olarak kullanılır. Veri tabanı yazılımlarını kurma ve C# uygulamalarıyla entegre etme süreci açıklanır. Geliştirilen projenin amacına yönelik veri tabanı seçimi önemsenir. SQL komutlarıyla veri tabanına bağlanma işleminin kavranması sağlanır. Veri tabanına ekleme, silme, güncelleme işlemleri yapılır.

6. ÜNİTE: VERİ TABANI İŞLEMLERİ

KAZANIM 6.1. VERİ TABANI YAZILIMLARINI C# ARAYÜZÜ İLE ENTEGRE EDER.

KAZANIM 6.2. VERİ TABANI TASARIMI YAPAR.

KAZANIM 6.3. TABLO İŞLEMLERİNİ GERÇEKLEŞTİRİR.

KAZANIM 6.4. TEMEL SQL KOMUTLARINI KULLANIR.

KAZANIM 6.5. İLİŞKİSEL VERİ TABANLARI OLUŞTURUR.

KAZANIM 6.6. C# PROGRAMLAMA ARAYÜZÜ İLE OLUŞTURULAN VERİ TABANLARINA BAĞLANTI SAĞLAR.

KAZANIM 6.1. Veri tabanı yazılımlarını C# arayüzü ile entegre eder.

- Veri tabanı yazılımlarını kurma ve C# uygulamalarıyla entegre etme anlatılır.
- Projelerde uygun veri tabanını seçmek için veri tabanı yazılım türleri açıklanır.
- C# programlama arayüzü ile veri tabanları arasında veri alışverişi yapma ve veri tabanı bağlantısını yönetme örnekleri inceletilir.

PROJELERDE UYGUN VERİ TABANINI SEÇMEK

KAZANIM 6.2. Veri tabanı tasarımı yapar.

- Veri tabanı tasarımı yapmayı ve veri tablolarını, alanlarını, ilişkilerini planlama anlatılır.
- Veri tabanı şeması oluştururken verilerin tutarlı ve erişilebilir olmasını sağlayacak durumlar vurgulanır.
- Farklı veri tabanı tasarım araçlarını kullanarak tasarım sürecini görselleştirme ve yönetme durumları örneklendirilir.

KAZANIM 6.3. Tablo işlemlerini gerçekleştirir.

- Veri tabanında tablo oluşturma, veri ekleme, güncelleme ve silme işlemleri anlatılır.
- Veri tabanında oluşturulan tabloları sorgulama ve filtreleme işlemlerine çalışılır.
- İşlevsel uygulamalar için tablo işlemlerini iş mantığıyla entegre etme durumu fark ettirilir.

KAZANIM 6.4. Temel SQL komutlarını kullanır.

- Select, Insert, Update, Delete temel SQL komutlarını kullanarak verileri çekme, veri ekleme, var olan veriyi güncelleme ve silme işlemleri uygulatılır.
- Veri tabanı sorguları oluştururken dinamik parametrelerle çalışma ve güvenli sorgu oluşturma becerileri geliştirilir.
- Karmaşık SQL sorguları (Join operatörleri, alt sorgular, indeksleme vb.) ile veri tabanlarını etkili bir şekilde sorgulama vurgulanır.

KAZANIM 6.5. İlişkisel veri tabanları oluşturur.

- Farklı tablolar arasındaki ilişkileri çözmesi için ilişkisel veri tabanlarının kullanımı ifade edilir.
- Veri tabanı modellemesi için veri tabanı şeması düzgün biçimde normalleştirilir.
- İlişkisel veri tabanlarının verileri tutma ve ilişkilendirme yeteneklerini kullanarak karmaşık veri tabanı yapıları oluşturulur.

KAZANIM 6.6. C# programlama arayüzü ile oluşturulan veri tabanlarına bağlantı sağlar.

- C# programlama arayüzünde oluşturulan tablolardan veri çekme, veri güncelleme ve veri silme temel işlemleri uygulatılır.

- Veri tabanı etkileşimli formlar tasarlatarak kullanıcıların arayüz üzerinden verilere erişme senaryoları gerçekleştirilir.

Değerler

- Sabır (Kazanım 6.2., 6.4.)
- Öz denetim (Kazanım 6.1., 6.3.)
- Sorumluluk (Kazanım 6.1., 6.2., 6.3., 6.4., 6.5., 6.6.)
- Saygı (Kazanım 6.1., 6.2., 6.3., 6.4., 6.5., 6.6.)

Alan Becerileri

- Soyutlama Becerisi (Kazanım 6.1., 6.3.)
- Algoritmik Düşünme Becerisi (Kazanım 6.2., 6.4.)
- Veri Toplama, İşleme, Dönüştürme Becerisi (Kazanım 6.2., 6.5.)
- Veri Gösterimi ve Analizi (Kazanım 6.3., 6.4., 6.5.)
- Model Doğrulama (Kazanım 6.6.)
- Test ve Doğrulama (Kazanım 6.6.)