EKLER

EK-1
TEMEL ALGORİTMA PROBLEMLERİ C# ÇÖZÜMLERİ

Tablo 1. TAP ChatGPT çözümleri

NO	ChatGPT

1	`using System;`
	`class Program {`
	`  static void Main() {`
	`  Console.Write("Sayı girin: ");`
	`  int n = int.Parse(Console.ReadLine());`
	`  int basamak = n.ToString().Length;`
	`  Console.WriteLine("Basamak sayısı: " + basamak);`
	`  }`
	`}`

2	`using System;`
	`class Program {`
	`  static void Main() {`
	`  int[] dizi = new int[10];`
	`  for (int i = 0; i < 10; i++) {`
	`  Console.Write("Sayı girin: ");`
	`  dizi[i] = int.Parse(Console.ReadLine());`
	`  }`
	`  }`
	`}`

3	`using System;`
	`class Program {`
	`  static void Main() {`
	`  int a = int.Parse(Console.ReadLine());`
	`  int b = int.Parse(Console.ReadLine());`
	`  if (a > 50 && b > 50) {`
	`  int c = a + b;`
	`  Console.WriteLine("Toplam: " + c);`
	`  } else {`
	`  Console.WriteLine("Sayılar uygun değil.");`
	`  }`
	`  }`
	`}`

4	`using System;`
	`class Program {`
	`  static void Main() {`
	`  int yil = int.Parse(Console.ReadLine());`
	`  bool artik = (yil % 4 == 0 && yil % 100 != 0) || (yil % 400 == 0);`
	`  Console.WriteLine(artik ? "Artık yıl" : "Artık yıl değil");`
	`  }`
	`}`

5	`using System;`
	`class Program {`
	`  static void Main() {`
	`  Console.Write("Terim sayısını girin: ");`
	`  int n = int.Parse(Console.ReadLine());`
	`  double toplam = 0;`
	`  for (int i = 0; i < n; i++) {`
	`  int payda = 1 + 2 * i;`
	`  double terim = 1.0 / payda;`

```
            toplam += (i % 2 == 0) ? terim : -terim;
        }
        Console.WriteLine("Toplam: " + toplam);
    }
}
```

6

```
using System;
class Program {
    static void Main() {
        int[] fib = new int[10];
        fib[0] = 0;
        fib[1] = 1;
        for (int i = 2; i < 10; i++)
            fib[i] = fib[i - 1] + fib[i - 2];
        foreach (int sayi in fib)
            Console.Write(sayi + " ");
    }
}
```

7

```
using System;
class Program {
    static void Main() {
        int[] notlar = new int[50];
        int basarili = 0;
        for (int i = 0; i < 50; i++) {
            Console.Write("Not girin: ");
            notlar[i] = int.Parse(Console.ReadLine());
            if (notlar[i] >= 50) basarili++;
        }
        Console.WriteLine("Başarılı öğrenci sayısı: " + basarili);
    }
}
```

8

```
using System;
class Program {
    static void Main() {
        Console.Write("Kelime girin: ");
        string kelime = Console.ReadLine().ToLower();
        int sesli = 0;
        foreach (char c in kelime)
            if ("aeıioöuü".Contains(c)) sesli++;
        double oran = (double)sesli / kelime.Length * 100;
        Console.WriteLine("Sesli harf oranı: %" + oran.ToString("F2"));
    }
}
```

9

```
using System;
class Program {
    static void Main() {
        int[] dizi = { 3, 5, 7, 10, 15, 20 };
        Console.Write("Aranan sayı: ");
        int aranan = int.Parse(Console.ReadLine());
        bool bulundu = false;
        for (int i = 0; i < dizi.Length; i++) {
            if (dizi[i] == aranan) {
                Console.WriteLine("Bulundu: İndis " + i);
```

```
                bulundu = true;
                break;
            }
        }
        if (!bulundu) Console.WriteLine("Bulunamadı.");
    }
}
```

10

```
using System;
class Program {
    static void QuickSort(int[] dizi, int sol, int sag) {
        if (sol >= sag) return;
        int pivot = dizi[(sol + sag) / 2];
        int i = sol, j = sag;
        while (i <= j) {
            while (dizi[i] < pivot) i++;
            while (dizi[j] > pivot) j--;
            if (i <= j) {
                int tmp = dizi[i];
                dizi[i] = dizi[j];
                dizi[j] = tmp;
                i++; j--;
            }
        }
        QuickSort(dizi, sol, j);
        QuickSort(dizi, i, sag);
    }
    static void Main() {
        int[] dizi = { 33, 2, 52, 106, 73, 10 };
        QuickSort(dizi, 0, dizi.Length - 1);
        foreach (int s in dizi) Console.Write(s + " ");
    }
}
```

Tablo 2. TAP Gemini çözümleri

NO	GEMİNİ

1	`using System;`
	`public class NumberDigits`
	`{`
	`  public static void Main(string[] args)`
	`  {`
	`  Console.Write("Bir sayı girin: ");`
	`  int number = Convert.ToInt32(Console.ReadLine());`
	`  int digitCount = 0;`
	`  if (number == 0)`
	`  {`
	`  digitCount = 1;`
	`  }`
	`  else`
	`  {`
	`  number = Math.Abs(number);`
	`  while (number > 0)`
	`  {`
	`  number /= 10;`
	`  digitCount++;`
	`  }`
	`  }`
	`  Console.WriteLine($"Girilen sayı {digitCount} basamaklıdır.");`
	`  }`
	`}`

2	`using System;`
	`public class ArrayInput`
	`{`
	`  public static void Main(string[] args)`
	`  {`
	`  int[] numbers = new int[10];`
	`  Console.WriteLine("10 adet sayı giriniz:");`
	`  for (int i = 0; i < 10; i++)`
	`  {`
	`  Console.Write($"Sayı {i + 1}: ");`
	`  numbers[i] = Convert.ToInt32(Console.ReadLine());`
	`  }`
	`  Console.WriteLine("Girilen sayılar:");`
	`  foreach (int num in numbers)`
	`  {`
	`  Console.Write($"{num} ");`
	`  }`
	`  }`
	`}`

3

```
using System;
public class ConditionalSum
{
    public static void Main(string[] args)
    {
        Console.Write("a sayısını girin: ");
        int a = Convert.ToInt32(Console.ReadLine());
        Console.Write("b sayısını girin: ");
        int b = Convert.ToInt32(Console.ReadLine());
        if (a > 50 && b > 50)
        {
            int c = a + b;
            Console.WriteLine($"c = a + b = {c}");
        }
        else
        {
            Console.WriteLine("Bu sayılar uygun değil.");
        }
    }
}
```

4

```
using System;
public class LeapYear
{
    public static void Main(string[] args)
    {
        Console.Write("Bir yıl girin: ");
        int year = Convert.ToInt32(Console.ReadLine());
        if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
        {
            Console.WriteLine($"{year} bir artık yıldır.");
        }
        else
        {
            Console.WriteLine($"{year} bir artık yıl değildir.");
        }
    }
}
```

5

```
using System;
public class SeriesSum
{
    public static void Main(string[] args)
    {
        Console.Write("Serinin kaç terimini hesaplamak istersiniz (n): ");
        int n = Convert.ToInt32(Console.ReadLine());
        double sum = 0.0;
        for (int i = 0; i < n; i++)
        {
            double term = 1.0 / (2 * i + 1);
            if (i % 2 == 1) // Tek indeksler için çıkarma
            {
                sum -= term;
            }
```

```
            else // Çift indeksler için toplama
            {
                sum += term;
            }
        }
        Console.WriteLine($"Serinin ilk {n} teriminin toplamı: {sum:F4}");
    }
}
```

6

```
using System;
public class Fibonacci
{
    public static void Main(string[] args)
    {
        int[] fibonacci = new int[10];
        fibonacci[0] = 0;
        fibonacci[1] = 1;
        for (int i = 2; i < 10; i++)
        {
            fibonacci[i] = fibonacci[i - 1] + fibonacci[i - 2];
        }
        Console.WriteLine("Fibonacci serisinin ilk 10 terimi:");
        foreach (int num in fibonacci)
        {
            Console.Write($"{num} ");
        }
    }
}
```

7

```
using System;
public class StudentSuccess
{
    public static void Main(string[] args)
    {
        int[] grades = new int[50]; // Varsayımsal olarak notları tutacak
dizi
        int successfulStudents = 0;
        // Notları rastgele oluşturalım veya kullanıcıdan alalım
        Random rand = new Random();
        for (int i = 0; i < 50; i++)
        {
            grades[i] = rand.Next(0, 101); // 0-100 arası rastgele not
            // Console.Write($"Öğrenci {i + 1} notu: {grades[i]} "); //
Notları görmek için
            if (grades[i] >= 50)
            {
                successfulStudents++;
            }
        }
        Console.WriteLine($"Başarılı öğrenci sayısı:
{successfulStudents}");
    }
}
```

8

```
using System;
public class VowelPercentage
```

```
{
    public static void Main(string[] args)
    {
        Console.Write("Maksimum 20 karakterli bir kelime girin: ");
        string word = Console.ReadLine();
        if (word.Length > 20)
        {
            Console.WriteLine("Kelime 20 karakterden uzun olamaz.");
            return;
        }
        int totalChars = word.Length;
        int vowelCount = 0;
        string vowels = "aeıioöuüAEIİOÖUÜ"; // Türkçe sesli harfler
        foreach (char c in word)
        {
            if (vowels.Contains(c))
            {
                vowelCount++;
            }
        }
        if (totalChars > 0)
        {
            double percentage = (double)vowelCount / totalChars * 100;
            Console.WriteLine($"Sesli harf oranı: {percentage:F2}%");
        }
        else
        {
            Console.WriteLine("Boş kelime girdiniz.");
        }
    }
}
```

9

```
using System;
public class SequentialSearch
{
    public static void Main(string[] args)
    {
        int[] arr = { 12, 34, 56, 78, 90, 23, 45, 67, 89, 10 };
        Console.Write("Aranacak sayıyı girin: ");
        int target = Convert.ToInt32(Console.ReadLine());
        int foundIndex = -1;
        for (int i = 0; i < arr.Length; i++)
        {
            if (arr[i] == target)
            {
                foundIndex = i;
                break; // Sayı bulunduğunda döngüyü sonlandır
            }
        }
        if (foundIndex != -1)
        {
            Console.WriteLine($"Sayı {target}, dizinin {foundIndex}.
indeksinde bulundu.");
```

```
        }
        else
        {
            Console.WriteLine($"Sayı {target}, dizide bulunamadı.");
        }
    }
}
```

10

```
using System;
public class QuickSortExample
{
    public static void Main(string[] args)
    {
        int[] arr = { 10, 7, 8, 9, 1, 5 };
        Console.WriteLine("Sıralanmamış dizi: " + string.Join(", ", arr));
        QuickSort(arr, 0, arr.Length - 1);
        Console.WriteLine("Sıralanmış dizi: " + string.Join(", ", arr));
    }
    public static void QuickSort(int[] arr, int low, int high)
    {
        if (low < high)
        {
            int pi = Partition(arr, low, high);
            QuickSort(arr, low, pi - 1);
            QuickSort(arr, pi + 1, high);
        }
    }
    public static int Partition(int[] arr, int low, int high)
    {
        int pivot = arr[high];
        int i = (low - 1);
        for (int j = low; j < high; j++)
        {
            if (arr[j] < pivot)
            {
                i++;
                Swap(arr, i, j);
            }
        }
        Swap(arr, i + 1, high);
        return i + 1;
    }
    public static void Swap(int[] arr, int i, int j)
    {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
}
```

Tablo 3. TAP Copilot çözümleri

NO	Copilot

1	`using System;`
	`class Program`
	`{`
	`  static void Main()`
	`  {`
	`  Console.Write("Bir sayı giriniz: ");`
	`  string input = Console.ReadLine();`
	`  if (int.TryParse(input, out int number))`
	`  {`
	`  int digitCount = input.Length;`
	`  Console.WriteLine($"Girilen sayı {digitCount} basamaklıdır.");`
	`  }`
	`  else`
	`  {`
	`  Console.WriteLine("Geçerli bir sayı giriniz.");`
	`  }`
	`  }`
	`}`

2	`using System;`
	`class Program`
	`{`
	`  static void Main()`
	`  {`
	`  int[] numbers = new int[10];`
	`  Console.WriteLine("10 elemanlı bir dizi giriniz:");`
	`  for (int i = 0; i < 10; i++)`
	`  {`
	`  Console.Write($"Eleman {i + 1}: ");`
	`  numbers[i] = int.Parse(Console.ReadLine());`
	`  }`
	`  Console.WriteLine("\nGirilen Dizi:");`
	`  foreach (var num in numbers)`
	`  {`
	`  Console.Write(num + " ");`
	`  }`
	`  }`
	`}`

3

```
using System;
class Program
{
    static void Main()
    {
        Console.Write("a sayısını giriniz: ");
        int a = int.Parse(Console.ReadLine());
        Console.Write("b sayısını giriniz: ");
        int b = int.Parse(Console.ReadLine());
        if (a > 50 && b > 50)
            Console.WriteLine($"Sonuç: {a + b}");
        else
            Console.WriteLine("Bu sayılar uygun değil.");
    }
}
```

4

```
using System;
class Program
{
    static void Main()
    {
        Console.Write("Yıl giriniz: ");
        int yil = int.Parse(Console.ReadLine());
        if ((yil % 4 == 0 && yil % 100 != 0) || (yil % 400 == 0))
            Console.WriteLine($"{yil} artık yıldır.");
        else
            Console.WriteLine($"{yil} artık yıl değildir.");
    }
}
```

5

```
using System;
class Program
{
    static void Main()
    {
        Console.Write("n terim sayısını giriniz: ");
        int n = int.Parse(Console.ReadLine());
        double sum = 0;
        for (int i = 0; i < n; i++)
        {
            sum += (i % 2 == 0 ? 1.0 : -1.0) / (2 * i + 1);
        }
        Console.WriteLine($"Seri toplamı: {sum}");
    }
}
```

6

```
using System;
class Program
{
    static void Main()
    {
        int[] fibonacci = new int[10];
        fibonacci[0] = 0;
        fibonacci[1] = 1;
        for (int i = 2; i < 10; i++)
```

```
        {
            fibonacci[i] = fibonacci[i - 1] + fibonacci[i - 2];
        }
        Console.WriteLine("Fibonacci Serisi:");
        foreach (var num in fibonacci)
        {
            Console.Write(num + " ");
        }
    }
}
```

7

```
using System;
class Program
{
    static void Main()
    {
        int[] notlar = new int[50];
        int basariliSayisi = 0;
        for (int i = 0; i < 50; i++)
        {
            Console.Write($"Öğrenci {i + 1} notu: ");
            notlar[i] = int.Parse(Console.ReadLine());
            if (notlar[i] >= 50) basariliSayisi++;
        }
        Console.WriteLine($"Başarılı öğrenci sayısı: {basariliSayisi}");
    }
}
```

8

```
using System;
class Program
{
    static void Main()
    {
        Console.Write("Bir kelime giriniz (maks. 20 karakter): ");
        string kelime = Console.ReadLine().ToLower();
        int sesliSayisi = 0;
        foreach (char c in kelime)
        {
            if ("aeiouöüı".Contains(c)) sesliSayisi++;
        }
        double oran = (double)sesliSayisi / kelime.Length * 100;
        Console.WriteLine($"Sesli harf oranı: {oran:F2}%");
    }
}
```

9

```
using System;
class Program
{
    static void Main()
    {
        int[] dizi = { 3, 7, 12, 18, 25, 31, 42, 58, 67, 80 };
        Console.Write("Aramak istediğiniz sayıyı giriniz: ");
        int aranan = int.Parse(Console.ReadLine());
        int index = -1;
        for (int i = 0; i < dizi.Length; i++)
```

```
        {
            if (dizi[i] == aranan)
            {
                index = i;
                break;
            }
        }
        if (index != -1)
            Console.WriteLine($"Sayı {index}. indiste bulundu.");
        else
            Console.WriteLine("Sayı dizide bulunamadı.");
    }
}
```

10

```
using System;
class Program
{
    static void QuickSort(int[] dizi, int sol, int sag)
    {
        if (sol >= sag) return;
        int pivot = dizi[sag];
        int i = sol;
        for (int j = sol; j < sag; j++)
        {
            if (dizi[j] < pivot)
            {
                (dizi[i], dizi[j]) = (dizi[j], dizi[i]);
                i++;
            }
        }
        (dizi[i], dizi[sag]) = (dizi[sag], dizi[i]);
        QuickSort(dizi, sol, i - 1);
        QuickSort(dizi, i + 1, sag);
    }
    static void Main()
    {
        int[] dizi = { 34, 7, 23, 32, 5, 62 };
        Console.WriteLine("Orijinal dizi: " + string.Join(", ", dizi));
        QuickSort(dizi, 0, dizi.Length - 1);
        Console.WriteLine("Sıralanmış dizi: " + string.Join(", ", dizi));
    }
}
```

Tablo 4. TAP Claude çözümleri

NO	Claude

1	`using System;`
	`class Program{`
	`static void Main(){`
	`int sayi=int.Parse(Console.ReadLine());`
	`int basamak=sayi==0?1:Math.Abs(sayi).ToString().Length;`
	`Console.WriteLine(basamak);`
	`}}`

2	`using System;`
	`class Program{`
	`static void Main(){`
	`int[]dizi=new int[10];`
	`for(int i=0;i<10;i++){`
	`dizi[i]=int.Parse(Console.ReadLine());`
	`}`
	`foreach(int sayi in dizi)Console.Write(sayi+" ");`
	`}}`

3	`using System;`
	`class Program{`
	`static void Main(){`
	`int a=int.Parse(Console.ReadLine());`
	`int b=int.Parse(Console.ReadLine());`
	`if(a>50&&b>50){`
	`int c=a+b;`
	`Console.WriteLine(c);`
	`}else Console.WriteLine("Bu sayılar uygun değil");`
	`}}`

4	`using System;`
	`class Program{`
	`static void Main(){`
	`int yil=int.Parse(Console.ReadLine());`
	`bool artikYil=(yil%4==0&&yil%100!=0)||(yil%400==0);`
	`Console.WriteLine(artikYil?"Artık yıl":"Artık yıl değil");`
	`}}`

5	`using System;`
	`class Program{`
	`static void Main(){`
	`int n=int.Parse(Console.ReadLine());`
	`double toplam=0;`
	`for(int i=1;i<=n;i++){`
	`toplam+=Math.Pow(-1,i-1)/(2*i-1);`
	`}`
	`Console.WriteLine(toplam);`
	`}}`

6	`using System;`
	`class Program{`
	`static void Main(){`
	`int[]fib=new int[10];`
	`fib[0]=0;fib[1]=1;`
	`for(int i=2;i<10;i++){`

```
fib[i]=fib[i-1]+fib[i-2];
}
foreach(int sayi in fib)Console.Write(sayi+" ");
}}
```

7

```
using System;
class Program{
static void Main(){
int[]notlar=new int[50];
int basarili=0;
for(int i=0;i<50;i++){
notlar[i]=int.Parse(Console.ReadLine());
if(notlar[i]>=50)basarili++;
}
Console.WriteLine(basarili);
}}
```

8

```
using System;
class Program{
static void Main(){
string kelime=Console.ReadLine();
int sesli=0;
foreach(char c in kelime.ToLower()){
if("aeiouıüöiuö".Contains(c))sesli++;
}
double yuzde=(double)sesli/kelime.Length*100;
Console.WriteLine($"{yuzde:F2}%");
}}
```

9

```
using System;
class Program{
static void Main(){
int[]dizi={1,3,5,7,9,11,13,15,17,19};
int aranan=int.Parse(Console.ReadLine());
int konum=-1;
for(int i=0;i<dizi.Length;i++){
if(dizi[i]==aranan){konum=i;break;}
}
Console.WriteLine(konum==-1?"Bulunamadı":$"Konum: {konum}");
}}
```

10

```
using System;
class Program{
static void Main(){
int[]dizi={64,34,25,12,22,11,90};
QuickSort(dizi,0,dizi.Length-1);
foreach(int sayi in dizi)Console.Write(sayi+" ");
}
static void QuickSort(int[]arr,int low,int high){
if(low<high){
int pi=Partition(arr,low,high);
QuickSort(arr,low,pi-1);
QuickSort(arr,pi+1,high);
}}
static int Partition(int[]arr,int low,int high){
int pivot=arr[high];
```

```
int i=low-1;
for(int j=low;j<high;j++){
if(arr[j]<pivot){
i++;
Swap(arr,i,j);
}}
Swap(arr,i+1,high);
return i+1;
}
static void Swap(int[]arr,int i,int j){
int temp=arr[i];
arr[i]=arr[j];
arr[j]=temp;
}}
```

EK-2
NESNEYE DAYALI PROGRAMLAMA PROBLEMLERİ C# ÇÖZÜMLERİ

Tablo 5. NDP ChatGPT çözümleri

NO	ChatGPT
1	

```
using System;
public class Kisi
{
    // Kisi sınıfının özellikleri
    public string Ad { get; set; }
    public int Yas { get; set; }
    // Yapıcı metod (Constructor)
    public Kisi(string ad, int yas)
    {
        Ad = ad;
        Yas = yas;
    }
    // Bilgileri yazdıran metot
    public void BilgileriYazdir()
    {
        Console.WriteLine($"Ad: {Ad}");
        Console.WriteLine($"Yaş: {Yas}");
    }
}
class Program
{
    static void Main()
    {
        // Kisi sınıfından yeni bir nesne oluşturuyoruz
        Kisi kisi1 = new Kisi("Ahmet", 30);
        // Bilgileri yazdırma metodu çağrılıyor
        kisi1.BilgileriYazdir();
    }
}
```

2

```
using System;
public class BankAccount
{
    // Bakiye değişkeni kapsülleme (encapsulation) ile private olarak tanımlandı
    private decimal bakiye;
    // Hesap sahibinin adı
    public string HesapSahibi { get; set; }
    // Yapıcı metod (Constructor)
    public BankAccount(string hesapSahibi, decimal baslangicBakiyesi)
    {
        HesapSahibi = hesapSahibi;
        bakiye = baslangicBakiyesi; // Başlangıç bakiyesi atanıyor
    }
    // Para yatırma metodu
    public void ParaYatir(decimal miktar)
    {
        if (miktar > 0)
        {
            bakiye += miktar;
            Console.WriteLine($"{miktar} TL yatırıldı. Güncel bakiye: {bakiye}
TL");
        }
        else
        {
            Console.WriteLine("Yatırılacak miktar pozitif olmalıdır!");
        }
    }
    // Para çekme metodu
    public void ParaCek(decimal miktar)
    {
        if (miktar > 0 && miktar <= bakiye)
        {
            bakiye -= miktar;
            Console.WriteLine($"{miktar} TL çekildi. Güncel bakiye: {bakiye} TL");
        }
        else
        {
            Console.WriteLine("Yetersiz bakiye veya geçersiz işlem!");
        }
    }
    // Bakiye sorgulama metodu
    public void BakiyeSorgula()
    {
        Console.WriteLine($"Hesap Sahibi: {HesapSahibi}");
        Console.WriteLine($"Güncel Bakiye: {bakiye} TL");
```

```
    }
}
class Program
{
    static void Main()
    {
        // Yeni bir BankAccount nesnesi oluşturuluyor
        BankAccount hesap = new BankAccount("Ahmet Yılmaz", 1000);
        // Para yatırma ve çekme işlemleri
        hesap.ParaYatir(500);
        hesap.ParaCek(300);
        hesap.BakiyeSorgula();
    }
}
```

3

```
using System;
public class Arac
{
    public string Marka { get; set; }
    public int Hiz { get; protected set; }
    public Arac(string marka)
    {
        Marka = marka;
        Hiz = 0; // Başlangıç hızı 0
    }
    // Sanal hızlanma metodu (override edilebilir)
    public virtual void Hizlan()
    {
        Hiz += 10;
        Console.WriteLine($"{Marka} hızlandı. Yeni hız: {Hiz} km/h");
    }
}
// Araba sınıfı, Arac sınıfından türetiliyor
public class Araba : Arac
{
    public Araba(string marka) : base(marka) { }
    // Hızlan metodu override ediliyor (arabalar daha hızlı hızlanır)
    public override void Hizlan()
```

```
    {
        Hiz += 20;
        Console.WriteLine($"{Marka} araba hızlandı. Yeni hız: {Hiz} km/h");
    }
}
// Bisiklet sınıfı, Arac sınıfından türetiliyor
public class Bisiklet : Arac
{
    public Bisiklet(string marka) : base(marka) { }
    // Hızlan metodu override ediliyor (bisikletler daha yavaş hızlanır)
    public override void Hizlan()
    {
        Hiz += 5;
        Console.WriteLine($"{Marka} bisiklet hızlandı. Yeni hız: {Hiz} km/h");
    }
}
class Program
{
    static void Main()
    {
        // Araba ve Bisiklet nesneleri oluşturuluyor
        Araba araba1 = new Araba("Toyota");
        Bisiklet bisiklet1 = new Bisiklet("Bianchi");
        // Araçları hızlandırıyoruz
        araba1.Hizlan();
        bisiklet1.Hizlan();
        araba1.Hizlan();
        bisiklet1.Hizlan();
    }
}
```

4

```
using System;
public abstract class Sekil
{
    // Alan hesaplama metodunu abstract olarak tanımlıyoruz
    public abstract double AlanHesapla();
}
// Daire sınıfı, Sekil sınıfından türetiliyor
public class Daire : Sekil
{
    public double YariCap { get; set; }
    public Daire(double yariCap)
    {
        YariCap = yariCap;
    }
    // Dairenin alanı: π * r² formülüyle hesaplanıyor
```

```
    public override double AlanHesapla()
    {
        return Math.PI * YariCap * YariCap;
    }
}
// Kare sınıfı, Sekil sınıfından türetiliyor
public class Kare : Sekil
{
    public double Kenar { get; set; }
    public Kare(double kenar)
    {
        Kenar = kenar;
    }
    // Karenin alanı: Kenar * Kenar formülüyle hesaplanıyor
    public override double AlanHesapla()
    {
        return Kenar * Kenar;
    }
}
class Program
{
    static void Main()
    {
        // Daire ve Kare nesneleri oluşturuluyor
        Sekil daire = new Daire(5);  // Yarıçap = 5
        Sekil kare = new Kare(4);    // Kenar = 4
        // Alan hesaplama metodunu çağırıyoruz
        Console.WriteLine($"Dairenin Alanı: {daire.AlanHesapla():F2}");
        Console.WriteLine($"Karenin Alanı: {kare.AlanHesapla():F2}");
    }
}
```

5

```
using System;
public abstract class Calisan
{
    public string Ad { get; set; }
    public int CalismaSaati { get; set; }
    public Calisan(string ad, int calismaSaati)
    {
        Ad = ad;
        CalismaSaati = calismaSaati;
    }
    // Soyut metod (alt sınıflar tarafından zorunlu olarak uygulanacak)
    public abstract double MaasHesapla();
    // Çalışan bilgilerini yazdırma metodu
    public void BilgileriYazdir()
```

```
    {
        Console.WriteLine($"Çalışan: {Ad}, Maaş: {MaasHesapla():C}");
    }
}
// Memur sınıfı, Calisan sınıfından türetiliyor
public class Memur : Calisan
{
    private const double SaatlikUcret = 50;
    public Memur(string ad, int calismaSaati) : base(ad, calismaSaati) { }
    // Memurların maaşı saatlik ücret çarpı çalışma saati olarak hesaplanıyor
    public override double MaasHesapla()
    {
        return CalismaSaati * SaatlikUcret;
    }
}
// Yönetici sınıfı, Calisan sınıfından türetiliyor
public class Yonetici : Calisan
{
    private const double AylikMaas = 10000;
    private const double Prim = 2000;
    public Yonetici(string ad, int calismaSaati) : base(ad, calismaSaati) { }
    // Yöneticilerin maaşı sabit maaş + prim olarak hesaplanıyor
    public override double MaasHesapla()
    {
        return AylikMaas + Prim;
    }
}
class Program
{
    static void Main()
    {
        // Memur ve Yönetici nesneleri oluşturuluyor
        Calisan memur1 = new Memur("Ahmet", 160);  // 160 saat çalıştı
        Calisan yonetici1 = new Yonetici("Zeynep", 0); // Sabit maaşlı yönetici
        // Bilgileri yazdırma
        memur1.BilgileriYazdir();
        yonetici1.BilgileriYazdir();
    }
}
```

6

```
using System;
using System.Collections.Generic;
public class Kitap
{
    public string Baslik { get; set; }
    public string Yazar { get; set; }
```

```
    public Kitap(string baslik, string yazar)
    {
        Baslik = baslik;
        Yazar = yazar;
    }
    public override string ToString()
    {
        return $"{Baslik} - {Yazar}";
    }
}
public class Kutuphane
{
    private List<Kitap> kitaplar;
    public Kutuphane()
    {
        kitaplar = new List<Kitap>();
    }
    // Kitap ekleme metodu
    public void KitapEkle(Kitap yeniKitap)
    {
        kitaplar.Add(yeniKitap);
        Console.WriteLine($"Kitap eklendi: {yeniKitap}");
    }
    // Kitap silme metodu (Başlığa göre silme)
    public void KitapSil(string baslik)
    {
        Kitap silinecekKitap = kitaplar.Find(k => k.Baslik == baslik);
        if (silinecekKitap != null)
        {
            kitaplar.Remove(silinecekKitap);
            Console.WriteLine($"Kitap silindi: {baslik}");
        }
        else
        {
            Console.WriteLine($"Kitap bulunamadı: {baslik}");
        }
    }
    // Kitapları listeleme metodu
    public void KitaplariListele()
    {
        Console.WriteLine("\nKütüphane'deki Kitaplar:");
        if (kitaplar.Count == 0)
        {
            Console.WriteLine("Kütüphane boş.");
            return;
```

```
        }
        foreach (var kitap in kitaplar)
        {
            Console.WriteLine(kitap);
        }
    }
}
class Program
{
    static void Main()
    {
        Kutuphane kutuphane = new Kutuphane();
        // Kitap ekleme
        kutuphane.KitapEkle(new Kitap("1984", "George Orwell"));
        kutuphane.KitapEkle(new Kitap("Suç ve Ceza", "Fyodor Dostoyevski"));
        kutuphane.KitapEkle(new Kitap("Küçük Prens", "Antoine de Saint-Exupéry"));
        // Kitapları listeleme
        kutuphane.KitaplariListele();
        // Kitap silme
        kutuphane.KitapSil("Suç ve Ceza");
        // Güncellenmiş kitap listesi
        kutuphane.KitaplariListele();
    }
}
```

7

```
using System;
// Ses çıkarabilen nesneler için bir arayüz tanımlıyoruz
public interface ISesCikarabilir
{
    void SesCikar(); // Metot bildirimi (gövdesiz)
}
// Kedi sınıfı, ISesCikarabilir arayüzünü implemente ediyor
public class Kedi : ISesCikarabilir
{
    public void SesCikar()
    {
        Console.WriteLine("Kedi: Miyav Miyav!");
    }
}
// Köpek sınıfı, ISesCikarabilir arayüzünü implemente ediyor
public class Kopek : ISesCikarabilir
{
    public void SesCikar()
    {
        Console.WriteLine("Köpek: Hav Hav!");
    }
```

```
}
class Program
{
    static void Main()
    {
        ISesCikarabilir kedi = new Kedi();
        ISesCikarabilir kopek = new Kopek();
        // Ses çıkarma metotlarını çağırıyoruz
        kedi.SesCikar();
        kopek.SesCikar();
    }
}
```

8

```
using System;
// Özel istisna sınıfı (Bakiye yetersizse hata fırlatmak için)
public class BakiyeYetersizException : Exception
{
    public BakiyeYetersizException(string mesaj) : base(mesaj) { }
}
// Banka hesabı sınıfı
public class BankaHesabi
{
    public string HesapSahibi { get; private set; }
    private decimal Bakiye;
    public BankaHesabi(string hesapSahibi, decimal baslangicBakiyesi)
    {
        HesapSahibi = hesapSahibi;
        Bakiye = baslangicBakiyesi;
    }
    // Para yatırma metodu
    public void ParaYatir(decimal miktar)
    {
        Bakiye += miktar;
        Console.WriteLine($"Başarıyla yatırıldı: {miktar:C}. Yeni bakiye:
{Bakiye:C}");
    }
    // Para çekme metodu (Hata yönetimi içerir)
    public void ParaCek(decimal miktar)
    {
        if (miktar > Bakiye)
        {
            throw new BakiyeYetersizException($"Bakiye yetersiz! Mevcut bakiye:
{Bakiye:C}");
        }
        Bakiye -= miktar;
        Console.WriteLine($"Başarıyla çekildi: {miktar:C}. Yeni bakiye:
{Bakiye:C}");
```

```
    }
}
class Program
{
    static void Main()
    {
        try
        {
            BankaHesabi hesap = new BankaHesabi("Ahmet Yılmaz", 500);
            // Para yatırma
            hesap.ParaYatir(200);
            // Geçerli para çekme işlemi
            hesap.ParaCek(400);
            // Hatalı para çekme işlemi (Bakiye yetersiz hatası alacak)
            hesap.ParaCek(500);
        }
        catch (BakiyeYetersizException ex)
        {
            Console.WriteLine($"Hata: {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Beklenmeyen bir hata oluştu: {ex.Message}");
        }
    }
}
```

9

```
using System;
public class VeritabaniBaglantisi
{
    private static VeritabaniBaglantisi _instance;
    private static readonly object _lock = new object(); // Thread-safe için kilit
nesnesi
    // Private constructor: Dışarıdan nesne oluşturulamaz
    private VeritabaniBaglantisi()
    {
        Console.WriteLine("Veritabanı bağlantısı oluşturuldu.");
    }
    // Tek örneği döndüren metot (Lazy Initialization + Thread Safety)
    public static VeritabaniBaglantisi GetInstance()
    {
        if (_instance == null)
        {
            lock (_lock) // Çoklu thread erişiminde güvenliği sağlamak için
            {
                if (_instance == null)
                {
```

```
                    _instance = new VeritabaniBaglantisi();
                }
            }
        }
        return _instance;
    }
    public void BaglantiAc()
    {
        Console.WriteLine("Veritabanı bağlantısı açıldı.");
    }
    public void BaglantiKapat()
    {
        Console.WriteLine("Veritabanı bağlantısı kapatıldı.");
    }
}
class Program
{
    static void Main()
    {
        // Aynı nesneyi tekrar tekrar çağırıyoruz
        VeritabaniBaglantisi baglanti1 = VeritabaniBaglantisi.GetInstance();
        baglanti1.BaglantiAc();
        VeritabaniBaglantisi baglanti2 = VeritabaniBaglantisi.GetInstance();
        baglanti2.BaglantiKapat();
        // İki nesnenin de aynı olup olmadığını kontrol edelim
        Console.WriteLine($"baglanti1 == baglanti2: {baglanti1 == baglanti2}");
    }
}
```

10

```
using System;
// Hayvanlar için ortak interface tanımlıyoruz
public interface IHayvan
{
    void SesCikar();
}
// Kus sınıfı, IHayvan arayüzünü implemente ediyor
public class Kus : IHayvan
{
    public void SesCikar()
    {
        Console.WriteLine("Kuş: Cik Cik!");
    }
}
// Balık sınıfı, IHayvan arayüzünü implemente ediyor
public class Balik : IHayvan
{
```

```
    public void SesCikar()
    {
        Console.WriteLine("Balık: (Balıklar ses çıkarmaz!)");
    }
}
// Hayvan nesnelerini oluşturan fabrika sınıfı
public class HayvanFabrikasi
{
    public static IHayvan HayvanOlustur(string tur)
    {
        if (tur.ToLower() == "kus")
        {
            return new Kus();
        }
        else if (tur.ToLower() == "balik")
        {
            return new Balik();
        }
        else
        {
            throw new ArgumentException("Geçersiz hayvan türü!");
        }
    }
}
class Program
{
    static void Main()
    {
        try
        {
            // Kuş nesnesi oluştur
            IHayvan kus = HayvanFabrikasi.HayvanOlustur("kus");
            kus.SesCikar();
            // Balık nesnesi oluştur
            IHayvan balik = HayvanFabrikasi.HayvanOlustur("balik");
            balik.SesCikar();
            // Geçersiz bir tür denemesi (Hata fırlatır)
            IHayvan gecersiz = HayvanFabrikasi.HayvanOlustur("kedi");
            gecersiz.SesCikar();
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Hata: {ex.Message}");
        }
    }}
```

Tablo 6. NDP Gemini çözümleri

NO	GEMİNİ

1	using System;
	public class Kisi
	{
	// Özellikler
	public string Ad { get; set; }
	public int Yas { get; set; }
	// Yapıcı metot (constructor)
	public Kisi(string ad, int yas)
	{
	Ad = ad;
	Yas = yas;
	}
	// Bilgileri yazdıran metot
	public void BilgileriYazdir()
	{
	Console.WriteLine($"Ad: {Ad}, Yaş: {Yas}");
	}
	}
	public class Program
	{
	public static void Main(string[] args)
	{
	// Kisi sınıfından nesne oluşturma
	Kisi kisi1 = new Kisi("Ayşe", 30);
	// Bilgileri yazdırma
	kisi1.BilgileriYazdir(); // Çıktı: Ad: Ayşe, Yaş: 30
	// Başka bir nesne oluşturma
	Kisi kisi2 = new Kisi("Mehmet", 25);
	kisi2.BilgileriYazdir(); // Çıktı: Ad: Mehmet, Yaş: 25
	}}

2

```
using System;
public class BankAccount
{
    // Bakiye değişkeni private olarak tanımlandı, dışarıdan doğrudan erişilemez.
    private double bakiye;
    // Yapıcı metot (constructor)
    public BankAccount(double baslangicBakiyesi)
    {
        if (baslangicBakiyesi < 0)
        {
            Console.WriteLine("Başlangıç bakiyesi negatif olamaz. 0 olarak
ayarlandı.");
            bakiye = 0;
        }
        else
        {
            bakiye = baslangicBakiyesi;
        }
    }
    // Para yatırma metodu
    public void ParaYatir(double miktar)
    {
        if (miktar > 0)
        {
            bakiye += miktar;
            Console.WriteLine($"{miktar:C} yatırıldı. Yeni bakiye: {bakiye:C}");
        }
        else
        {
            Console.WriteLine("Geçersiz miktar. Pozitif bir değer girin.");
        }
    }
    // Para çekme metodu
    public void ParaCek(double miktar)
    {
        if (miktar > 0 && miktar <= bakiye)
        {
            bakiye -= miktar;
            Console.WriteLine($"{miktar:C} çekildi. Yeni bakiye: {bakiye:C}");
        }
        else if (miktar <= 0)
        {
            Console.WriteLine("Geçersiz miktar. Pozitif bir değer girin.");
        }
        else
```

```
        {
            Console.WriteLine("Yetersiz bakiye.");
        }
    }
    // Bakiyeyi döndüren metot (isteğe bağlı)
    public double BakiyeOgren()
    {
        return bakiye;
    }
}
public class Program
{
    public static void Main(string[] args)
    {
        // BankAccount sınıfından nesne oluşturma
        BankAccount hesap = new BankAccount(1000);
        // Para yatırma ve çekme işlemleri
        hesap.ParaYatir(500);
        hesap.ParaCek(200);
        hesap.ParaCek(1500); // Yetersiz bakiye
        // Bakiyeyi yazdırma (isteğe bağlı)
        Console.WriteLine($"Güncel bakiye: {hesap.BakiyeOgren():C}");
    }
}
```

3

```
using System;
// Üst sınıf (base class)
public class Arac
{
    public virtual void Hizlan()
    {
        Console.WriteLine("Araç hızlanıyor.");
    }
}
// Alt sınıf (derived class) - Bisiklet
public class Bisiklet : Arac
{
    public override void Hizlan()
    {
        Console.WriteLine("Bisiklet pedal çevirerek hızlanıyor.");
    }
}
// Alt sınıf (derived class) - Araba
public class Araba : Arac
{
    public override void Hizlan()
```

```
    {
        Console.WriteLine("Araba gaza basarak hızlanıyor.");
    }
}
public class Program
{
    public static void Main(string[] args)
    {
        // Arac sınıfından nesne oluşturma
        Arac arac = new Arac();
        arac.Hizlan(); // Çıktı: Araç hızlanıyor.
        // Bisiklet sınıfından nesne oluşturma
        Bisiklet bisiklet = new Bisiklet();
        bisiklet.Hizlan(); // Çıktı: Bisiklet pedal çevirerek hızlanıyor.
        // Araba sınıfından nesne oluşturma
        Araba araba = new Araba();
        araba.Hizlan(); // Çıktı: Araba gaza basarak hızlanıyor.
        // Arac tipinde değişkenlere alt sınıf nesnelerini atama (polimorfizm)
        Arac aracBisiklet = new Bisiklet();
        aracBisiklet.Hizlan(); // Çıktı: Bisiklet pedal çevirerek hızlanıyor.
        Arac aracAraba = new Araba();
        aracAraba.Hizlan(); // Çıktı: Araba gaza basarak hızlanıyor.
    }
}
```

4

```
using System;
// Üst sınıf (base class)
public class Sekil
{
    public virtual double AlanHesapla()
    {
        return 0; // Varsayılan değer
    }
}
// Alt sınıf (derived class) - Daire
public class Daire : Sekil
{
    public double Yaricap { get; set; }
    public override double AlanHesapla()
    {
        return Math.PI * Yaricap * Yaricap;
    }
}
// Alt sınıf (derived class) - Kare
public class Kare : Sekil
{
```

```
    public double Kenar { get; set; }
    public override double AlanHesapla()
    {
        return Kenar * Kenar;
    }
}
public class Program
{
    public static void Main(string[] args)
    {
        // Sekil tipinde değişkenlere alt sınıf nesnelerini atama (polimorfizm)
        Sekil daire = new Daire { Yaricap = 5 };
        Sekil kare = new Kare { Kenar = 4 };
        // Alan hesaplama
        Console.WriteLine($"Dairenin alanı: {daire.AlanHesapla()}"); // Çıktı:
Dairenin alanı: 78,53981633974483
        Console.WriteLine($"Karenin alanı: {kare.AlanHesapla()}"); // Çıktı:
Karenin alanı: 16
        // Sekil dizisi oluşturma ve polimorfizm uygulama
        Sekil[] sekiller = new Sekil[] { daire, kare };
        foreach (Sekil sekil in sekiller)
        {
            Console.WriteLine($"Şeklin alanı: {sekil.AlanHesapla()}");
        }
    }
}
```

5

```
using System;
// Soyut sınıf (abstract class)
public abstract class Calisan
{
    public string Ad { get; set; }
    public string Soyad { get; set; }
    // Soyut metot (abstract method)
    public abstract double MaasHesapla();
    // Soyut olmayan metot (normal metot)
    public void BilgileriYazdir()
    {
        Console.WriteLine($"Ad: {Ad}, Soyad: {Soyad}, Maaş: {MaasHesapla():C}");
    }
}
// Alt sınıf (derived class) - Memur
public class Memur : Calisan
{
    public double Maas { get; set; }
    public override double MaasHesapla()
    {
```

```
        return Maas;
    }
}
// Alt sınıf (derived class) - Yonetici
public class Yonetici : Calisan
{
    public double TabanMaas { get; set; }
    public double Bonus { get; set; }
    public override double MaasHesapla()
    {
        return TabanMaas + Bonus;
    }
}
public class Program
{
    public static void Main(string[] args)
    {
        // Memur nesnesi oluşturma
        Memur memur = new Memur { Ad = "Ayşe", Soyad = "Demir", Maas = 5000 };
        memur.BilgileriYazdir(); // Çıktı: Ad: Ayşe, Soyad: Demir, Maaş: 5.000,00
TL
        // Yonetici nesnesi oluşturma
        Yonetici yonetici = new Yonetici { Ad = "Mehmet", Soyad = "Yılmaz",
TabanMaas = 8000, Bonus = 2000 };
        yonetici.BilgileriYazdir(); // Çıktı: Ad: Mehmet, Soyad: Yılmaz, Maaş:
10.000,00 TL
        // Calisan tipinde dizi oluşturma ve polimorfizm uygulama
        Calisan[] calisanlar = new Calisan[] { memur, yonetici };
        foreach (Calisan calisan in calisanlar)
        {
            calisan.BilgileriYazdir();
        }
    }
}
```

6

```
using System;
using System.Collections.Generic;
// Kitap sınıfı
public class Kitap
{
    public string Baslik { get; set; }
    public string Yazar { get; set; }
    public Kitap(string baslik, string yazar)
    {
        Baslik = baslik;
        Yazar = yazar;
    }
```

```
    public override string ToString()
    {
        return $"Başlık: {Baslik}, Yazar: {Yazar}";
    }
}
// Kutuphane sınıfı
public class Kutuphane
{
    private List<Kitap> kitaplar;
    public Kutuphane()
    {
        kitaplar = new List<Kitap>();
    }
    // Kitap ekleme metodu
    public void KitapEkle(Kitap kitap)
    {
        kitaplar.Add(kitap);
        Console.WriteLine($"{kitap.Baslik} kütüphaneye eklendi.");
    }
    // Kitap silme metodu
    public void KitapSil(Kitap kitap)
    {
        if (kitaplar.Remove(kitap))
        {
            Console.WriteLine($"{kitap.Baslik} kütüphaneden silindi.");
        }
        else
        {
            Console.WriteLine($"{kitap.Baslik} kütüphanede bulunamadı.");
        }
    }
    // Tüm kitapları listeleme metodu
    public void KitaplariListele()
    {
        Console.WriteLine("Kütüphanedeki Kitaplar:");
        foreach (Kitap kitap in kitaplar)
        {
            Console.WriteLine(kitap);
        }
    }
}
public class Program
{
    public static void Main(string[] args)
    {
```

```
        // Kütüphane nesnesi oluşturma
        Kutuphane kutuphane = new Kutuphane();
        // Kitap nesneleri oluşturma
        Kitap kitap1 = new Kitap("Suç ve Ceza", "Fyodor Dostoyevski");
        Kitap kitap2 = new Kitap("1984", "George Orwell");
        Kitap kitap3 = new Kitap("Yüzüklerin Efendisi", "J.R.R. Tolkien");
        // Kitapları kütüphaneye ekleme
        kutuphane.KitapEkle(kitap1);
        kutuphane.KitapEkle(kitap2);
        kutuphane.KitapEkle(kitap3);
        // Kitapları listeleme
        kutuphane.KitaplariListele();
        // Kitap silme
        kutuphane.KitapSil(kitap2);
        // Kitapları tekrar listeleme
        kutuphane.KitaplariListele();
    }
}
```

7

```
using System;
// Arayüz (interface) tanımlama
public interface ISesCikarabilir
{
    void SesCikar();
}
// Kedi sınıfı (arayüzü uygular)
public class Kedi : ISesCikarabilir
{
    public void SesCikar()
    {
        Console.WriteLine("Miyav!");
    }
}
// Kopek sınıfı (arayüzü uygular)
public class Kopek : ISesCikarabilir
{
    public void SesCikar()
    {
        Console.WriteLine("Hav!");
    }
}
public class Program
{
    public static void Main(string[] args)
    {
        // Kedi ve Kopek nesneleri oluşturma
```

```
        Kedi kedi = new Kedi();
        Kopek kopek = new Kopek();
        // Arayüz tipinde değişkenlere nesneleri atama (polimorfizm)
        ISesCikarabilir sesCikaranKedi = kedi;
        ISesCikarabilir sesCikaranKopek = kopek;
        // Ses çıkarma metotlarını çağırma
        sesCikaranKedi.SesCikar(); // Çıktı: Miyav!
        sesCikaranKopek.SesCikar(); // Çıktı: Hav!
        // Arayüz dizisi oluşturma ve polimorfizm uygulama
        ISesCikarabilir[] sesCikaranlar = new ISesCikarabilir[] { kedi, kopek };
        foreach (ISesCikarabilir sesCikaran in sesCikaranlar)
        {
            sesCikaran.SesCikar();
        }
    }
}
```

8

```
using System;
// Özel hata sınıfı
public class YetersizBakiyeException : Exception
{
    public YetersizBakiyeException(string message) : base(message)
    {
    }
}
// Banka hesabı sınıfı
public class BankaHesabi
{
    public double Bakiye { get; private set; }
    public BankaHesabi(double baslangicBakiyesi)
    {
        Bakiye = baslangicBakiyesi;
    }
    public void ParaYatir(double miktar)
    {
        if (miktar > 0)
        {
            Bakiye += miktar;
            Console.WriteLine($"{miktar:C} yatırıldı. Yeni bakiye: {Bakiye:C}");
        }
        else
        {
            Console.WriteLine("Geçersiz miktar. Pozitif bir değer girin.");
        }
    }
    public void ParaCek(double miktar)
```

```
    {
        if (miktar > 0 && miktar <= Bakiye)
        {
            Bakiye -= miktar;
            Console.WriteLine($"{miktar:C} çekildi. Yeni bakiye: {Bakiye:C}");
        }
        else if (miktar <= 0)
        {
            Console.WriteLine("Geçersiz miktar. Pozitif bir değer girin.");
        }
        else
        {
            throw new YetersizBakiyeException("Yetersiz bakiye. Para çekme işlemi
başarısız.");
        }
    }
}
public class Program
{
    public static void Main(string[] args)
    {
        BankaHesabi hesap = new BankaHesabi(1000);
        try
        {
            hesap.ParaCek(1500); // Yetersiz bakiye hatası fırlatacak
        }
        catch (YetersizBakiyeException ex)
        {
            Console.WriteLine($"Hata: {ex.Message}");
        }
        hesap.ParaYatir(500);
        hesap.ParaCek(200);
        Console.WriteLine($"Güncel bakiye: {hesap.Bakiye:C}");
    }
}
```

9

```
using System;
public class VeritabaniBaglantisi
{
    // Tek örnek (instance) değişkeni
    private static VeritabaniBaglantisi instance;
    // Özel yapıcı metot (private constructor)
    private VeritabaniBaglantisi()
    {
        // Veritabanı bağlantı işlemleri burada yapılır
        Console.WriteLine("Veritabanı bağlantısı kuruldu.");
    }
```

```
    // Tek örneği döndüren metot
    public static VeritabaniBaglantisi Instance
    {
        get
        {
            if (instance == null)
            {
                instance = new VeritabaniBaglantisi();
            }
            return instance;
        }
    }
    // Veritabanı işlemleri için örnek metot
    public void VeriEkle(string veri)
    {
        Console.WriteLine($"Veritabanına veri eklendi: {veri}");
    }
}
public class Program
{
    public static void Main(string[] args)
    {
        // Tek örneğe erişim
        VeritabaniBaglantisi baglanti1 = VeritabaniBaglantisi.Instance;
        VeritabaniBaglantisi baglanti2 = VeritabaniBaglantisi.Instance;
        // Aynı örneğe erişildiğini kontrol etme
        Console.WriteLine($"baglanti1 == baglanti2: {baglanti1 == baglanti2}"); //
Çıktı: baglanti1 == baglanti2: True
        // Veritabanı işlemleri
        baglanti1.VeriEkle("Örnek veri");
        baglanti2.VeriEkle("Başka bir veri");
    }
}
```

10

```
using System;
// Hayvan arayüzü
public interface IHayvan
{
    void SesCikar();
}
// Kuş sınıfı
public class Kus : IHayvan
{
    public void SesCikar()
    {
        Console.WriteLine("Cik cik!");
    }
```

```
}
// Balık sınıfı
public class Balik : IHayvan
{
    public void SesCikar()
    {
        Console.WriteLine("Glup glup!");
    }
}
// Hayvan fabrikası
public class HayvanFabrikasi
{
    public IHayvan HayvanOlustur(string tur)
    {
        if (tur.ToLower() == "kus")
        {
            return new Kus();
        }
        else if (tur.ToLower() == "balik")
        {
            return new Balik();
        }
        else
        {
            return null;
        }
    }
}
public class Program
{
    public static void Main(string[] args)
    {
        // Hayvan fabrikası nesnesi oluşturma
        HayvanFabrikasi fabrika = new HayvanFabrikasi();
        // Kuş nesnesi oluşturma
        IHayvan kus = fabrika.HayvanOlustur("kus");
        kus.SesCikar(); // Çıktı: Cik cik!
        // Balık nesnesi oluşturma
        IHayvan balik = fabrika.HayvanOlustur("balik");
        balik.SesCikar(); // Çıktı: Glup glup!
        // Geçersiz tür
        IHayvan gecersiz = fabrika.HayvanOlustur("kedi");
        if (gecersiz == null)
        {
            Console.WriteLine("Geçersiz hayvan türü.");}}}
```

Tablo 7. NDP Copilot çözümleri

NO	Copilot

1	using System;
	class Kisi
	{
	public string Ad { get; set; }
	public int Yas { get; set; }
	public void BilgileriYazdir()
	{
	Console.WriteLine($"Ad: {Ad}, Yaş: {Yas}");
	}
	}
	class Program
	{
	static void Main()
	{
	Kisi kisi = new Kisi { Ad = "Halil", Yas = 25 };
	kisi.BilgileriYazdir();
	}
	}

2	using System;
	class BankAccount
	{
	private decimal bakiye;
	public BankAccount(decimal baslangicBakiyesi)
	{
	bakiye = baslangicBakiyesi;
	}
	public void ParaYatir(decimal miktar)
	{
	bakiye += miktar;
	Console.WriteLine($"Yeni bakiye: {bakiye} TL");
	}
	public void ParaCek(decimal miktar)
	{
	if (miktar > bakiye)
	Console.WriteLine("Yetersiz bakiye!");
	else
	{
	bakiye -= miktar;
	Console.WriteLine($"Yeni bakiye: {bakiye} TL");
	}
	}
	}
	class Program

```
{
    static void Main()
    {
        BankAccount hesap = new BankAccount(1000);
        hesap.ParaYatir(500);
        hesap.ParaCek(300);
    }
}
```

3

```
using System;
class Arac
{
    public virtual void Hizlan() => Console.WriteLine("Araç hızlanıyor...");
}
class Bisiklet : Arac
{
    public override void Hizlan() => Console.WriteLine("Bisiklet pedal çevirerek
hızlanıyor...");
}
class Araba : Arac
{
    public override void Hizlan() => Console.WriteLine("Araba motor gücüyle
hızlanıyor...");
}
class Program
{
    static void Main()
    {
        Arac bisiklet = new Bisiklet();
        Arac araba = new Araba();
        bisiklet.Hizlan();
        araba.Hizlan();
    }
}
```

4

```
using System;
abstract class Sekil
{
    public abstract double AlanHesapla();
}
class Daire : Sekil
{
    public double YariCap { get; set; }
    public override double AlanHesapla() => Math.PI * YariCap * YariCap;
}
class Kare : Sekil
{
    public double Kenar { get; set; }
```

```
    public override double AlanHesapla() => Kenar * Kenar;
}
class Program
{
    static void Main()
    {
        Sekil daire = new Daire { YariCap = 5 };
        Sekil kare = new Kare { Kenar = 4 };
        Console.WriteLine($"Daire Alanı: {daire.AlanHesapla()}");
        Console.WriteLine($"Kare Alanı: {kare.AlanHesapla()}");
    }
}
```

5

```
using System;
abstract class Calisan
{
    public abstract double MaasHesapla();
}
class Memur : Calisan
{
    public override double MaasHesapla() => 5000;
}
class Yonetici : Calisan
{
    public override double MaasHesapla() => 10000;
}
class Program
{
    static void Main()
    {
        Calisan memur = new Memur();
        Calisan yonetici = new Yonetici();
        Console.WriteLine($"Memur maaşı: {memur.MaasHesapla()} TL");
        Console.WriteLine($"Yönetici maaşı: {yonetici.MaasHesapla()} TL");
    }
}
```

6

```
using System;
using System.Collections.Generic;
class Kitap
{
    public string Ad { get; set; }
}
class Kutuphane
{
    private List<Kitap> kitaplar = new List<Kitap>();
    public void KitapEkle(Kitap kitap) => kitaplar.Add(kitap);
    public void KitaplariListele()
```

```
    {
        foreach (var kitap in kitaplar) Console.WriteLine($"Kitap: {kitap.Ad}");
    }
}
class Program
{
    static void Main()
    {
        Kutuphane kutuphane = new Kutuphane();
        kutuphane.KitapEkle(new Kitap { Ad = "C# Programlama" });
        kutuphane.KitaplariListele();
    }
}
```

7

```
using System;
interface ISesCikarabilir
{
    void SesCikar();
}
class Kedi : ISesCikarabilir
{
    public void SesCikar() => Console.WriteLine("Miyav!");
}
class Kopek : ISesCikarabilir
{
    public void SesCikar() => Console.WriteLine("Hav hav!");
}
class Program
{
    static void Main()
    {
        ISesCikarabilir kedi = new Kedi();
        ISesCikarabilir kopek = new Kopek();
        kedi.SesCikar();
        kopek.SesCikar();
    }
}
```

8

```
using System;
class BankAccount
{
    private decimal bakiye;
    public BankAccount(decimal baslangicBakiyesi) => bakiye = baslangicBakiyesi;
    public void ParaCek(decimal miktar)
    {
        if (miktar > bakiye)
            throw new InvalidOperationException("Yetersiz bakiye!");
        bakiye -= miktar;
```

```
    }
}
class Program
{
    static void Main()
    {
        try
        {
            BankAccount hesap = new BankAccount(500);
            hesap.ParaCek(600);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
}
```

9

```
using System;
class VeritabaniBaglantisi
{
    private static VeritabaniBaglantisi _instance;
    private VeritabaniBaglantisi() { }
    public static VeritabaniBaglantisi GetInstance()
    {
        return _instance ??= new VeritabaniBaglantisi();
    }
}
class Program
{
    static void Main()
    {
        var baglanti1 = VeritabaniBaglantisi.GetInstance();
        var baglanti2 = VeritabaniBaglantisi.GetInstance();
        Console.WriteLine(baglanti1 == baglanti2); // true
    }
}
```

10

```
using System;
abstract class Hayvan
{
    public abstract void SesCikar();
}
class Kus : Hayvan
{
    public override void SesCikar() => Console.WriteLine("Cik cik!");
}
class Balik : Hayvan
```

```
{
    public override void SesCikar() => Console.WriteLine("Balık ses çıkarmaz!");
}
class HayvanFabrikasi
{
    public static Hayvan HayvanOlustur(string tur)
    {
        return tur.ToLower() switch
        {
            "kus" => new Kus(),
            "balik" => new Balik(),
            _ => throw new ArgumentException("Geçersiz hayvan türü!")
        };
    }
}
class Program
{
    static void Main()
    {
        Console.Write("Kuş veya Balık oluştur (kus/balik): ");
        string secim = Console.ReadLine();
        try
        {
            Hayvan hayvan = HayvanFabrikasi.HayvanOlustur(secim);
            hayvan.SesCikar();
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
}
```

Tablo 8. NDP Claude çözümleri

NO	Claude

1	`using System;`
	`class Kisi`
	`{`
	`  public string Ad;`
	`  public int Yas;`
	`  public void BilgileriYazdir()`
	`  {`
	`  Console.WriteLine($"Ad: {Ad}, Yaş: {Yas}");`
	`  }`
	`}`
	`class Program`
	`{`
	`  static void Main()`
	`  {`
	`  Kisi kisi1 = new Kisi();`
	`  kisi1.Ad = "Ahmet";`
	`  kisi1.Yas = 25;`
	`  Kisi kisi2 = new Kisi();`
	`  kisi2.Ad = "Ayşe";`
	`  kisi2.Yas = 30;`
	`  Console.WriteLine("Kişi Bilgileri:");`
	`  kisi1.BilgileriYazdir();`
	`  kisi2.BilgileriYazdir();`
	`  Console.ReadKey();`
	`  }`
	`}`

2	`using System;`
	`class BankAccount`
	`{`
	`  private double bakiye;`
	`  public void ParaYatir(double miktar)`
	`  {`
	`  if (miktar > 0)`
	`  {`
	`  bakiye += miktar;`
	`  Console.WriteLine($"{miktar} TL yatırıldı. Yeni bakiye: {bakiye} TL");`
	`  }`
	`  else`
	`  {`
	`  Console.WriteLine("Geçersiz miktar!");`
	`  }`
	`  }`

```
    public void ParaCek(double miktar)
    {
        if (miktar > 0 && miktar <= bakiye)
        {
            bakiye -= miktar;
            Console.WriteLine($"{miktar} TL çekildi. Kalan bakiye: {bakiye} TL");
        }
        else
        {
            Console.WriteLine("Yetersiz bakiye veya geçersiz miktar!");
        }
    }
    public double BakiyeGoruntule()
    {
        return bakiye;
    }
}
class Program
{
    static void Main()
    {
        BankAccount hesap = new BankAccount();

        Console.WriteLine($"Başlangıç bakiyesi: {hesap.BakiyeGoruntule()} TL");

        hesap.ParaYatir(1000);
        hesap.ParaYatir(500);
        hesap.ParaCek(300);
        hesap.ParaCek(2000);

        Console.WriteLine($"Son bakiye: {hesap.BakiyeGoruntule()} TL");
        Console.ReadKey();
    }
}
```

3

```
using System;
class Arac
{
    public string Marka;
    public int Hiz;

    public virtual void Hizlan()
    {
        Hiz += 10;
        Console.WriteLine($"Araç hızlandı. Hız: {Hiz} km/h");
    }
```

```
}
class Bisiklet : Arac
{
    public override void Hizlan()
    {
        Hiz += 5;
        Console.WriteLine($"Bisiklet pedal çevrilerek hızlandı. Hız: {Hiz} km/h");
    }
}
class Araba : Arac
{
    public override void Hizlan()
    {
        Hiz += 20;
        Console.WriteLine($"Araba gaza basarak hızlandı. Hız: {Hiz} km/h");
    }
}
class Program
{
    static void Main()
    {
        Bisiklet bisiklet = new Bisiklet();
        bisiklet.Marka = "Trek";

        Araba araba = new Araba();
        araba.Marka = "Toyota";

        Console.WriteLine("Bisiklet:");
        bisiklet.Hizlan();
        bisiklet.Hizlan();

        Console.WriteLine("\nAraba:");
        araba.Hizlan();
        araba.Hizlan();

        Console.ReadKey();
    }
}
```

4

```
using System;
class Sekil
{
    public virtual double AlanHesapla()
    {
        return 0;
    }
```

```
}
class Daire : Sekil
{
    public double Yaricap;
    public override double AlanHesapla()
    {
        return Math.PI * Yaricap * Yaricap;
    }
}
class Kare : Sekil
{
    public double Kenar;

    public override double AlanHesapla()
    {
        return Kenar * Kenar;
    }
}
class Program
{
    static void Main()
    {
        Daire daire = new Daire();
        daire.Yaricap = 5;

        Kare kare = new Kare();
        kare.Kenar = 4;

        Console.WriteLine($"Daire alanı (r=5): {daire.AlanHesapla():F2}");
        Console.WriteLine($"Kare alanı (kenar=4): {kare.AlanHesapla():F2}");

        // Polimorfizm örneği
        Sekil[] sekiller = {daire, kare};

        Console.WriteLine("\nPolimorfizm ile:");
        for (int i = 0; i < sekiller.Length; i++)
        {
            Console.WriteLine($"Şekil {i+1} alanı:
{sekiller[i].AlanHesapla():F2}");
        }
        Console.ReadKey();
    }
}
```

5

```
using System;
abstract class Calisan
{
```

```
    public string Ad;
    public int Id;

    public abstract double MaasHesapla();

    public void BilgiGoster()
    {
        Console.WriteLine($"Ad: {Ad}, ID: {Id}, Maaş: {MaasHesapla():F2} TL");
    }
}
class Memur : Calisan
{
    public double TemelMaas;

    public override double MaasHesapla()
    {
        return TemelMaas;
    }
}
class Yonetici : Calisan
{
    public double TemelMaas;
    public double YoneticiEki;

    public override double MaasHesapla()
    {
        return TemelMaas + YoneticiEki;
    }
}
class Program
{
    static void Main()
    {
        Memur memur = new Memur();
        memur.Ad = "Ahmet";
        memur.Id = 1001;
        memur.TemelMaas = 15000;

        Yonetici yonetici = new Yonetici();
        yonetici.Ad = "Ayşe";
        yonetici.Id = 2001;
        yonetici.TemelMaas = 20000;
        yonetici.YoneticiEki = 5000;

        Console.WriteLine("Çalışan Bilgileri:");
```

```
        memur.BilgiGoster();
        yonetici.BilgiGoster();
        Console.ReadKey();
    }
}
```

6

```
using System;
using System.Collections.Generic;
class Kitap
{
    public string Ad;
    public string Yazar;
    public int Sayfa;
    public void BilgiGoster()
    {
        Console.WriteLine($"Kitap: {Ad}, Yazar: {Yazar}, Sayfa: {Sayfa}");
    }
}
class Kutuphane
{
    private List<Kitap> kitaplar;

    public Kutuphane()
    {
        kitaplar = new List<Kitap>();
    }

    public void KitapEkle(Kitap kitap)
    {
        kitaplar.Add(kitap);
        Console.WriteLine($"'{kitap.Ad}' kitabı eklendi.");
    }
    public void KitapSil(string kitapAdi)
    {
        for (int i = 0; i < kitaplar.Count; i++)
        {
            if (kitaplar[i].Ad == kitapAdi)
            {
                kitaplar.RemoveAt(i);
                Console.WriteLine($"'{kitapAdi}' kitabı silindi.");
                return;
            }
        }
        Console.WriteLine($"'{kitapAdi}' kitabı bulunamadı.");
    }
    public void KitaplariListele()
```

```
    {
        Console.WriteLine($"\nKütüphanede {kitaplar.Count} kitap var:");
        for (int i = 0; i < kitaplar.Count; i++)
        {
            Console.Write($"{i+1}. ");
            kitaplar[i].BilgiGoster();
        }
    }
}
class Program
{
    static void Main()
    {
        Kutuphane kutuphane = new Kutuphane();

        Kitap kitap1 = new Kitap();
        kitap1.Ad = "Suç ve Ceza";
        kitap1.Yazar = "Dostoyevski";
        kitap1.Sayfa = 500;

        Kitap kitap2 = new Kitap();
        kitap2.Ad = "1984";
        kitap2.Yazar = "George Orwell";
        kitap2.Sayfa = 350;

        kutuphane.KitapEkle(kitap1);
        kutuphane.KitapEkle(kitap2);
        kutuphane.KitaplariListele();

        kutuphane.KitapSil("1984");
        kutuphane.KitaplariListele();

        Console.ReadKey();
    }}
```

7

```
using System;
interface SesCikarabilir
{
    void SesCikar();
}
class Kedi : SesCikarabilir
{
    public string Ad;

    public void SesCikar()
    {
```

```
        Console.WriteLine($"{Ad} miyavlıyor: Miyav miyav!");
    }
}
class Kopek : SesCikarabilir
{
    public string Ad;

    public void SesCikar()
    {
        Console.WriteLine($"{Ad} havlıyor: Hav hav!");
    }
}
class Program
{
    static void Main()
    {
        Kedi kedi = new Kedi();
        kedi.Ad = "Pamuk";

        Kopek kopek = new Kopek();
        kopek.Ad = "Karabaş";

        kedi.SesCikar();
        kopek.SesCikar();

        // Interface referansı ile
        Console.WriteLine("\nInterface referansı ile:");
        SesCikarabilir[] hayvanlar = {kedi, kopek};

        for (int i = 0; i < hayvanlar.Length; i++)
        {
            hayvanlar[i].SesCikar();
        }

        Console.ReadKey();
    }
}
```

8

```
using System;
class YetersizBakiyeException : Exception
{
    public YetersizBakiyeException(string mesaj) : base(mesaj)
    {
    }
}
class BankaHesabi
```

```
{
    private double bakiye;

    public BankaHesabi(double baslangicBakiye)
    {
        bakiye = baslangicBakiye;
    }

    public void ParaYatir(double miktar)
    {
        if (miktar <= 0)
            throw new ArgumentException("Yatırılacak miktar pozitif olmalı!");

        bakiye += miktar;
        Console.WriteLine($"{miktar} TL yatırıldı. Yeni bakiye: {bakiye} TL");
    }

    public void ParaCek(double miktar)
    {
        if (miktar <= 0)
            throw new ArgumentException("Çekilecek miktar pozitif olmalı!");

        if (miktar > bakiye)
            throw new YetersizBakiyeException($"Yetersiz bakiye! Mevcut bakiye:
{bakiye} TL, Çekilmek istenen: {miktar} TL");

        bakiye -= miktar;
        Console.WriteLine($"{miktar} TL çekildi. Kalan bakiye: {bakiye} TL");
    }

    public double BakiyeGoruntule()
    {
        return bakiye;
    }
}
class Program
{
    static void Main()
    {
        BankaHesabi hesap = new BankaHesabi(1000);

        try
        {
            Console.WriteLine($"Başlangıç bakiyesi: {hesap.BakiyeGoruntule()} TL");
```

```
            hesap.ParaYatir(500);
            hesap.ParaCek(800);
            hesap.ParaCek(1000); // Hata fırlatacak
        }
        catch (YetersizBakiyeException ex)
        {
            Console.WriteLine($"Hata: {ex.Message}");
        }
        catch (ArgumentException ex)
        {
            Console.WriteLine($"Geçersiz işlem: {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Beklenmeyen hata: {ex.Message}");
        }

        Console.WriteLine($"Son bakiye: {hesap.BakiyeGoruntule()} TL");
        Console.ReadKey();
    }
}
```

9

```
using System;
public class VeritabaniBaglantisi
{
    // Tek örneği tutacak static field
    private static VeritabaniBaglantisi _instance = null;

    // Private constructor
    private VeritabaniBaglantisi()
    {
        Console.WriteLine("VeritabaniBaglantisi oluşturuldu.");
    }

    // Singleton instance'ını döndürür
    public static VeritabaniBaglantisi Instance
    {
        get
        {
            if (_instance == null)
                _instance = new VeritabaniBaglantisi();

            return _instance;
        }
    }
```

```
    public void Baglan()
    {
        Console.WriteLine("Veritabanına bağlanıldı.");
    }
}
// Kullanım
class Program
{
    static void Main()
    {
        VeritabaniBaglantisi db1 = VeritabaniBaglantisi.Instance;
        VeritabaniBaglantisi db2 = VeritabaniBaglantisi.Instance;

        Console.WriteLine($"Aynı instance mı? {db1 == db2}");

        db1.Baglan();
    }
}
```

10

```
using System;
// Base sınıf
public abstract class Hayvan
{
    public abstract void SesCikar();
    public abstract void Hareket();
}
// Kus sınıfı
public class Kus : Hayvan
{
    public override void SesCikar()
    {
        Console.WriteLine("Kuş cıvıldıyor: Cik cik!");
    }

    public override void Hareket()
    {
        Console.WriteLine("Kuş uçuyor.");
    }
}
// Balik sınıfı
public class Balik : Hayvan
{
    public override void SesCikar()
    {
        Console.WriteLine("Balık sessizce yaşıyor...");
    }
```

```
    public override void Hareket()
    {
        Console.WriteLine("Balık yüzüyor.");
    }
}
// Factory sınıfı
public class HayvanFabrikasi
{
    public static Hayvan HayvanOlustur(string hayvanTuru)
    {
        switch (hayvanTuru.ToLower())
        {
            case "kus":
                return new Kus();
            case "balik":
                return new Balik();
            default:
                throw new ArgumentException($"Bilinmeyen hayvan türü:
{hayvanTuru}");
        }
    }
}
// Kullanım
class Program
{
    static void Main()
    {
        Console.WriteLine("=== Factory Tasarım Deseni ===\n");

        // Kuş oluştur
        Hayvan kus = HayvanFabrikasi.HayvanOlustur("kus");
        Console.WriteLine("Kuş oluşturuldu:");
        kus.SesCikar();
        kus.Hareket();

        Console.WriteLine();

        // Balık oluştur
        Hayvan balik = HayvanFabrikasi.HayvanOlustur("balik");
        Console.WriteLine("Balık oluşturuldu:");
        balik.SesCikar();
        balik.Hareket();

        Console.WriteLine();
```

```
        // Hatalı parametre
        try
        {
            Hayvan bilinmeyen = HayvanFabrikasi.HayvanOlustur("kedi");
        }
        catch (ArgumentException ex)
        {
            Console.WriteLine($"Hata: {ex.Message}");
        }
    }
}
```