

(7)
C# .NET
PROGRAMLAMAYA GİRİŞ

7- PROGRAMLAMAYA GİRİŞ

Neden C# .NET programlama dili ?

C, Java ailesinden bir programlama dilidir

Platformdan (işletim sistemi-Makine) bağımsız yazılım geliştirme

Nesne tabanlı (OOP – Object Oriented Programming)

Console, GUI, Web (server side), uygulamaları oluşturabilirsiniz

Kodları C ve Java ya çok benzediğinden bunlardan birini bilen çok kolay adapte olur. Yada da C# öğrenen bunları çok kolay öğrenir

En popüler dillerden bir tanesidir

Yeteri kadar kaynak bulabilirsiniz (kitap, ders notu, program örneği vb.)

7- PROGRAMLAMAYA GİRİŞ

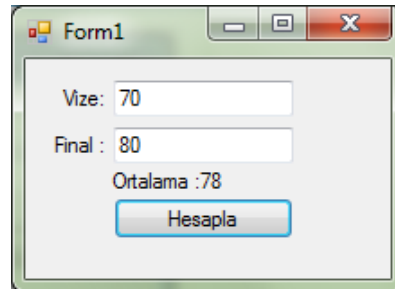
C# .NET programlama dili

C# .NET

Console
Uygulamalar

```
Vize Notunu Giriniz:70  
Final Notunu Giriniz:80  
Ortalama: 78
```

Visual Application
Grafik-Arabirim Uygulamalar
Graphics User Interface (GUI)



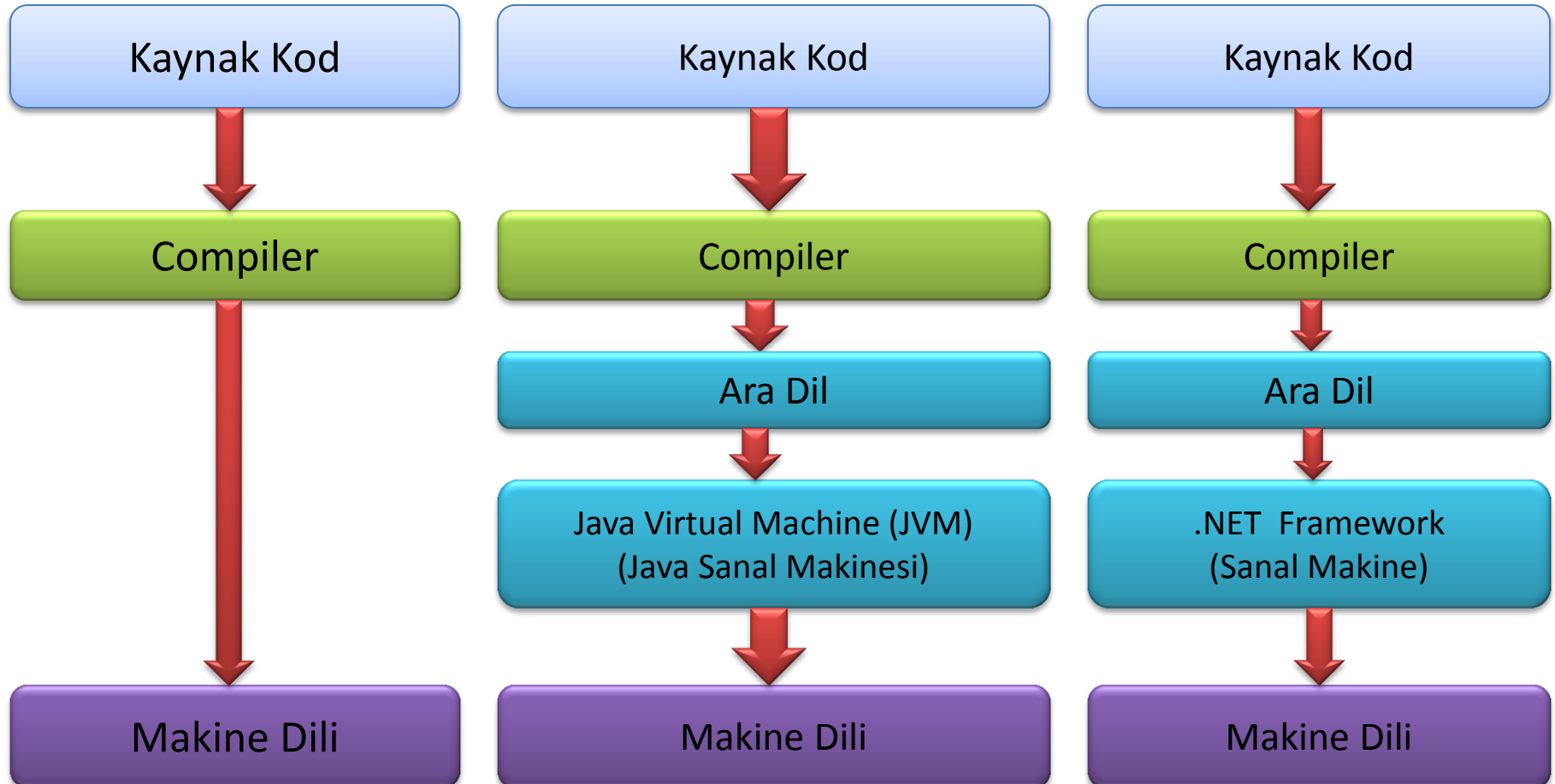
Web
(C# .net aspx)

Smart Device
(Mobil telefon, PDA)



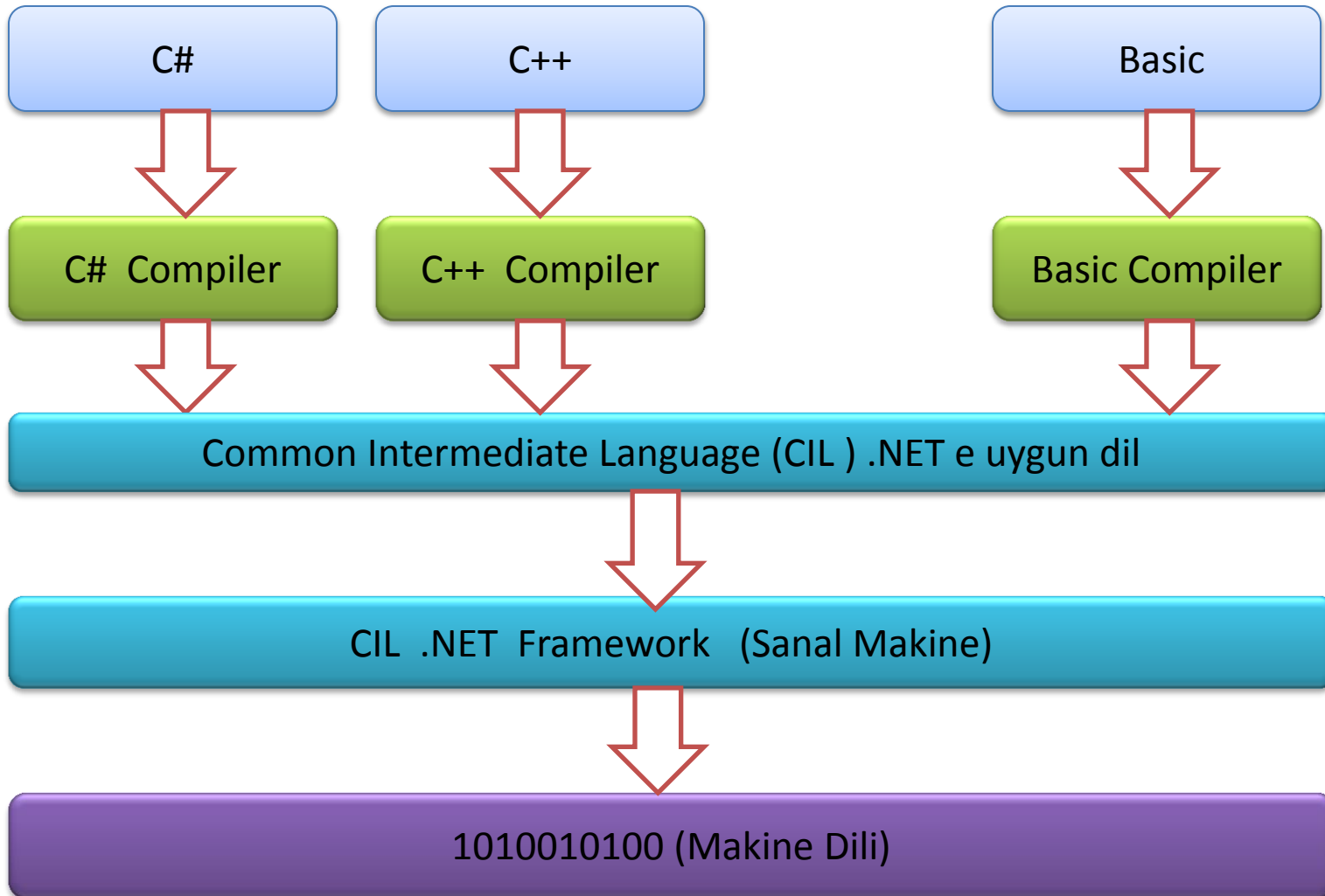
7- PROGRAMLAMAYA GİRİŞ

Platformdan (işlemci- işletim sistemi) bağımsız programlar



7- PROGRAMLAMAYA GİRİŞ

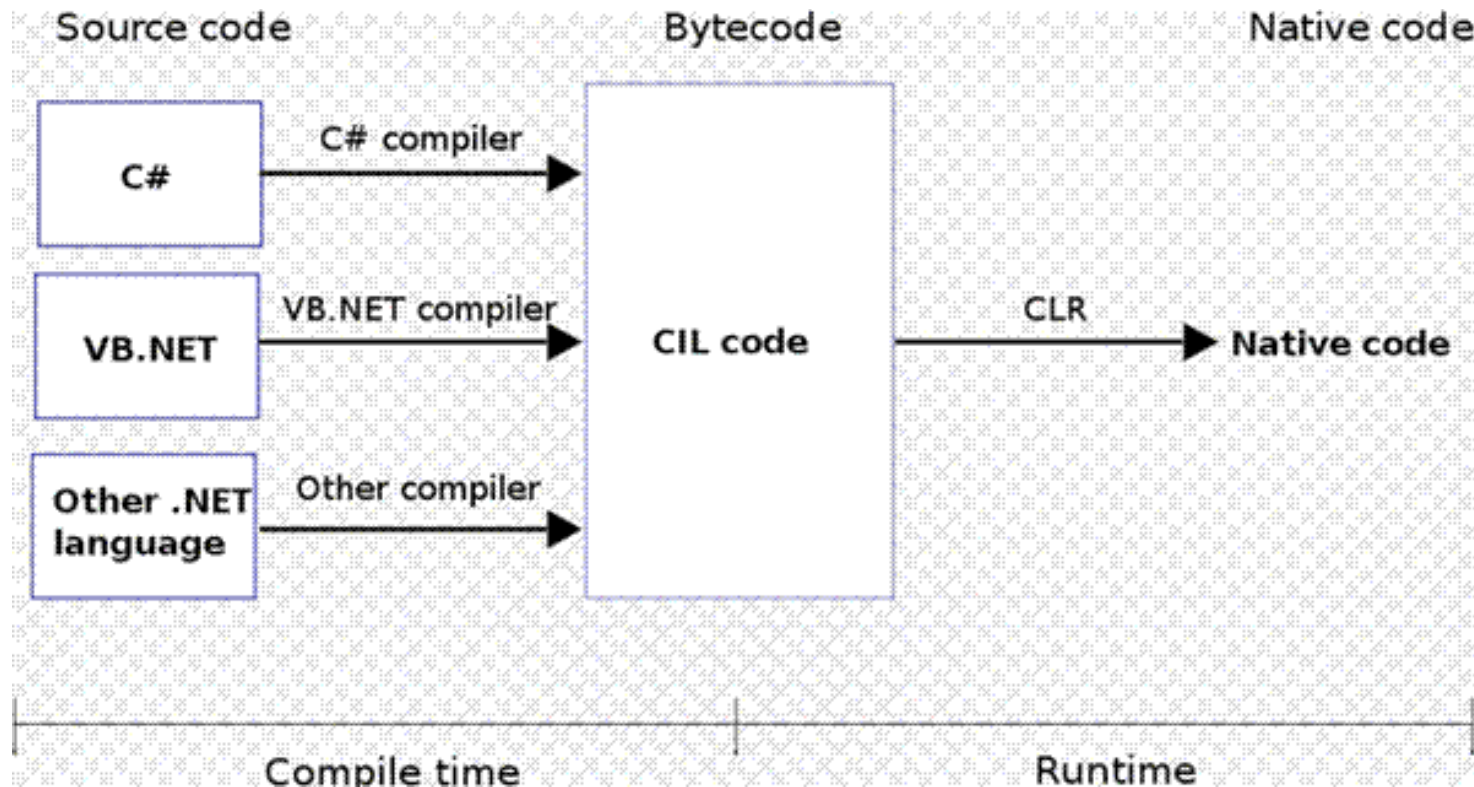
.NET Mimarisi



.Net Uygulamasının Çalışma Süreci:

Compile Time: Yazmış olduğunuz uygulamanın CIL(Common Intermediate Language) çevrildiği zaman sürecidir. Her hangi bir dil de yazılmış .Net uygulaması, her dile ait derleyiciler ile ortak bir dile CIL(Bytecode) çevrilir.

Run Time: Uygulamanın çalıştığı süreçtir, Compile Time da oluşturulan ByteCode'lar CLR(Common Language Runtime) ile satır satır yorumlanarak (Interpreter) makine diline çevrilir ve uygulamanın çalışması sağlanır



C# .net Programlama dilinin genel yapısı

```
using System;
```

```
static void Main(string[] args)
```

```
{
```

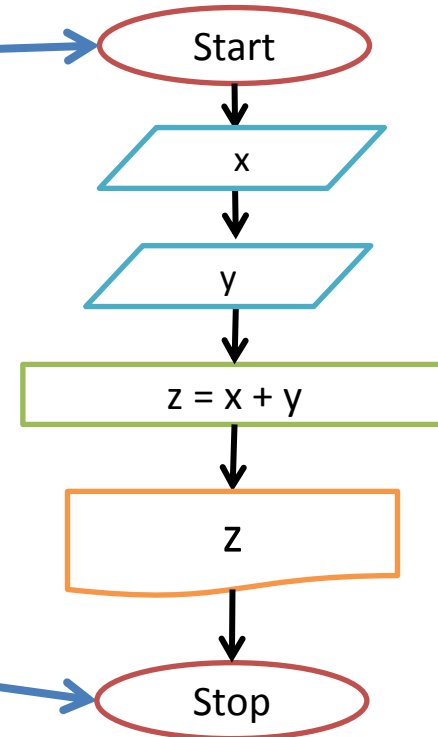
```
    int x = int.Parse(Console.ReadLine() );
```

```
    int y = int.Parse(Console.ReadLine());
```

```
    int z = x + y;
```

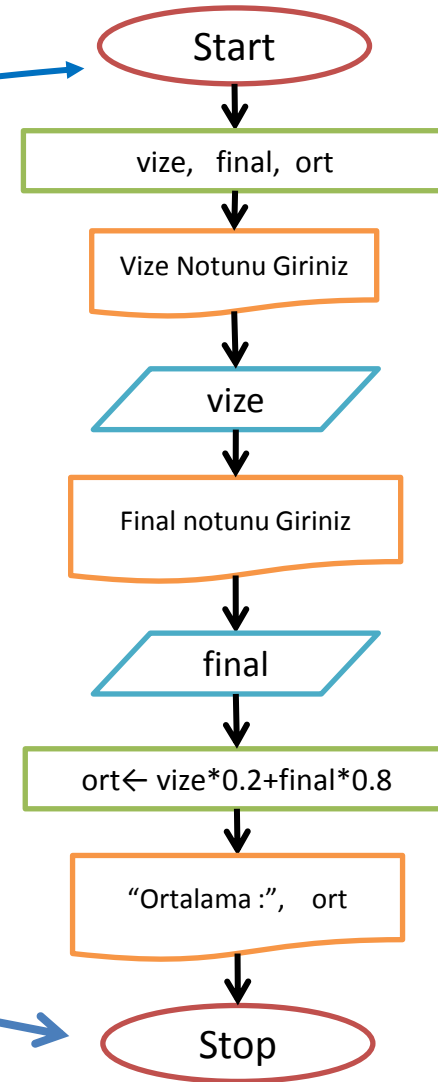
```
    Console.Write(z);
```

```
}
```



C# .net Programlama dilinin genel yapısı

```
1 using System;
2
3 namespace ornek1
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             int vize;
10            int final;
11            double ort;
12            Console.Write("Vize Notunu Giriniz:");
13            vize = int.Parse(Console.ReadLine());
14
15            Console.Write("Final Notunu Giriniz:");
16            final = int.Parse(Console.ReadLine());
17            ort = vize * 0.2 + final * 0.8;
18            Console.Write("Ortalama: {0} ", ort);
19            Console.ReadKey();
20        }
21    }
22 }
23
```



7- PROGRAMLAMAYA GİRİŞ

7.4 **Değişken** : program içerisinde değeri dinamik olarak değişebilen yapılardır. Belirli bir tipi vardır. Bu yüzden tanımlandığı tipteki verileri kabul eder.

Sabit : program içerisinde değeri değişmeyen yapılardır. Belirli bir tipi vardır. Bu yüzden tanımlandığı tipteki verileri kabul eder.

Değişken ve Sabit ismi vermek kuralları :

- 1- İngiliz alfabesindeki 26 harf, (0-9) rakamlar ve “_” (alt tire) kullanılmalıdır.
- 2- Dilin kendi komutlarını (reserved word) değişken ismi olarak kullanılamaz.
- 3- ilk karakteri rakam olamaz
- 4- Harf, alt tire, rakamlar haricinde başka karakter kullanamaz.
- 5- en az 1 en çok 16383 karakter uzunluğunda olabilir

7- PROGRAMLAMAYA GİRİŞ

DOĞRU

adi_soyadi
vize1
_final
devam
x
_
anastasmumsatsana

YANLIŞ

adi soyadi
1vize
final-
class
x+y
@edu
maku@gamil.com

NOT: Değişken ismi verirken içinde tutacağı veriyi çağrıştıracak isim verilmesi önerilir.

Örneğin: adi, soyadi, vize, maas, ortalama, toplam

7- PROGRAMLAMAYA GİRİŞ

7.5- Değişken ismi verirken **Macar Metodu** (Hungarian Notation-Medhodology) :

Değişken ismi, **değişkenin tipini** ve **içerisinde barındıracak bilgiyi** çağrıştıracak şekilde verilmesidir.

Örneğin; değişkende vize bilgisi tutulacaksa; vize değeri de 0 ile 100 arasında bir sayı olabileceği bellidir. O halde vize bilgisi için uygun değişken verme kuralı şöyle olmalıdır:

Tutulacak veri muhtevası : vize bilgileri

Tutulacak veri değeri : 0- 100 arası tamsayı bir değer

intVize şeklinde verilmesi uygundur.

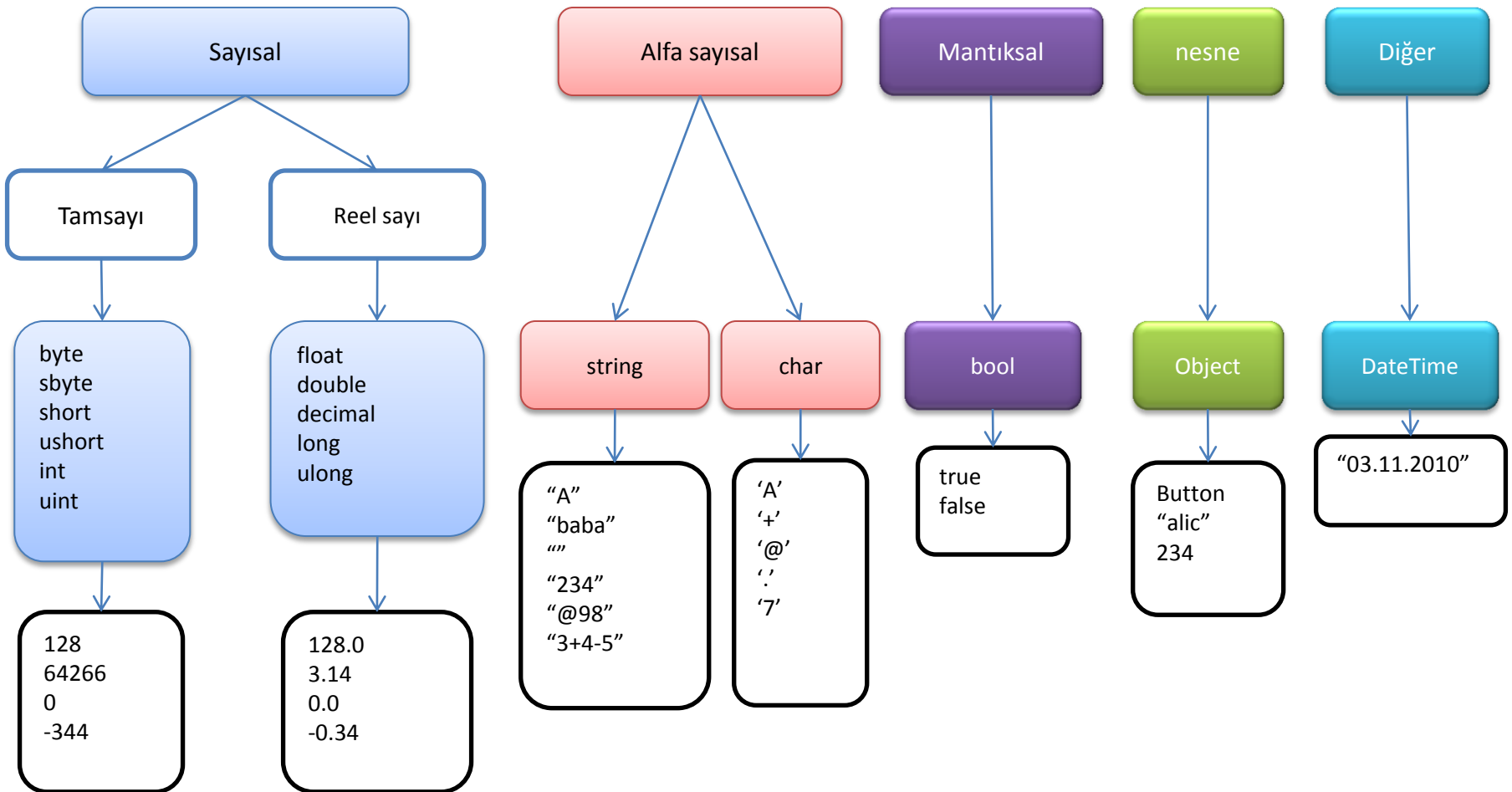
vize	yerine	int_vize	veya	intVize	veya	Vize_int
cevap	yerine	char_cevap	veya	charCevap	veya	Cevap_char
devam	yerine	bool_devam	veya	boolDevam	veya	Devam_bool
adi	yerine	string_adi	veya	stringAdi	veya	Adi_string

C# Reserved Words

abstract	as	base	bool	break	byte
case	catch	char	checked	class	const
continue	decimal	default	delegate	do	double
else	enum	event	explicit	extern	false
finally	fixed	float	for	foreach	goto
if	implicit	in	int	interface	internal
is	lock	long	namespace	new	null
object	operator	out	override	params	private
protected	public	readonly	ref	return	sbyte
sealed	short	sizeof	stackalloc	static	string
struct	switch	this	throw	true	try
typeof	uint	ulong	unchecked	unsafe	ushort
using	virtual	void	volatile	while	by
descending	from	group	into	orderby	select
var	where	yield	TRUE	FALSE	

7- PROGRAMLAMAYA GİRİŞ

7.6- C#. Net de kullanılan değişken ve sabit tipleri :

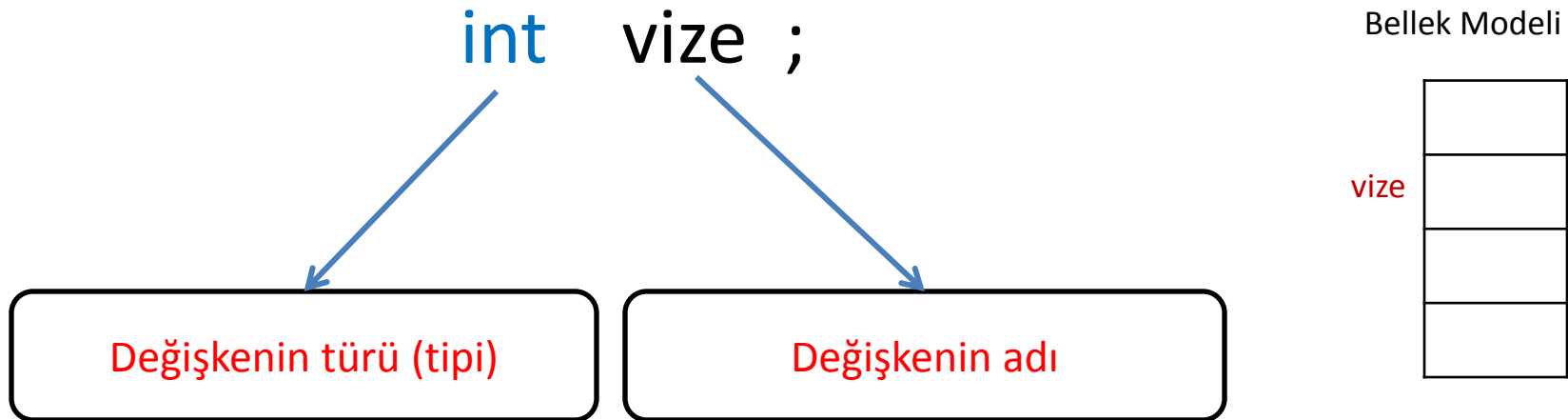


7.6- C#. Net deki **değişken** tipleri :

Veri tipi	Byte uzunluğu	.NET veri tipi	Sınırları(alabileceği değerler)
char	1	Char	Herhangi bir ASCII karakterler
string	-	string	Karakter veya karakter topluluğu (alfabetik ifadeler)
bool	1	Boolean	true , false
byte	1	Byte	(0-255)
sbyte	1	Sbyte	(-128 ile +127))
short	1	Int16	(-32.768 ile +32.767)
ushort	2	UInt16	(0-65.535)
int	2	Int32	(-2.147.483.647 ile 2.147.483.647)
uint	4	UInt32	0 – 4.294.967.295
float	4	Single	+/-1,5 *10 ⁻⁴⁵ ile +/- 3.4 *10 ³⁸
double	8	Double	+/- 0,5 *10 ⁻³²⁴ ile +/- 1.7 *10 ³⁰⁸
decimal	8	Decimal	Virgülden sonra 28 basamak hassasiyetinde
long	8	Int64	-9.223.372.036.854.775.808 ile 9.223.372.036.854.775.807
ulong	8	UInt64	0- f f f f f f f f f f f f f f f f (0 - 2 ⁶⁴)

7- PROGRAMLAMAYA GİRİŞ

7.7- C# da değişken tanımlama işlemi:



```
int vize, final, odev, ortalama ;
```

```
int vize=0, final=0, ortalama =0 ;
```

NOT : Bir programda aynı isimde birden çok değişken veya sabit olamaz

```
int vize;  
byte vize; // HATALI daha önce vize adında değişken tanımlanmış
```

7- PROGRAMLAMAYA GİRİŞ

Değişken tanımlama ve bellek modeli

```
using System;

namespace ornek1
{
    class Program
    {
        static void Main (string [] args )
        {
            int     vize = 70 ;
            int     final = 80 ;
            double   ort = vize *0.2 + final*0.8 ;
            string   adi="Ali";
        }
    }
}
```

Değişkenlerin RAM daki durumu

vize	70
final	80
ort	78
adi	"Ali"

- Açılan küme parantezi ({) kadar, kapanan küme parantezi (}) olmalı
- Küme parantezleri al alta gelmeli.

7- PROGRAMLAMAYA GİRİŞ

7.8- C# da değişken tanımlama örnekleri:

```
bool cevap ;
```

```
float buce , maas, gelir, gider, blanco;
```

```
char devam='E' ;    string devam="E";
```

```
string adi, soyadi, yanit="E" ;
```

```
Object butce =null;
```

```
Random r = new Random();
```

```
DateTime tarih = DateTime.Now;
```

```
int sayi = 0xFF ; // 0xff hexadecimal sayıdır ve onluk karşılığı 255 tir.
```

7- PROGRAMLAMAYA GİRİŞ

7.8- C# da **SABİT** tanımlama ve örnekleri:

```
const float pisayisi = 3.14 ;
```

```
const string mesaj = " Devam Etmek İçin Bir Tuşa Basınız" ;
```

```
const char devam='E' ;
```

```
const string s= " Bugün hava " + (23+5) + " "derecedir" ; // Hatalı çünkü bir işlem var
```

```
const string adi; // hatalı çünkü sabit bir başlangıç değeri verilmeli
```

```
adi= "Ali" ; // sonradan değeri değiştirilemez
```

7- PROGRAMLAMAYA GİRİŞ

NOT : C# , C, C++ , Java gibi dillerde değişken ve fonksiyon (komut) tanımlamalarında **BÜYÜK**, **küçük** harf **AYIRIMI** vardır :

```
Console.Write (); // console.Write() ; veya Console.write () ; yanlış olacaktır
```

```
float butce, Butce ;
```

```
char devam='E', Devam='E' ;
```

```
string yanit="EVET", Yanit="EVET" ;
```

```
TextBox textBox1, textbox1 ;
```

```
random r; // YANLIŞTIR (Random olmalı)
```

```
datetime tarih; // YANLIŞTIR (DateTime olmalı)
```

7- PROGRAMLAMAYA GİRİŞ

7.9- Tip Dönüşümleri : veriler farklı tipteki değişkenlerden başka bir değişkene aktarılmak istenebilir. Bu durumda yeni gideceği değişkenin tipine uygun olarak tip dönüşümü yapılmalıdır.

```
string vize= "70" , final = " 90";
```

```
double ort = Convert.ToDouble (vize) * 0.2 + Convert.ToDouble ( final ) *0.8 ;
```

```
double ort = double.Parse(vize) * 0.2 + double.Parse ( final ) * 0.8;
```

```
int x = int.Parse("245");
```

```
int x = int.Parse("25+75"); // HATA
```

```
string notu = Convert.ToString( 65 ) ;
```

7- PROGRAMLAMAYA GİRİŞ

7.9- Tip Dönüşümleri : veriler farklı tipteki değişkenlerden başka bir değişkene aktarılmak istenebilir. Bu durumda yeni gideceği değişkenin tipine uygun olarak tip dönüşümü yapılmalıdır.

```
double ort = 70;
```

```
string sonuc = ort.ToString() ; // sonuc = Convert.ToString(ort);
```

```
string x = 657.ToString() ;
```

```
string y = (3.14).ToString() ;
```

7- PROGRAMLAMAYA GİRİŞ

```
string  sicaklik= "Bugün Hava Sıcaklığı" + 27.ToString() + " Derecedir" ;
```

```
string  sicaklik= "Bugün Hava Sıcaklığı" + 27 + " Derecedir" ;
```

27 rakamı otomatik olarak kendinden önceki tip olan **string**' e dönüştürülecektir.

```
string  sicaklik= "Sıcaklık " + 27 + 13 + " derecedir" ; // Sıcaklık 2713 dercedir
```

```
string  sicaklik= "Sıcaklık " + (27 + 13) + " derecedir" ; // Sıcaklık 40 dercedir
```

7- PROGRAMLAMAYA GİRİŞ

7.9- Taban Aritmetik' li Tip Dönüşümleri

```
string x = 182.ToString("X"); // B6
```

```
int y = int.Parse( x, System.Globalization.NumberStyles.HexNumber); // 182
```

```
Convert.ToString ( sayi , taban) ; // sayi ' yi, taban' a çevrilir
```

```
Console.Write ( Convert.ToString (7 , 2) ) ; // ( 0 1 1 1 )2
```

7- PROGRAMLAMAYA GİRİŞ

7.10- Paketleme (BOXING) :

```
int  ort = (int) 3.14; // 3 olur
```

```
int  n= 5/ 2 ;        // 2 olur
```

```
double  x= 5/ 2 ;     // 2 olur
```

```
double  x= 5.0/ 2 ; // 2.5 olur
```

```
double  x= 5/ 2.0 ; // 2.5 olur
```

```
Char  c = (Char) 65 ; // c' nin değeri A olur
```


7- PROGRAMLAMAYA GİRİŞ

7.11- Açıklama (Remark) Operatörü;

//
veya
/* */

Açıklama yamak için kullanılır. **Komut olarak işlem görmez.**

```
int x= 5 ; // x değişkenine 5 aktarılır  
// int y= 7 ;
```

```
/*  
int x = 5 ;  
int y = 5 +8 ;  
string s = "Büyük beyinler fikirlerle ilgilenir. ";  
*/
```

7- PROGRAMLAMAYA GİRİŞ

7.11- OPERATÖRLER:

Atama

= ata (x=2;)
+= topla ata
-=
*= çarp ata
/=
%=
>>=
<<=

Aritmetik

+
-
*
/
%

İlişkisel

>
>=
<
<=
==
!=

Mantıksal

&& and
|| OR
! Not

Bit

& ve
| veya
~ değil
^ xor
>> sağa kaydır
<< sola kaydır

Unary

++
--

7- PROGRAMLAMAYA GİRİŞ

```
int x = 5 ;  
x += 4 ; // x ' 4 ekle. Yani x = x + 4; tür.  
x %=2 ; // x ' deki değer mod 2 yapılır ve sonuç x ' e atanır. Yani x = x %2;  
x >>=1 ; // x ' deki değer in tüm bitleri bir kez sağa kaydırılıp sonuç x ' e atanır.
```

```
bool devam = true;  
string mail = "milkucar@gmail.com" ;  
char dogrumu = 'e' ;
```

```
if ( x>5 ) ..... ; // x 5 ' den büyük ise  
if ( x>5 && x<10 ) ..... ; // x 5 ' den VE 10 'dan küçük ise  
if ( x>5 || x<10 ) ..... ; // x 5 ' den VEYA 10 'dan küçük ise  
if ( !(x>5) ) ..... ; // x 5 ' den büyük DEĞİLSE  
if ( x != null ) ..... ; // x null (hiç) ' den FARKLI ise
```

```
int sayac = 0;  
sayac ++;  
++ sayac ;
```

```
int say = 0;  
say --;  
-- say ;
```

7- PROGRAMLAMAYA GİRİŞ

7.11- OPERATÖRLER:

```
int x= 5 + 8;
```

```
x += 6; // x = x + 6;
```

```
int y ;
```

```
y++ ; ++y ; // y = y + 1;
```

```
x++;
```

```
++x;
```

```
x =x + 1;
```

```
x += 1;
```

```
//Hepsi aynı işi yapar
```

```
int k=5;
```

```
int m=k++;
```

<u>k</u>	<u>m</u>
6	5

```
int k=5;
```

```
int m=++k;
```

<u>k</u>	<u>m</u>
6	6

7- PROGRAMLAMAYA GİRİŞ

OPERATÖRLER:

```
if( x> 5) Console.Write("A");
```

```
if( x> 5 && x<= 7) Console.Write("B");
```

```
if( x != 3) Console.Write("A");
```

```
if( harfnot=="DC" || harfnot=="DD")  
Console.Write("şartlı");
```

7- PROGRAMLAMAYA GİRİŞ

7.11- & (AND) bitisel operatörü doğruluk tablosu

AND (&)
Doğruluk tablosu

&	0	1
0	0	0
1	0	1

`int z = 5 & 3; // 1 olur`

5 -> 0 1 0 1
3 -> 0 0 1 1

0 0 0 1 -> (1)₁₀

Örn: x'in üçüncü bitinin 1 olup olmadığını test etmek

`if((x & 00100) > 0) ise x'in üçüncü biti 1 dir`

x -> b b b b b b
3 -> 0 0 0 1 0 0

y -> b 1 ise y 1 olur, b 0 ise y 0 olur

7- PROGRAMLAMAYA GİRİŞ

7.11- | (OR) bitsel operatörü doğruluk tablosu

OR (|) Doğruluk tablosu

	0	1
0	0	1
1	1	1

```
int x = 5 ;  
int y = 3 ;  
int z = 5 | 3; // 7 olur
```

```
5 -> 0 1 0 1  
3 -> 0 0 1 1  
-----  
      0 1 1 1 -> 7
```

7- PROGRAMLAMAYA GİRİŞ

7.11- ^ (XOR) bitsel operatörü doğruluk tablosu

XOR (^) Doğruluk tablosu

\wedge	0	1
0	0	1
1	1	0

```
int x = 5 ;  
int y = 3 ;  
int z = 5 ^ 3; // 6 olur
```

```
5 -> 0 1 0 1  
3 -> 0 0 1 1  
-----  
      0 1 1 0 -> 6
```


7- PROGRAMLAMAYA GİRİŞ

7.11- ~ (NOT) bitisel operatörü doğruluk tablosu

Değil (~) Doğruluk tablosu

~		
	~ 1	0
	~ 0	1

```
int x = 5 ;  
int y = ~ x ;
```

```
5   -> 0 1 0 1  
~ 5 -> 1 0 1 0 -> 10
```

7- PROGRAMLAMAYA GİRİŞ

Örnek: Çıkarma işlemi kullanmadan x sayısından y sayısını çıkartınız?

```
namespace ornek
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 5;
            int y = 3;
            y = ~y; // y' nin bitlerini ters çevir (complement)
            y += 1; // y' ye 1 ekle
            int z = x + y;
            Console.Write( z ); // 2 yazar
        }
    }
}
```

3-> 0 1 1
1 0 0 bitleri ter çevir
1 1 ekle
+ -----
1 0 1 (-) işaretli 3 sayısı

5-> 1 0 1
1 0 1
+ -----
(0 1 0)₂ = (2)₁₀

7- PROGRAMLAMAYA GİRİŞ

7.11- << Sola shift (kaydırma) operatörleri

```
int x = 3 ;  
int y = x << 2 ; // x , iki kez sola kaydır
```

3 -> 0 0 1 1

0 1 1 0 -> değeri 6 , birinci sola kaydırma tüm bitler bir sola kayar

1 1 0 0 -> değeri 12 , ikinci sola kaydırma

Not : Her sola kaydırmada sayı 2 ile **ÇARPILIR**. Örn. 3 değeri bir kez sola kaydırılırsa 6 eder. İki kez sola kaydırılırsa 12 eder.

7- PROGRAMLAMAYA GİRİŞ

7.11- >> Sağa shift (kaydırma) operatörleri

```
int x = 8 ;  
int y = x >> 2 ; // x'i 2 kez sağa kaydır
```

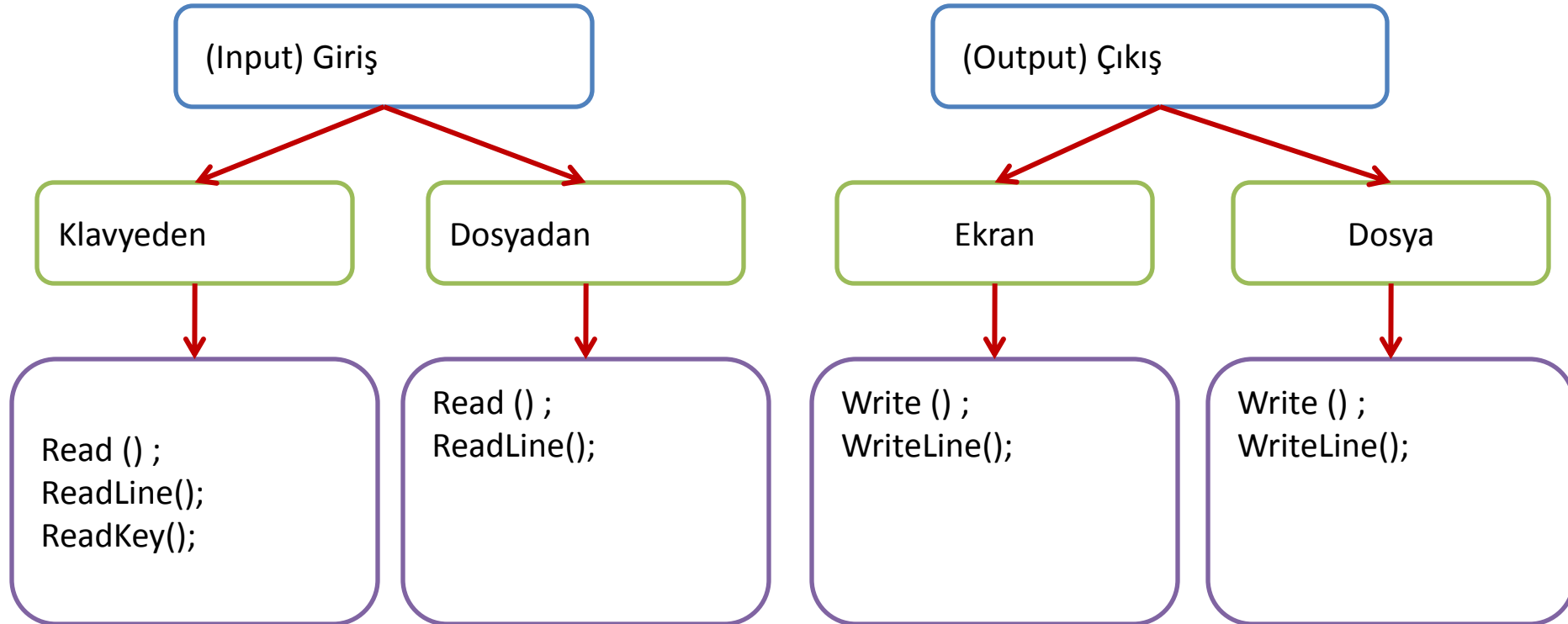
8 -> 1 0 0 0

0 1 0 0 -> değeri 4 , birinci sağa kaydırma. Tüm bitler bir sağa kayar
0 0 1 0 -> değeri 2 , ikinci sağa kaydırma

Not : Her sağa kaydırmada sayı 2 ile **BÖLÜNÜR**. Örn. 8 değeri bir kez sağa kaydırılırsa 4 eder. İki kez sağa kaydırılırsa 2 eder.

7- PROGRAMLAMAYA GİRİŞ

7.12- C# .NET I/O (Giriş / Çıkış Komutları- fonksiyonları) :



7- PROGRAMLAMAYA GİRİŞ

Program I/O (Giriş / Çıkış Komutları- fonksiyonları) :

```
int x = Console.Read( ) ; // basılan tuşun ASCII kodu okur
```

```
int y = Console.Read( ) ; // bilgi girişi beklenir girip Enter'e basınca y' ye ise Enter kodu(13) aktarılır
```

```
string adi = Console.ReadLine ( ) ; // Enter tuşuna basılınca basılan tuşları programa gönderir
```

```
Console.Write ("computer") ; // computer yazar ve imleç aynı satırda bekler
```

```
Console.Write ( 5 + 8 - 4/2-9) ; // önce işlemi yapar sonra yazar
```

```
Console.Write ( " Toplam = "+5+8) ; // otomatik tip dönüşümü yaparak «Toplam=58» yazar
```

```
Console.WriteLine (" Computer") ; // computer yazar ve imleç bir alt satıra geçer
```

7- PROGRAMLAMAYA GİRİŞ

7.13. Formatlı Yazdırma :

```
Console.Write( " Sonuç : {0} dir. ", 5 );
```

```
Console.Write( " {0} + {1} = {2} ", 5, 8, 13 );
```

```
string ad = Console.ReadLine(); // klavyeden Ali değeri girip Enter tuşuna basınız  
string soy= Console.ReadLine(); // klavyeden Can değeri girip Enter tuşuna basınız  
int v= int.Parse (Console.ReadLine()); // klavyeden 70 değeri girip Enter tuşuna basınız  
Console.Write( " Adı: {0} \n Soyadı: {1} \n Vize= {2} ", ad, soy, v ); // \n satır başı yapar
```

Ekran çıktısı

Adı : Ali
Soyadı : Can
Vize : 70

7- PROGRAMLAMAYA GİRİŞ

7.13. Formatlı Yazdırma :

```
Console.Write( " maaşı: {0,6} Yaşı:{1,4}" , maas, yas );
```

```
Maaşı :__1560 Yaşı : __ 28
```

```
Console.Write( " maaşı: {0,-6} Yaşı:{1,-4}" , maas, yas );
```

```
Maaşı :1560 __ Yaşı : 28__
```

```
Console.Write ( (2134567.679).ToString("###,###.##") ); // 2.134.567,68
```


7- PROGRAMLAMAYA GİRİŞ

(\) White Space karakterleri- Kurtarma karakteri :

Dilde özel anlamı olan karakterleri özel anlamdan çıkartır.

Özel bir anlamı olmayan karakterlere de özel anlam kazandırır

r -> r karakteri

\r -> Enter

**** -> kurtarma karakteri

**** -> \ karakteri

KARAKTER	ANLAMI
\n	New line (Satır başı)
\r	Carridge Return (enter)
\t	Horizantal (yatay) tab
\v	Vertical(dikey) tab
\f	Form feed (yazıcıdan bir sayfa)
\a	Alert (buzzer dan beep sesi)
\"	"
\'	'
\\	\

7- PROGRAMLAMAYA GİRİŞ

White Space karakterleri- Kurtarma karakteri (\) ile @ kullanımı :

```
string yol = "C:\\tablo\\notes\\renew";
```

Sonuç → C:\tablo\notes\renew

```
string yol = @ "C : \tablo\notes\renew";
```

Sonuç → C:\tablo\notes\renew

7- PROGRAMLAMAYA GİRİŞ

White Space karakterleri- Kurtarma karakteri(\) ile @ kullanımı :

```
string s = "MAKU \nMYO\nBILGISAYAR";
```

```
MAKU
MYO
BILGISAYAR
```

```
string k = @ "windows\nsystem\nwin32";
```

EKRANA NASIL YAZAR ?

```
windows\nsystem\nwin32
```



SORULAR

1-Aşağıdaki değişken tanımlamalarından hangisi geçerli bir tanımlamadır ?

A) ortalama not b) case c) 2vize d) @facebook e)_w

2- $x = x + 1$; dengi olan ifadesine aşağıdakilerden hangisi olabilir?

I-) $x++$; II-) $++x$; III-) $x+=1$;

A) I-II b) I c) III d) I-II-III e) I-IV

3- `int x=4, y=3; int z=x & y;` ise z nin değeri ne olur?

a) 0 b) 1 c) 2 d) 3 e) 5

4- Aşağıdakilerden değişkenlerin hangisi macar yöntemine göre gösterilmiştir.?

a) vize b) Vize_int c) vize_1 d) vint e) int Vize

5- `string s=@ "MYO\nBIL";` komutuna göre s değişkenin değeri ekrana yazdırılırsa nasıl bir çıktı veririr?

a) MYO\nBIL b) MYO c) @MYO d) @MYO\nBIL e) @MYO
BIL @BIL BIL

SORULAR

6- `int x = (5 >=k) ? 8 : 9 ;` komutuna göre `k`' nın 5 değeri için `x` 'in değeri nasıl olur ?

A) `x` b) `k` c) `8` d) `9` e) `5`

7- `Console.Write (Convert.ToString(15,16)) ;` komutundan sonra ekranda ne yazar ?

A) `f` b) `15` c) `16` d) `"15,16"` e) I Hiçbiri

8- C# . Net için aşağıdakilerden hangisi doğrudur ?

I- Oluşturulan programın çalışması için bilgisayarda framework kurulu olmalı

II- Nesne Tabanlı bir dildir

III- Web uygulamaları geliştirmeye müsait bir dildir

a) I b) I – II c) II- III d) I-II-III e) I-III

9- `string s = "\\\\";` komutuna göre `s` değişkenin değeri ekrana yazdırılırsa nasıl bir çıktı veririr ?

a) `\` b) `\\` c) `\\\` d) `\\\\"` e) Hiçbir şey yazmaz

10- `string s = "Derinliği " + 15 + 22 + " m.";` komutuna göre `s` değişkenin değeri ekrana yazdırılırsa nasıl bir çıktı veririr ?

a) Derinliği 1522 m. b) Derinliği 37. c) Hata veririr d) Hiçbir şey yazmaz e) null yazar

Yazılım Geliştirme Modelleri

Yazılım Geliştirme Modelleri

Yazılım Geliştirme Modelleri

Daha emniyetli yazılımların daha kısa sürede, daha az bütçeyle ve daha az hatayla geliştirilmesi için sürekli yeni teknolojiler ve modeller bulunmaya çalışılmaktadır. Bunlardan bazıları;

- I) Gelişigüzel Geliştirme
- II) Şelale (Waterfall) Modeli
- III) V- Modeli
- IV) Barok Modeli
- V) Helezonik (Spiral) Model
- VI) Arttırımsal (Incremental) Geliştirme Modeli
- VII) Döngüsel Model
- VIII) Çevik Yazılım Geliştirme Metodları

Yazılım Geliştirme Modelleri

I) Gelişigüzel Geliştirme

Gelişigüzel geliştirmede belirlenmiş bir model ya da yöntem bulunmaz.

Genellikle kişiye bağlı yazılım geliştirme şeklinde yapılır ve bu yüzden yazılımın izlenebilirliği, bakım yapılabilirliği oldukça zordur.

1960'lı yıllarda uygulanan bu yöntem, genellikle basit programlama içeren ve çoğunlukla tek bir kişinin üretim yaptığı yöntemidir.

II) Şelale (Waterfall) Modeli

Şelale modeli yakın zamanlara kadar en popüler yazılım geliştirme modeli olarak görülmüştür.

Geleneksel yazılım geliştirme modeli olarak da bilinir.

Şelale modelinde yazılım, aşamalar en az birer kez tekrarlanarak geliştirilir.

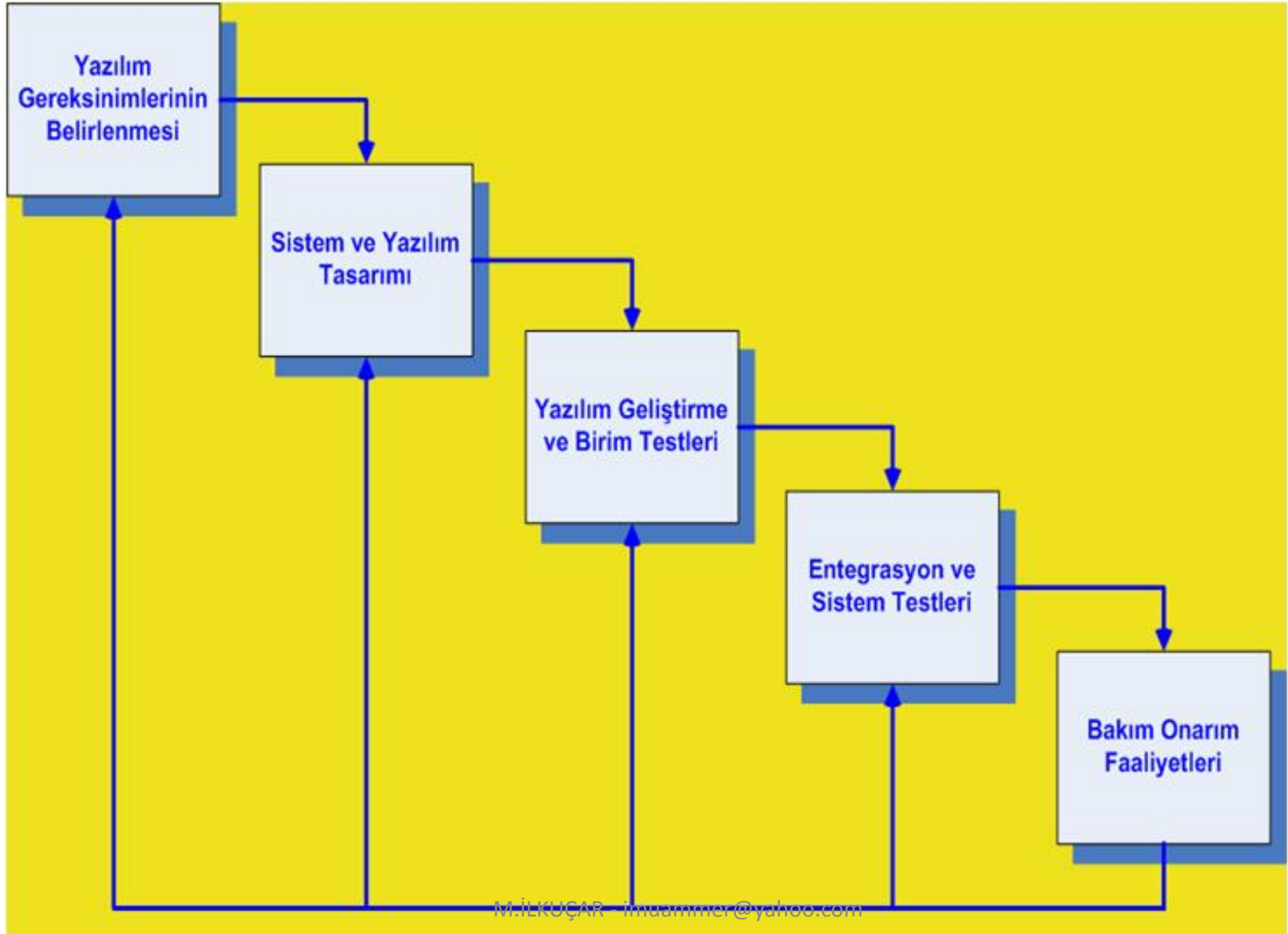
Çok iyi tanımlanmış ve üretimi az zaman gerektiren projeler için uygun bir model olmakla birlikte günümüzde kullanımı gittikçe azalmaktadır.

III) V -Modeli

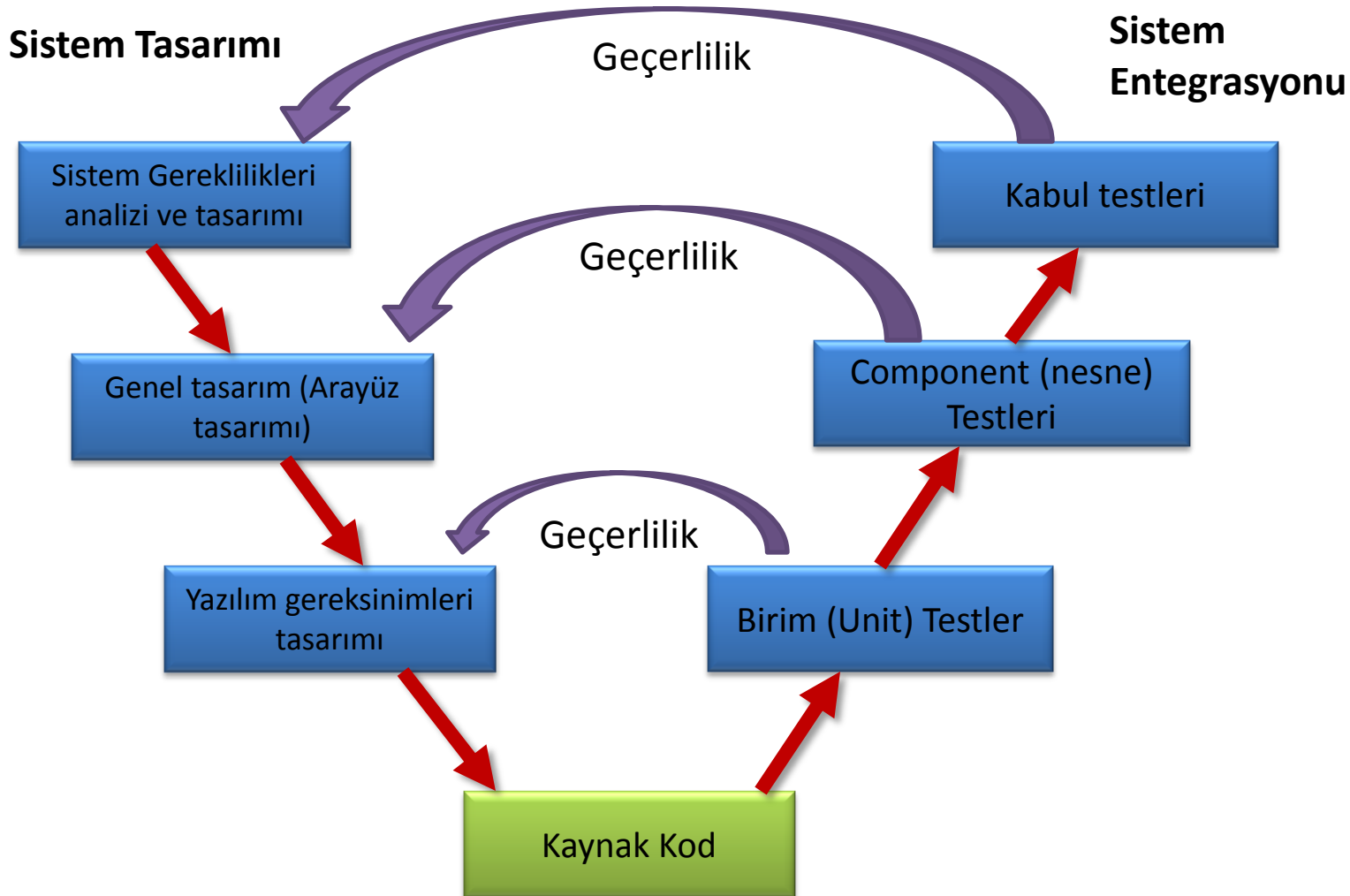
Bu model popüler geliştirme modellerindendir. Sistemin iki bölümden oluşur ;tasarım ve kabul.

Kabul bölümünde isteklere göre tasarım aşamasının uygun adımına gidilerek programda düzeltmeler yapılabilir.

Yazılım Geliştirme Modelleri- Şelale (Waterfall) Modeli



Yazılım Geliştirme Modelleri- V Modeli



Yazılım Geliştirme Modelleri

IV) Barok Modeli

1970'li yıllarda ortaya çıkan Barok modelinde, program yaşam döngüsü temel adımları doğrusal bir şekilde ele alınır ve geliştirilir. Aşamalar arasında gereken geri dönüşlerin nasıl yapılacağı tanımlı değildir.

Bu modelde, dokümantasyon günümüz modellerinden farklı olarak ayrı bir süreç olarak ele alınır ve yazılımın geliştirme ve test faaliyetleri tamamlandıktan sonra yapılmasını öngörür. Günümüz yazılım geliştirme projelerinde uygulanan bir model olmaktan çıkmıştır.

V) Helezonik (Spiral) Model

Spiral yazılım geliştirme modeli temel olarak dört ana bölüm içerir.

Bunlar, planlama, risk yönetimi, üretim ve kullanıcı değerlendirmeleri olarak tanımlanabilir.

Planlama, üretilecek ara ürün için işin planlanması, amaç ve kısıt ve alternatiflerin belirlenmesi, bir önceki adımda üretilmiş olan ürün ile tümleştirme yapılması faaliyetlerini içerir. Risk yönetiminde, alternatifler değerlendirilir ve risk analizi yapılır. Üretim, planlanmış ara ürünün geliştirildiği aşamadır. Bu aşamadan sonra, kullanıcı değerlendirmesi kısmında, ara ürün hakkında kullanıcıların test ve değerlendirmeleri yapılır.

Yazılım Geliştirme Modelleri

VI) Arttımsal (Incremental) Geliştirme Modeli

Arttımsal model, yazılımın küçük parçalara ayrılarak döngüsel olarak geliştirilmesi fikrine dayanır. Proje süresi, artırım (veya döngü) olarak tanımlanan küçük zaman dilimlerine bölünür. Proje bir çok döngünün gerçekleştirilmesi ile ilerler. Her döngünün sonunda, projeye ait planlanmış çıktılar elde edilir ve yazılıma yeni bir fonksiyonalitye eklenir. Bu sayede yazılım arttımsal olarak geliştirilir.

Projenin bir döngüsünde henüz tümleştirme süreci sonlanmamışken , diğer bir döngünün döngünün tasarım süreci başlayabilir. Dolayısı ile, bu model yazılım geliştirmenin doğasına daha uygun olarak görünmektedir. Her döngüde yeni bilgi ve tecrübeler edinilir ve bunlar projenin geliştirilmesi aşamasında çok değerli katkılar yapar.

Arttımsal modelin en önemli avantajlarından biri, projenin ilk safhalarında elde edilen çıktıların projenin ilerleyen aşamalarında değişikliğe uğraması halinde bile büyük bir maliyete neden olmadan bu değişikliklerin yapılabilir olmasıdır.

Yazılım Geliştirme Modelleri

VII) Döngüsel Model

Döngüsel yazılım geliştirme modeli artırımsal model çok benzerlik taşır. Bazı kaynaklar döngü veya artırım içeren modelleri evrimsel model olarak nitelendirirler.

Döngüsel yazılım geliştirme modeli, proje yaşam döngüsündeki tüm süreçleri içeren döngülerden oluşur. Artırımsal modelden farkı döngülerin içerdiği süreçlerdir.

Artırımsal modelde, her döngüde tasarım, kodlama, test ve entegrasyon süreçleri bulunurken döngüsel modelde planlamadan başlayarak tüm proje süreçleri kapsanır.

Yazılım Geliştirme Modelleri

VIII) Çevik Yazılım Geliştirme Metodları

Çevik yazılım geliştirme metodları, hantal olduğu düşünülen yazılım geliştirme modellerine bir alternatif olarak 90'lı yılların ortalarında gelişmeye başlamıştır.

Bu modelde döngüler içerir ve bu anlamda döngüsel modelle benzerlikler taşır. Ancak, çevik metodlarda döngüler çok kısa döngü sayısı da oldukça fazladır.

Bu metodlar, proje içerisindeki her bir döngüde yazılımın bir sürümünü küçük bir proje olarak ele alır ve bunu tedarik edenin katılımı ile test eder.

Her döngüde projenin öncelikleri yeniden belirlenir ve proje boyunca ortaya çıkan değişikliklere uyum sağlanır.

Bu metodlar, doküman üretiminin yazılım geliştirme faaliyetlerini yavaşlattığına inandığından doküman üretme yerine yüz yüze görüşmelere ağırlık verir.

Döngüler sonucunda ortaya çıkan yazılım birimleri tedarik makamı tarafından değerlendirilir ve yapılan her yorum yeni bir versiyon olarak yazılıma eklenir.