



Swift & iOS Uygulama Geliştirme Eğitimi

Arrays



```
var boşArray = [String]()
```

```
var boşArray2 = Array<String>()
```

```
var optionalArray:Array<String>?
```

Arrays



```
var birimler = [  
    "İdari ve Mali Hizmetler",  
    "Bilişim Hizmetleri",  
    "Ağ ve Sistem Yönetimi",  
    "Teknik Hizmetler",  
    "Yazılım Geliştirme"]
```

Arrays



```
var idariBirim = birimler.first
```

```
var teknikHizmetler = birimler[3]
```

```
var yazılımBirimi = birimler.last
```

```
birimler[1] = "Değişken Birim"
```

Arrays



```
birimler.append("Sürpriz Birim")
```

```
birimler.removeLast()
```

```
birimler.removeAtIndex(0)
```

```
birimler.removeAll(keepCapacity: false)
```

```
birimler = []
```

Dictionaries



```
var boşDictionary = [String,String]()
```

```
var boşDictionary2 = Dictionary<String,String>()
```

```
var optionalDictionary:Dictionary<String,String>?
```

Dictionaries



```
var öğrenciBilgileri = [  
    "Ad"           : "Süleyman" ,  
    "Soyad"        : "Çalık"   ,  
    "Numara"       : "12345"   ,  
    "KayıtYılı"    : "2014"   ]
```

Dictionaries



```
öğrenciBilgileri.updateValue("Halil", forKey: "Ad")
```

```
öğrenciBilgileri.removeValueForKey("Numara")
```

```
öğrenciBilgileri["Sınıf"] = "1"
```


Function



```
// Parametre almayan ve dönüş değeri olmayan fonksiyon
func merhabaDe() {
    println("Merhaba!")
}
```

```
// Bir Parametre alan ve dönüş değeri olmayan fonksiyon
func merhabaDe(isim:String) {
    println("Merhaba: \ (isim)")
}
```

Function



```
// İki Parametre alan ve dönüş değeri olmayan fonksiyon
func birseySoyle(birsey:String , isim:String) {
    println(birsey + " " + isim)
}
```

```
// İki Parametre alan ve dönüş değeri Int olan fonksiyon
func topla(ilkSayi:Int , ikinciSayi:Int) -> Int {
    return ilkSayi + ikinciSayi
}
```

Function



```
// Int parametre alan ve Tuple dönen fonksiyon
func ikiyeBol(sayi:Int) -> (Int,Int) {
    var bolum = sayi / 2
    var kalan = sayi % 2

    return (bolum,kalan)
}
```

Class



```
class User {  
    var firstName:String  
    var lastName:String  
  
    init(firstName:String, lastName:String) {  
        self.firstName = firstName  
        self.lastName = lastName  
    }  
  
    func fullName() -> String {  
        return firstName + " " + lastName  
    }  
}
```

Class



```
var kullanıcı = User(firstName: "Bill", lastName: "Gates")  
var kullanıcıAdı = kullanıcı.fullName()  
// Bill Gates
```

Class



```
class Student : User {  
    var number:Int  
  
    init(firstName: String, lastName: String, number:Int) {  
        self.number = number  
        super.init(firstName:firstName, lastName:lastName)  
    }  
}
```

Class



```
var öğrenci =  
Student(firstName:"Steve", lastName:"Jobs", number:123)  
  
var öğrenciAdı = öğrenci.fullName()  
// Steve Jobs
```

Struct



```
struct Vapur {  
    var kalkış : String  
    var varış : String  
    var yolcuSayısı: Int  
}
```

```
var kadıköyVapuru =  
Vapur(kalkış: "Karaköy", varış: "Kadıköy", yolcuSayısı: 200)
```


Struct vs Class



```
class User {  
    var name: String  
    init(name: String) {  
        self.name = name  
    }  
}
```

```
var userA = User(name: "Steve")  
var userB = userA // userA ve userB aynı adresi gösteriyor!  
userB.name = "Bill"
```

```
println(userA.name) // "Bill"  
println(userB.name) // "Bill"
```

Struct vs Class



```
struct User {  
    var name: String  
    init(name: String) {  
        self.name = name  
    }  
}
```

```
var userA = User(name: "Steve")  
var userB = userA // userA ve userB farklı 2 adresi  
gösteriyor!  
userB.name = "Bill"
```

```
println(userA.name) // "Steve"  
println(userB.name) // "Bill"
```

Playground Zamani



View



- “View” temelde, dörtgen bir alanı ifade eder.
- İlgili alana çizim yapar ve kullanıcı aksiyonlarını karşılar.
- Her bir view’in, tek bir superView’i vardır.
- Ancak birçok subview’i olabilir.
- Subview’lerin sırası önemlidir.
- Son eklenen en üstte görünür.

Window



```
class UIWindow : UIView {  
    ...  
}
```

- View hiyerarşisinin en tepesindeki view
- Uygulamalarda tek bir window olur
- Storyboard'lardan sonra pek kullanılmıyor

View Coordinates



- **CGFloat**
 - Grafik kütüphanesinde kullanılan float tipi
- **CGPoint**
 - **x** ve **y** adında CGFloat değerleri tutan bir struct
- **CGSize**
 - **width** ve **height** adında CGFloat değerleri tutan bir struct
- **CGRect**
 - **origin** adında bir CGPoint
 - **size** adında bir CGSize tutan bir struct

View Coordinates



(0,0)

X

- View'lerin Koordinat başlangıcı sol-üsttür
- Birimi point'tir (pixel değil)
- UIView, 3 önemli property barındırır
 - var frame: CGRect
 - var bounds: CGRect
 - var center: CGPoint

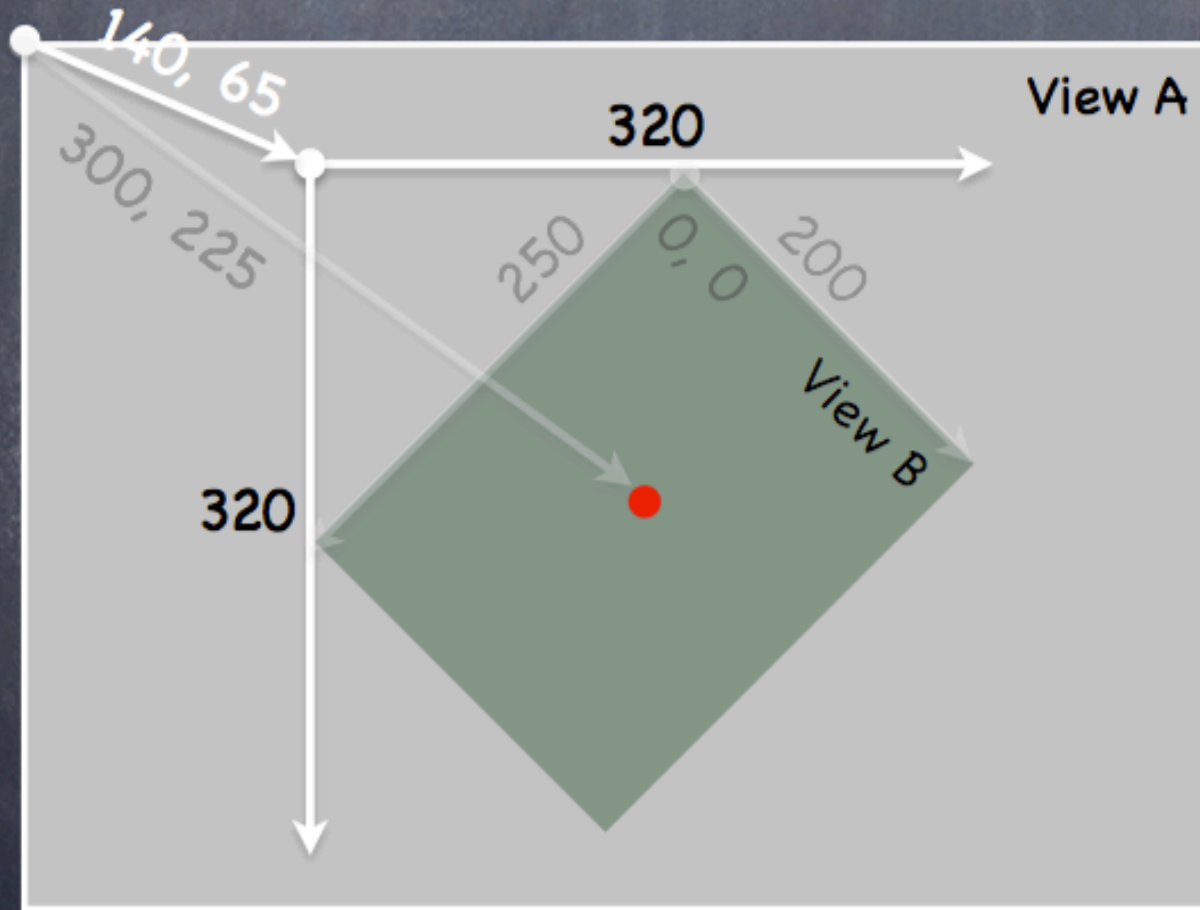
Y

View Coordinates

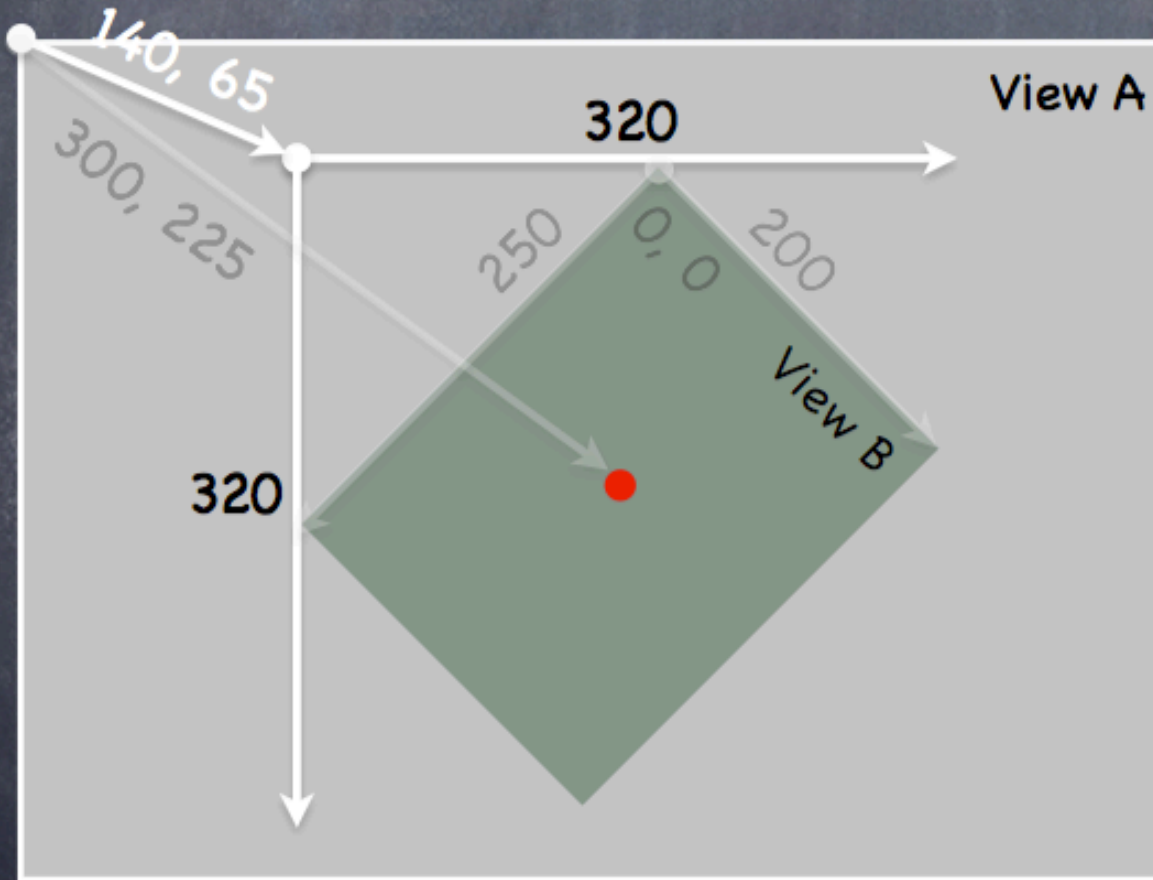


- **bounds**
 - Bir view'in kendisine göre koordinat durumu
- **frame**
 - Bir view'in superView'ine göre olan koordinat durumu
- **center**
 - Bir view'in superView'e göre merkez noktası

View Coordinates

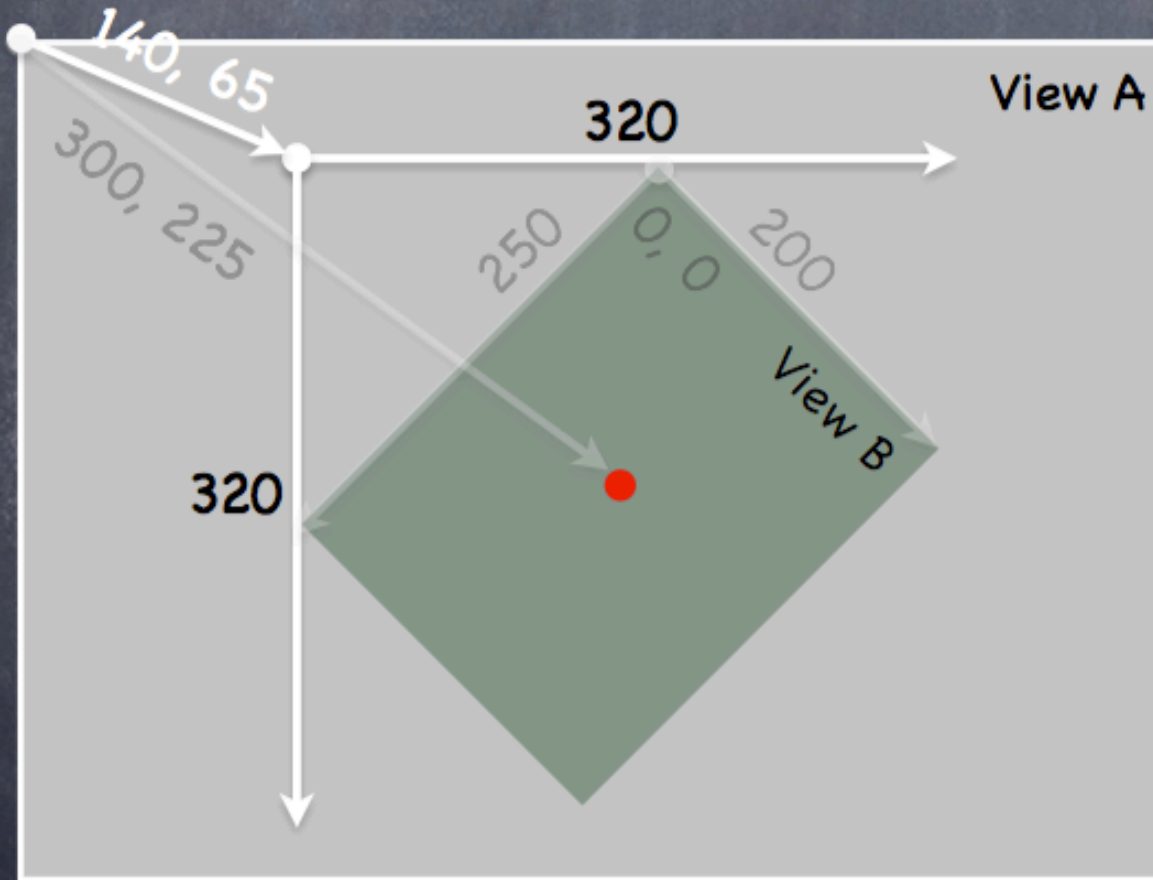


View Coordinates



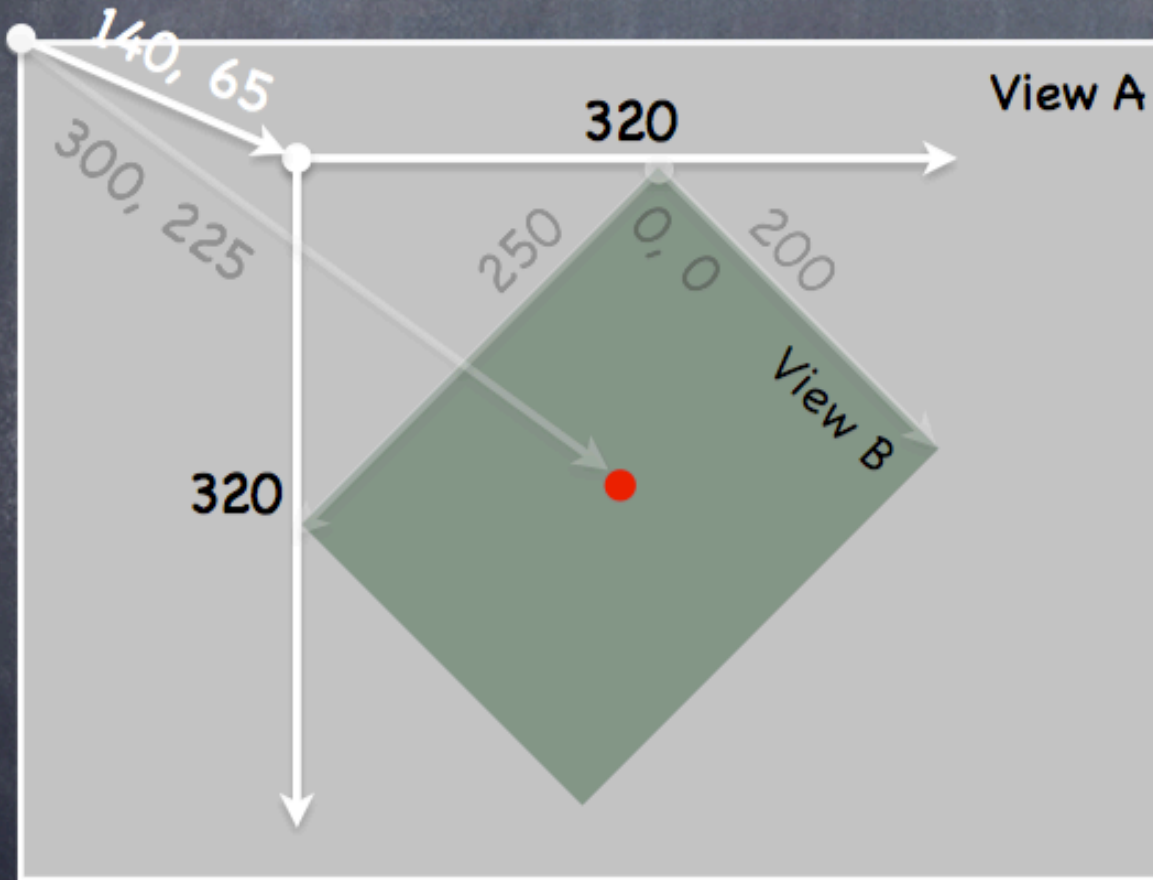
B'nin **bounds** değeri ?

View Coordinates



B'nin **bounds** değeri
((0,0) , (200,250))

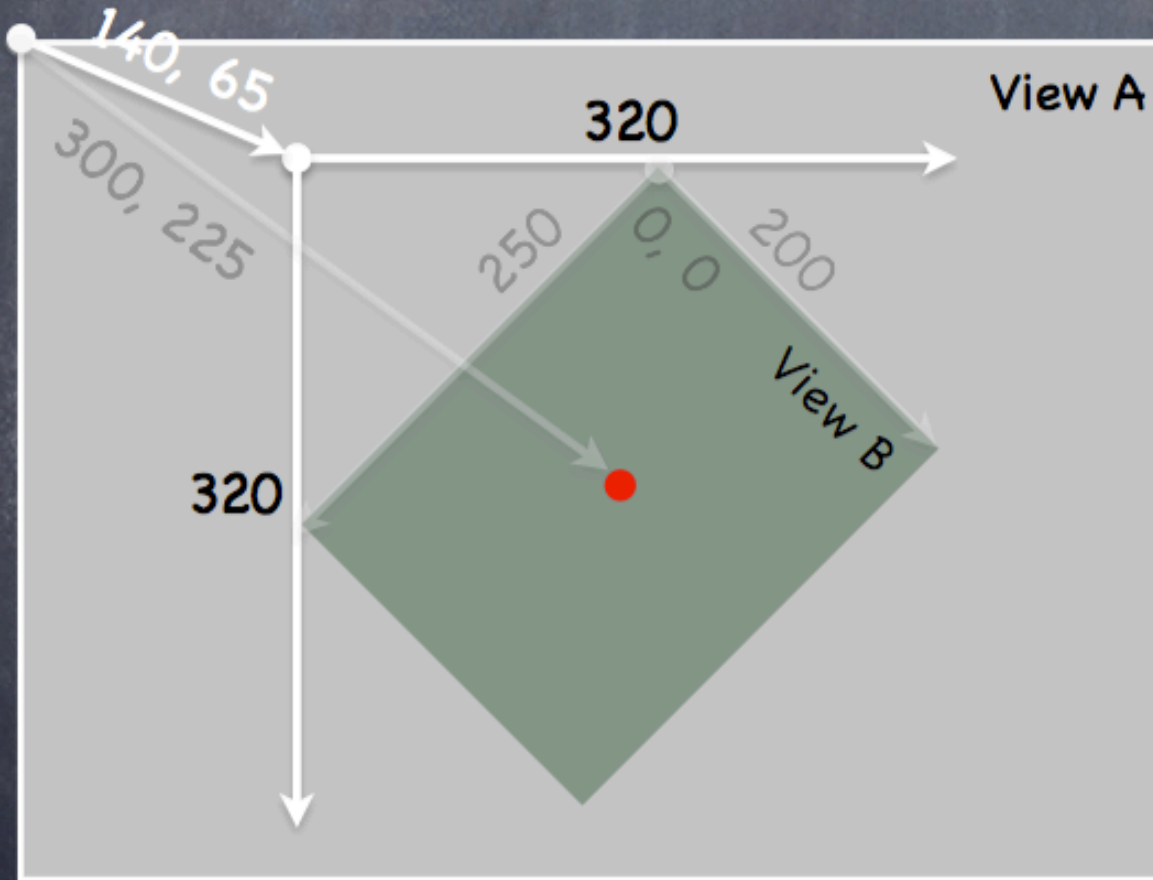
View Coordinates



B'nin **bounds** değeri
((0,0) , (200,250))

B'nin **frame** değeri ?

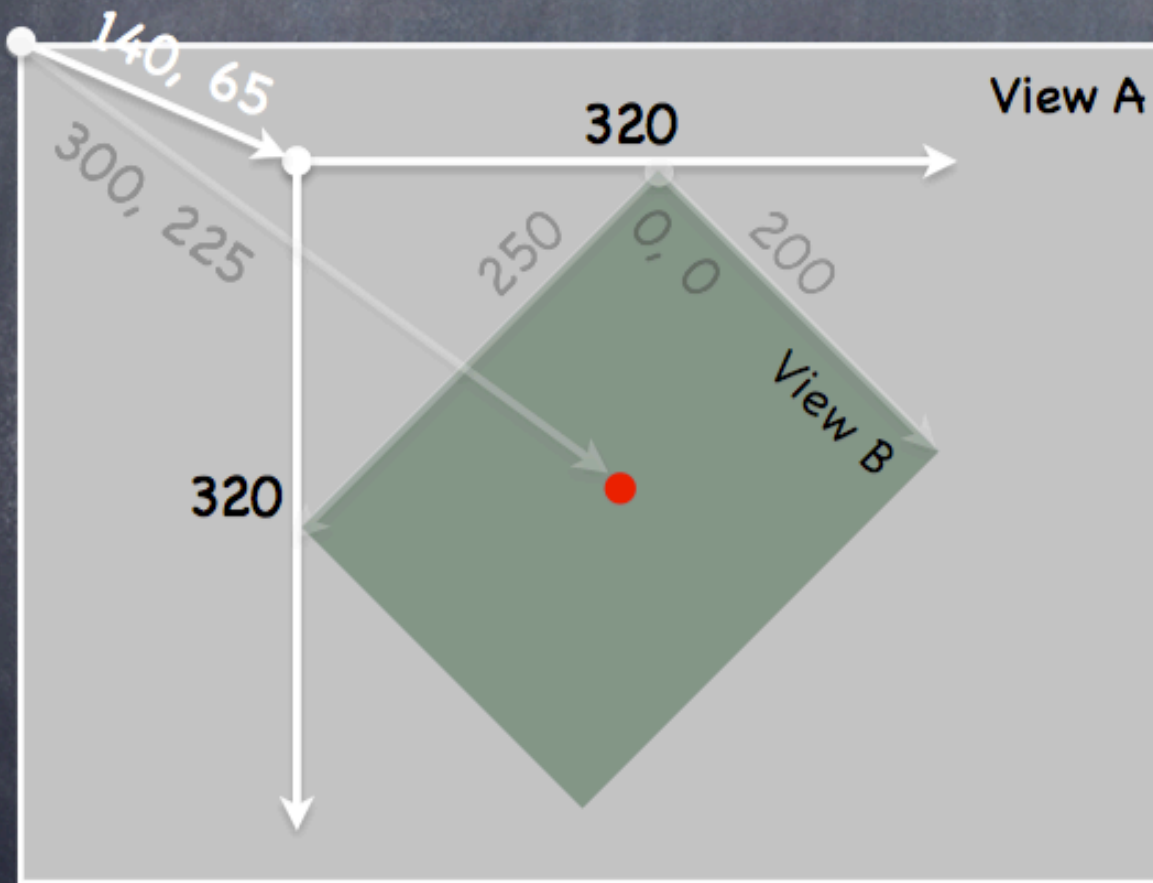
View Coordinates



B'nin **bounds** değeri
((0,0) , (200,250))

B'nin **frame** değeri
((140,65),(320,320))

View Coordinates

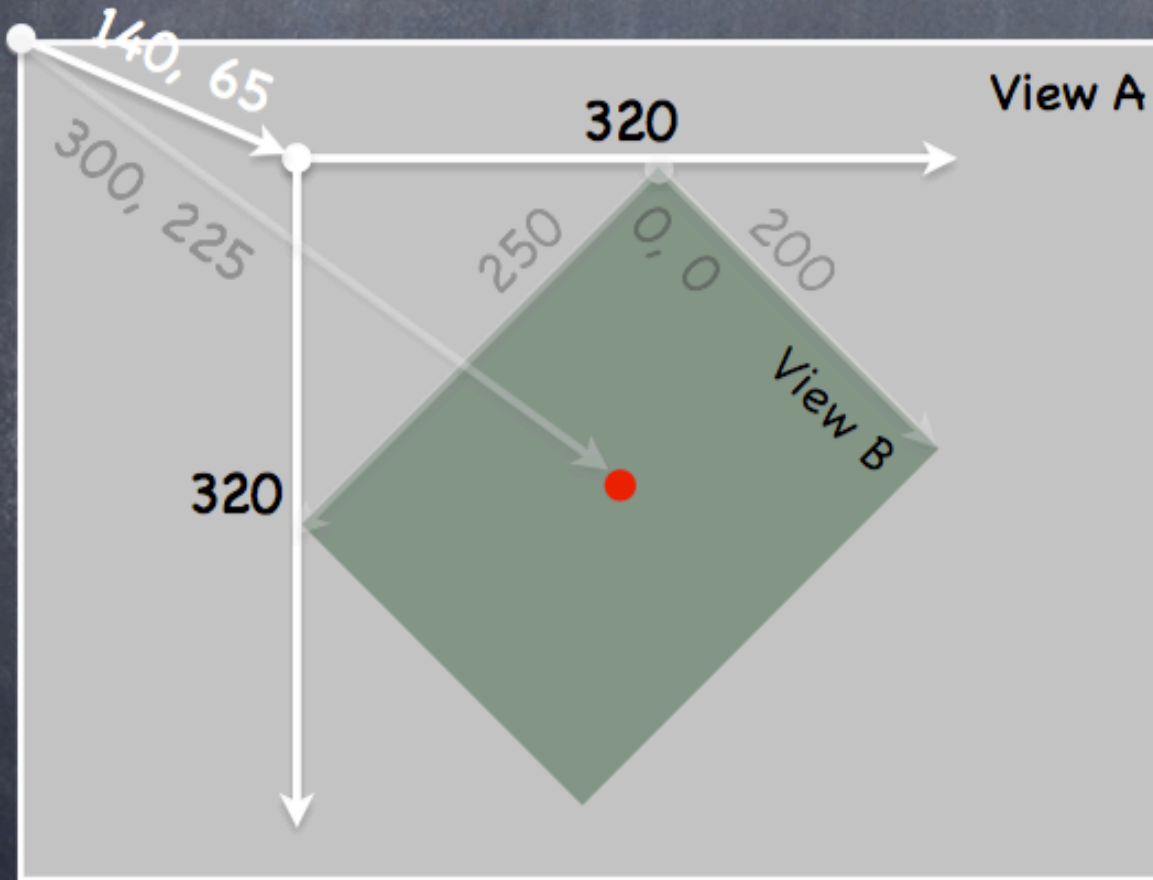


B'nin **bounds** değeri
((0,0) , (200,250))

B'nin **frame** değeri
((140,65),(200,250))

B'nin **center** değeri ?

View Coordinates



B'nin **bounds** değeri
((0,0) , (200,250))

B'nin **frame** değeri
((140,65),(320,320))

B'nin **center** değeri
(300,225)

View Coordinates



- `self.frame = self.superview.frame`
yerine
- `self.frame = self.superview.bounds`
kullanın