Not: Triple apostrtophe(‘’’ „„ ‘’’) birden fazla satıra yayılan dizeler oluşturmada kullanılır.

Not: “super” keywordü ata sınıfın özelliğini alıp kullanması için tanımlanır.

Not: Bir değişkene on pressed içerisinde parantez verildiği zaman, onpressed bir atama yapmak, bir değer döndürmek isteyeceğinden hata verir. Anonim bir fonksiton içerisnde parantez verilip parametre tanımlanan bir fonksiyon tanımlanırsa sorun olmayacaktır.

Not: Dart da çoğu ifadeyi .toString() ile string veri yapısına dönüştürmek mümkün.

Not: “Interpolation” kavramı temel olarak tanımlanmış iki değişkenin +(plus) işareti ile değil de parantezler içerisinde “$” işareti kullanımı ile birleştirilmesi işlemidir.
‘${degiskenismim .round()}’

Not: String içinde $ gibi özel ifadeleri kullanmak için “\” kullanılır. “String \$”

Not: *StateManagment:* Hangi değişkeni nerde yöneteceğimizi, widget tree’yi nasıl oluşturacağımızı, değişkenleri birbirinden nasıl ayıracağımızı, hangi değişkene nereden ne zaman ulaşacağımızı ve değiştirme yetkisi vereceğimizi belirlememizi sağlayan, *arabirim denetim durumunun yönetimi’*dir.

Not: setState, State class’ının yanı örneğin stateless veya statefull sınıfının bir metodudur. Bir widget içinde bu widget’e ait setState metotunu kullanan başka bir widgetı bu sebeple extract edemeyiz.

Not: *Syntactic Sugar* Örneği: (context) {return RoutePink();} / (context) => RoutePink()

Not: Dart Null Safety içerir. Bu özellik devre dışı bırakılabilir.

Not: *Map:* veri saklama yapısıdır. Key ve value değerlerini birbiri ile eşleştirir.

Not: *Object*: bütun veri tiplerini Dart obje olarak tutar. Tüm veri tiplerinin atasıdır.

Not: *Static:* herhangi bir obje üretilmese dahi programın her yerinden ulaşılabilmeyi sağlar.

Not: *Final:* daha sonra değiştirilemez bir değişken keyword’üdür.

Not: *const* compile time’da yani programımız henüz çalışmaya başlamadığı anda kullanılacak sabitler için, *final* ise run time’da çalışacak sabitler için tanımlanır.

Not: initialRoute: Uygulama ilk açıldığında gidilmesi gereken route’u ifade eder.

Not: Callback içerisinde parantez kullanmayız çünkü yalnızca adresini ifade etmek isteriz.

Not: extends yerine with: tamamen mirascısı olmasın ama metotlarını ve ozlelıklerını kullanbılsın demektir. Böylece başka sınıfların da metotlarına ulaşabilir bir hale gelir. Örneğin, (Class ClassIsmım with IlkSınıf, IkıncıSınıf, UcuncuSInıf{}).

Not: Parantez içerisindeki argüman gövde içerisinde kullanılmıyor ama yine de parantez içerisinde parametre olarak belirtilmesi gererkiyorsa “_” kullanabiliriz.

Not: Uygulama mimarisinde, her iş ve servisin uzmanı sınıflar farklı olmalı ve loosely-coupled, yani birbirlerine en zayıf şekilde bağlı olmalılardır. Arayüz ekranlarının olabildiğince bu tip servis ve verilerden ayırılması gerekir.

Not: Syntactic Sugar Önerisi: "*??*", yazılan değerin null olması halinde devamına yazılan değeri atama işlemi yapar. Yani null koşulunu/kontrolünü gerçekleştirir.

Not: => rotasyonu yalnızca bir satırlık returnler için kullanılan bir kısaltma işlemidir.

Not: "static" anahtar kelimesi, bir sınıfın sahip olduğu verilerin farklı nesneleri arasındaki değerleri kalıcı hale getirmesine olanak tanır. Elimizdeki sınıfın herhangi bir obje oluşturulmadan kullanılabilecek, hemen ulaşılabilecek bir instances field tanımlayabilmemizi sağlar.

Not: *Iterable Sınıfı* Dart da listelerin bir üst atası olan sınıfıdır.

Not: List list1 = ['a', 'b', …list2, …list3] : list 2 listesinin elemanlarını list 1 listesinin devamına ekler.

Not: Flutterda tüm widgetlar bir sınıftır.

Not: Stream de return yerine yield kullanılır. async yerine async* kullanılır.

Not: Statefull widget ilk çalıştığında build metodu çalışmadan önce state'i başlatırken yapılması gerekenler *iniState()* içerisine yazılır.

Not: "singleton" sadece bir tane obje üretilmesi gereken durumları ifade eder.

Not: *Regular Expression:* Bu gibi girilen metinleri kontrol etme durumları ve kısıtları RegEx başlığı altında inceleniyor.

Not: "factory" anahtar kelimesi: yapıcıların başına eklenen bir keyword. Yapıcıda return kullanılıyorsa ve duruma göre obje oluşturuluyor ise kullanılır (return … diyip bir obje oluşturuluyor ise).

Not: ListView ile çalışırken Flexible ve Expanded lar hayat kurtarıcıdır.

Not: Null Coalescing Operator "??"

Not: *Provider'ın state management paketi olarak dezavantajı:* Her yeni widget dalı için ayrı context oluşturulur ve bu contextler yalnızca kendi içerisini ilgilendirir. Diğer contextlere erişim sağlayamaz. Provider context vasıtası ile iletişim kurabilen bir paket. Oysa GetX veya riverPod context ten, provider dan bağımsız çalışır.

Not: statein stateful içindeki veriye ulaşabilmesi için "widget" yazmamız gerekir.