

Proje Raporu

Özet—Bu proje, "Magic Coffee" adıyla geliştirilen, kullanıcıların kahve ve tatlı siparişi verebildiği ve siparişlerini gerçek zamanlı olarak takip edebildiği bir mobil uygulamadır. Uygulama, kullanıcıların istedikleri kahve veya tatlıyı seçmelerini ve ödeme işlemlerini kayıtlı kredi kartı bilgileriyle gerçekleştirmelerini sağlar. Backend tarafı MySQL veritabanı kullanılarak geliştirilmiştir ve users, baristas, branches, orders, coffees, desserts ve credit_cards gibi çeşitli tabloları içerir. Uygulama, kullanıcılara bir profil sayfası, sepet yönetimi haritadan şube seçimi gibi özellikler sunar. Barista yönetimi ve sipariş takibi entegrasyonları sayesinde uygulama, kahve dükkanı operasyonlarını kolaylaştırarak kullanıcı deneyimini geliştirir.

I. Proje Tanımı

Günümüzde kahve dükkanları hem fiziksel sipariş hem de mobil uygulamalar üzerinden sipariş alma süreçlerini optimize etmeye çalışmaktadır. Ancak pek çok küçük ve orta ölçekli kahve dükkanı, kullanıcıların mobil cihazlardan sipariş verebileceği, siparişlerinin durumunu takip edebileceği ve personel ile etkileşim kurabileceği entegre sistemlerden yoksundur.

Magic Coffee müşterilerin mobil cihazlar üzerinden kahve ve tatlı siparişi verebilmesini, siparişlerini takip edebilmesini yöneticilerin ise şubeleri ile ilgili işlem yapabilmesini sağlar

II. Yapılan Araştırmalar

- Proje geliştirilirken, başlangıçta kullanıcı arayüzü değişikliklerini yönetmek için setState kullanıldı. Ancak yaptığımız araştırmalar sonucunda, setState'in büyük projelerde sürdürülebilir olmadığı ([1]) ve karmaşıklığı artırdığı fark edildi. Bunun yerine, durumu daha verimli ve profesyonel bir şekilde yönetmek için Provider paketi tercih edildi. Provider, merkezi bir durum yönetimi sağlayarak, yalnızca gerekli bileşenlerin yeniden render edilmesini sağlar, böylece uygulamanın performansı artırılmış olur. Bu sayede uygulamanın durum yönetimi daha modüler ve sürdürülebilir hale geldi.
- Proje geliştirilirken, başlangıçta uygulama içindeki veri yönetimini basit bir liste yapısı ile gerçekleştirmeye çalıştık. Ancak, veri büyüdükçe uygulama performansında önemli düşüşler gözlemlendi. Bu sorunun üstesinden gelmek için, veri yönetimi ve depolama işlemlerini daha verimli hale getirmek amacıyla Future yapısı araştırıldı.[2] Sonuç olarak, asenkron veri işlemleri ve akış yönetimi için bu yapıları kullanarak veri işleme süreci hızlandırıldı. Bu yaklaşım sayesinde, kullanıcıya daha hızlı yanıt veren ve daha stabil bir uygulama deneyimi sağlandı
- Yapılan araştırmalarda, Flutter'da http paketinin kullanımı yaygın bir çözüm olarak öne çıkmaktadır.

(http dökümanları: [3]) http paketi, RESTful API'lerle kolayca iletişim kurmayı ve JSON formatındaki verileri işlemeyi sağlar. Projeye bu paket dahil edilerek, API çağrıları yapıldı ve alınan JSON verileri model sınıflarına dönüştürüldü. Bu sayede, uygulama verilerini işlemek daha kolay hale geldi.

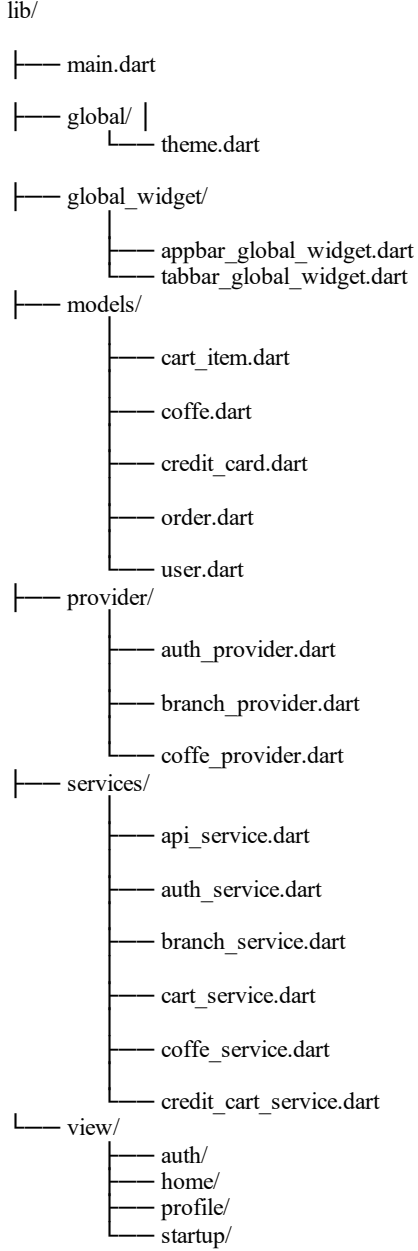
III. Yazılım Mimarisi

Magic Coffee projesinde yazılım mimarisi, katmanlı ve modüler bir yapı üzerinde planlanmıştır. Projenin mobil arayüzü Flutter ile geliştirilip uygulamanın giriş noktası main.dart dosyasıdır. Ekranlar view/ klasöründe, veri modelleri models/, sunucuya yapılan API istekleri services/, özel tasarlanmış arayüz bileşenleri ise global_widget/ klasöründe organize edilmiştir. Frontend tarafında IconButton ,SnackBar ,TextField, Container gibi Flutter widgetleri kullanılmıştır. API katmanı ile iletişim için HTTP paketi kullanılmıştır. State management (durum yönetimi) için Flutter'da yaygın olarak kullanılan Provider paketi tercih edilmiştir , sql tarafında Mysql , api tarafında ise node.js tercih edilmiştir.

A. Proje Katmanları

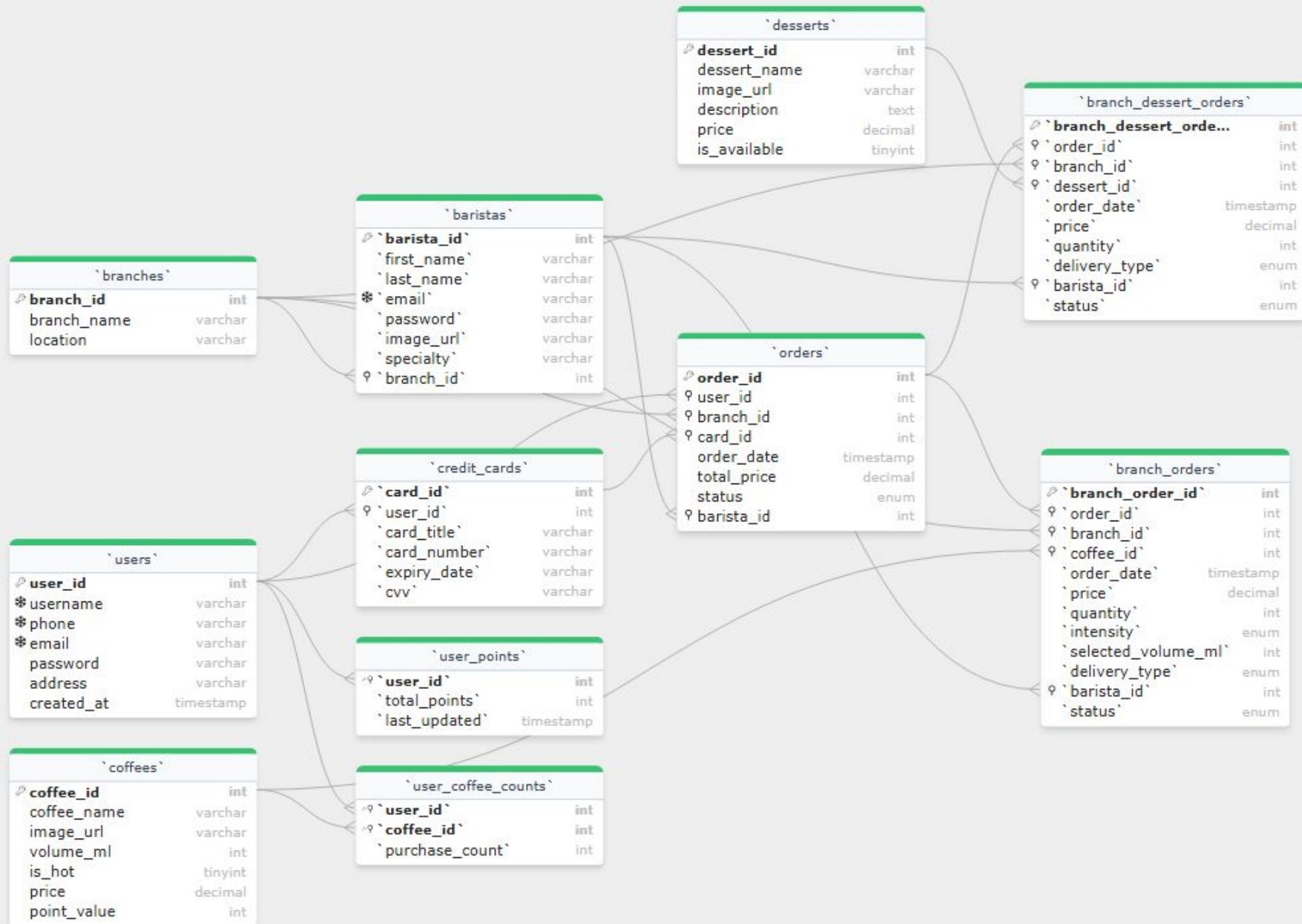
- Kullanıcı Arayüzü (UI)(view ,global, view_global dosyaları), Katmanı: Flutter framework'ü ile geliştirilen bu katman, kullanıcı ile etkileşimi sağlar. Kullanıcı sayfaları ve widget'ları bu katmanda yer alır.
- Durum Yönetimi (provider dosyası) Katmanı: Uygulamanın durumu, Provider paketi ile yönetilmektedir. Bu katman, uygulamanın veri akışını yönetir ve kullanıcı etkileşimlerine göre UI'yi günceller.
- Servis (services dosyası) Katmanı: Veritabanı işlemleri ve dış API hizmetleri bu katmanda yönetilir. API servisleri,ve veritabanı işlemleri burada yer alır.
- Model(models dosyası) Katmanı: Verilerin temsil edildiği model sınıflarını içerir. Bu katmanda cart_item.dart, coffe.dart, user.dart gibi modeller yer alır ve uygulamanın veri yapıları burada tanımlanır.

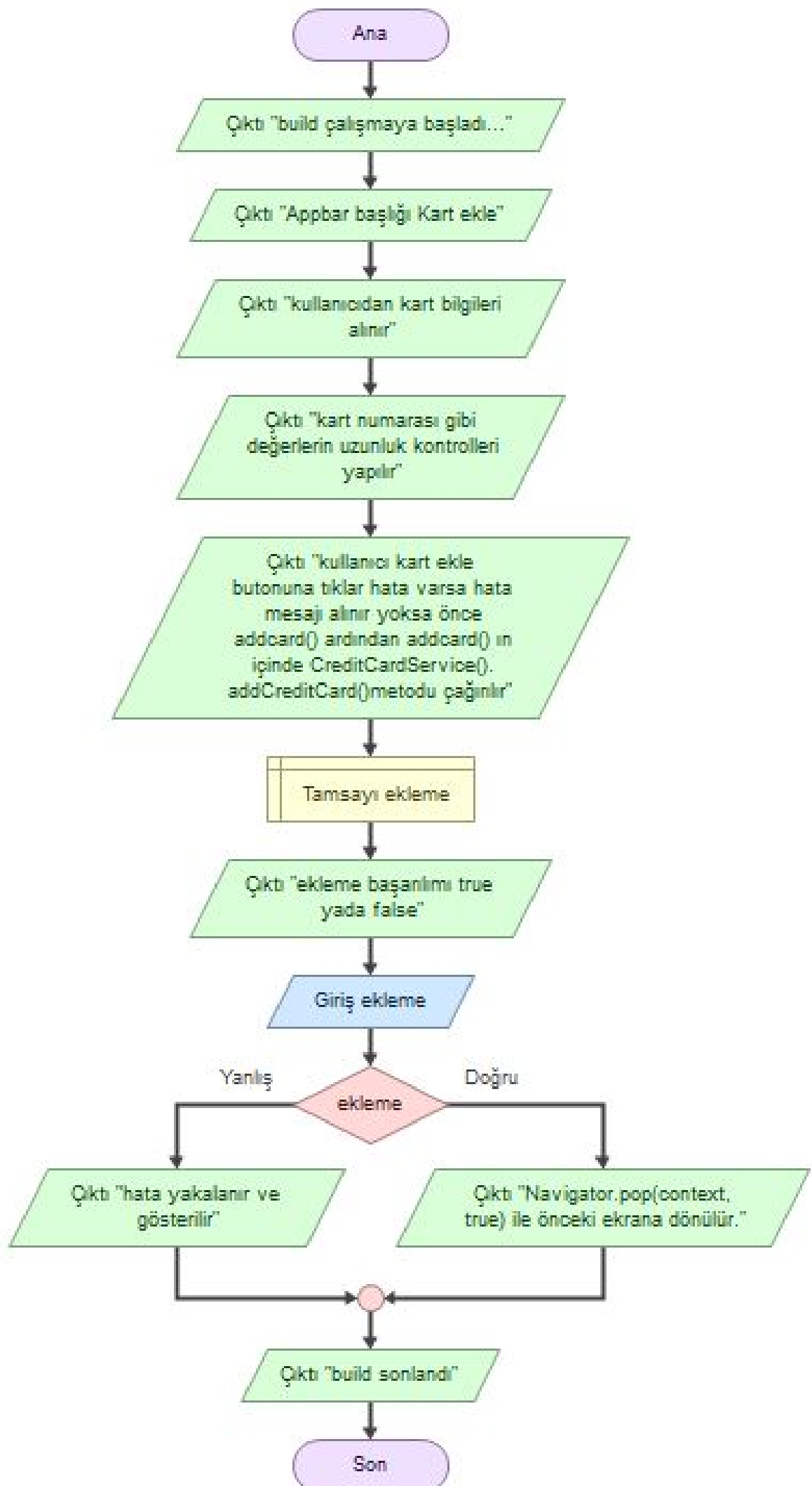
B. Dosya Yapısı

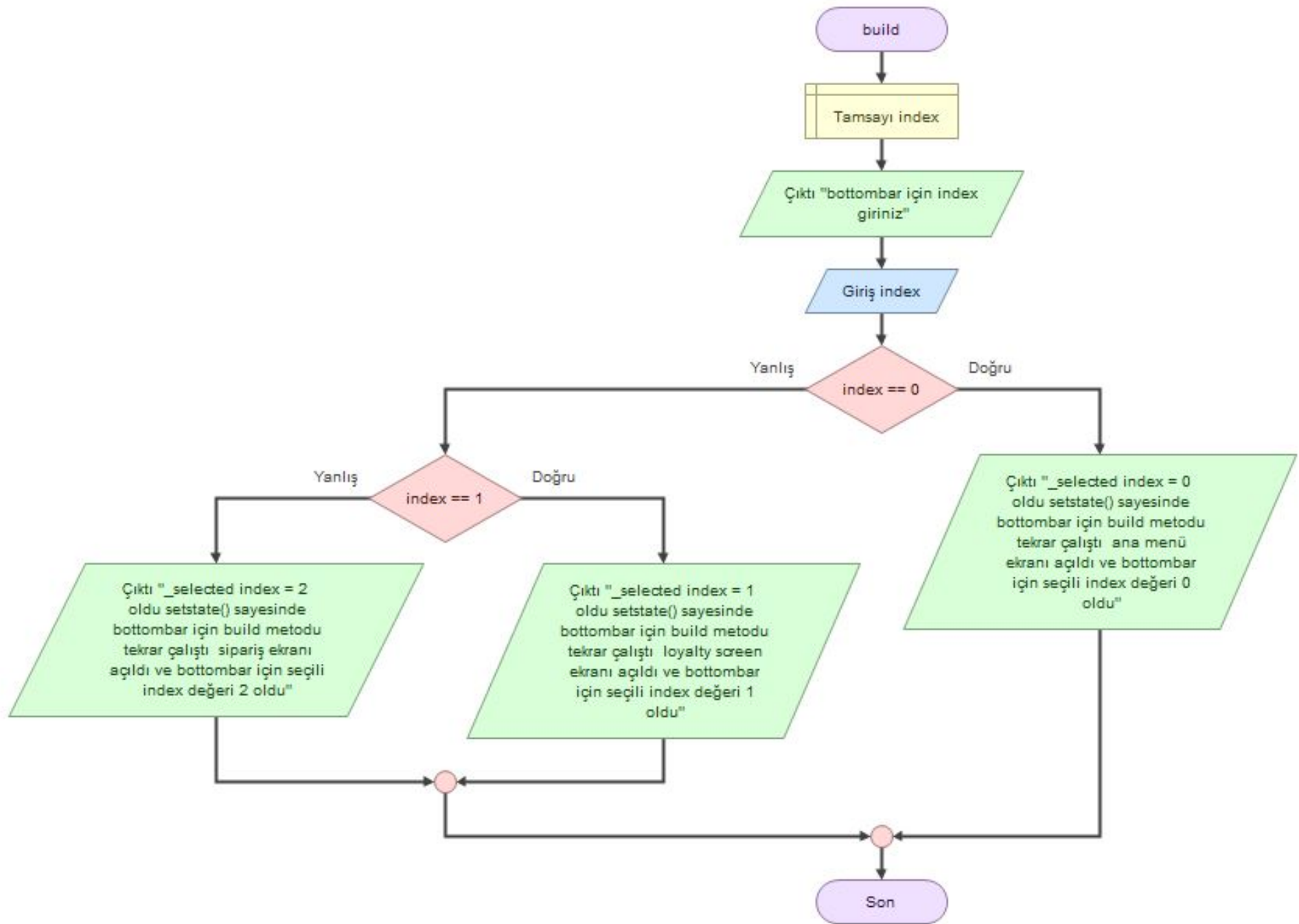


IV. Kaynakça

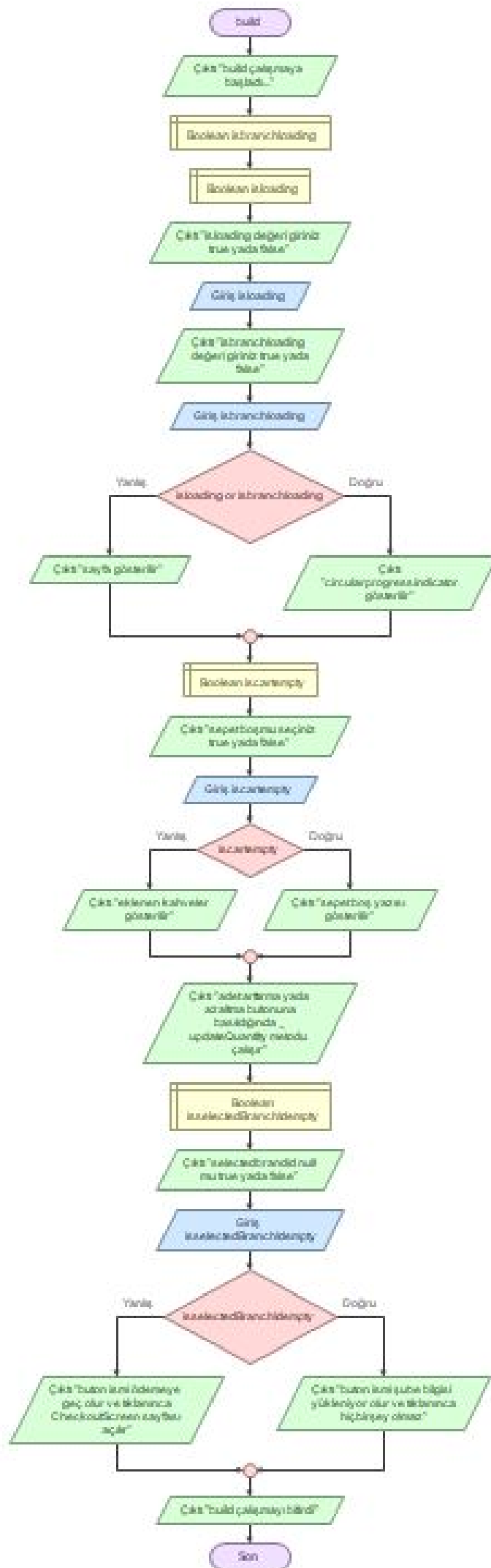
- [1] state management dökümanı : <https://docs.flutter.dev/data-and-backend/state-mgmt/simple>
- [2] future yapısı dökümanı : <https://api.flutter.dev/flutter/dart-async/Future-class.html>
- [3] http dökümanı : <https://pub.dev/packages/http>
- [4] flutter dökümanı : <https://docs.flutter.dev/>
- [5] mysql dökümanı : <https://dev.mysql.com/doc/>
- [6] node.js dökümanı : <https://nodejs.org/docs/latest/api/>

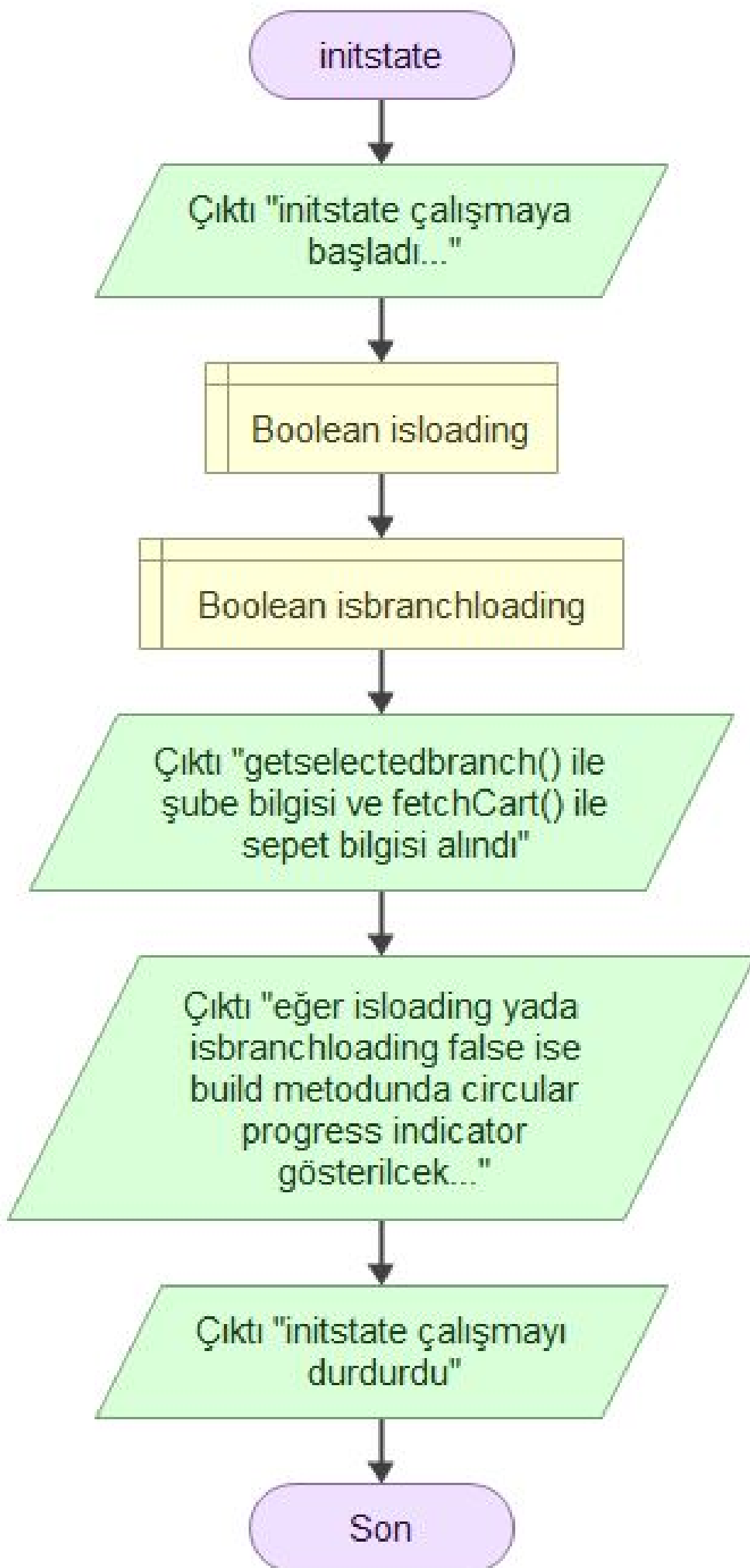


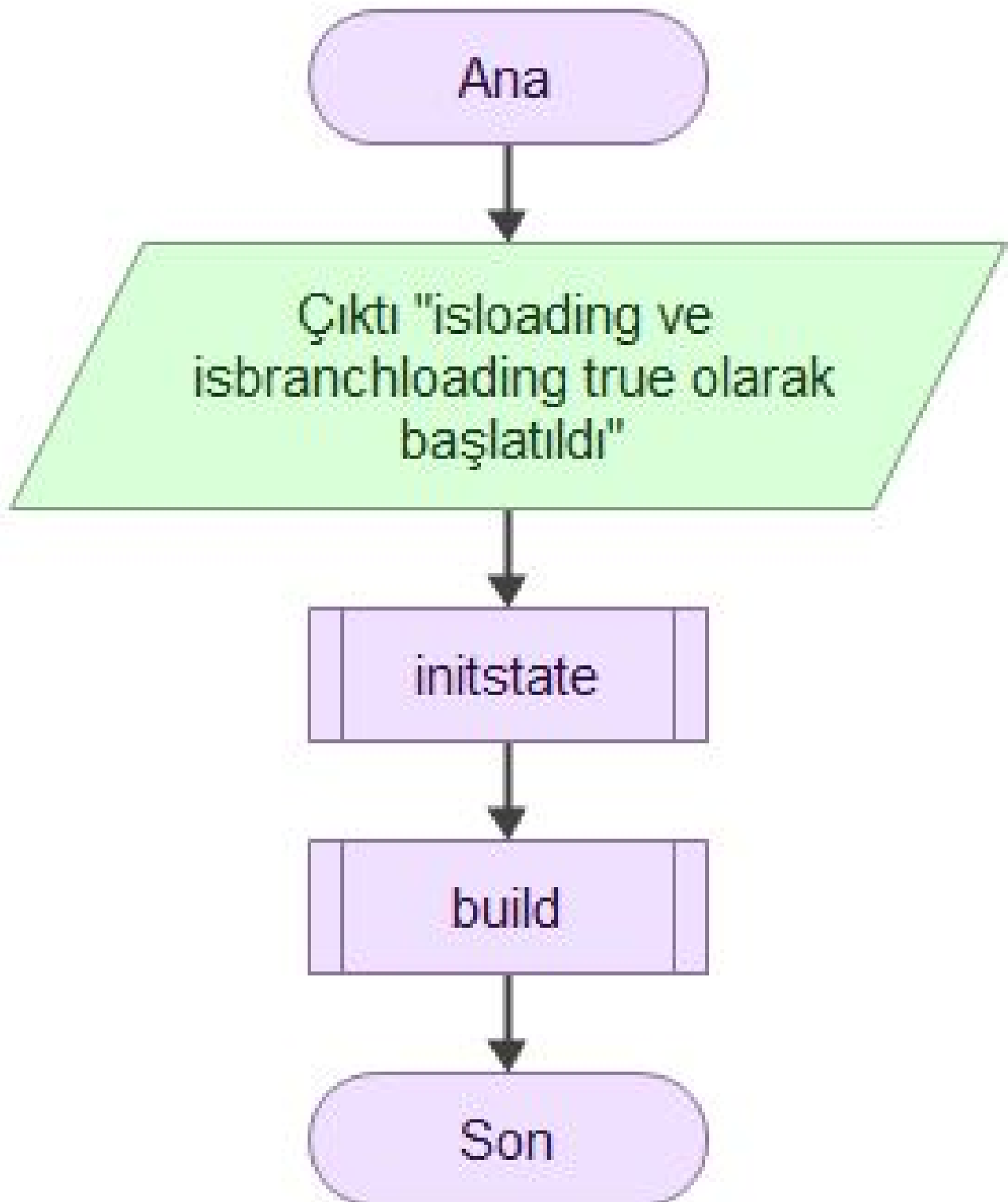


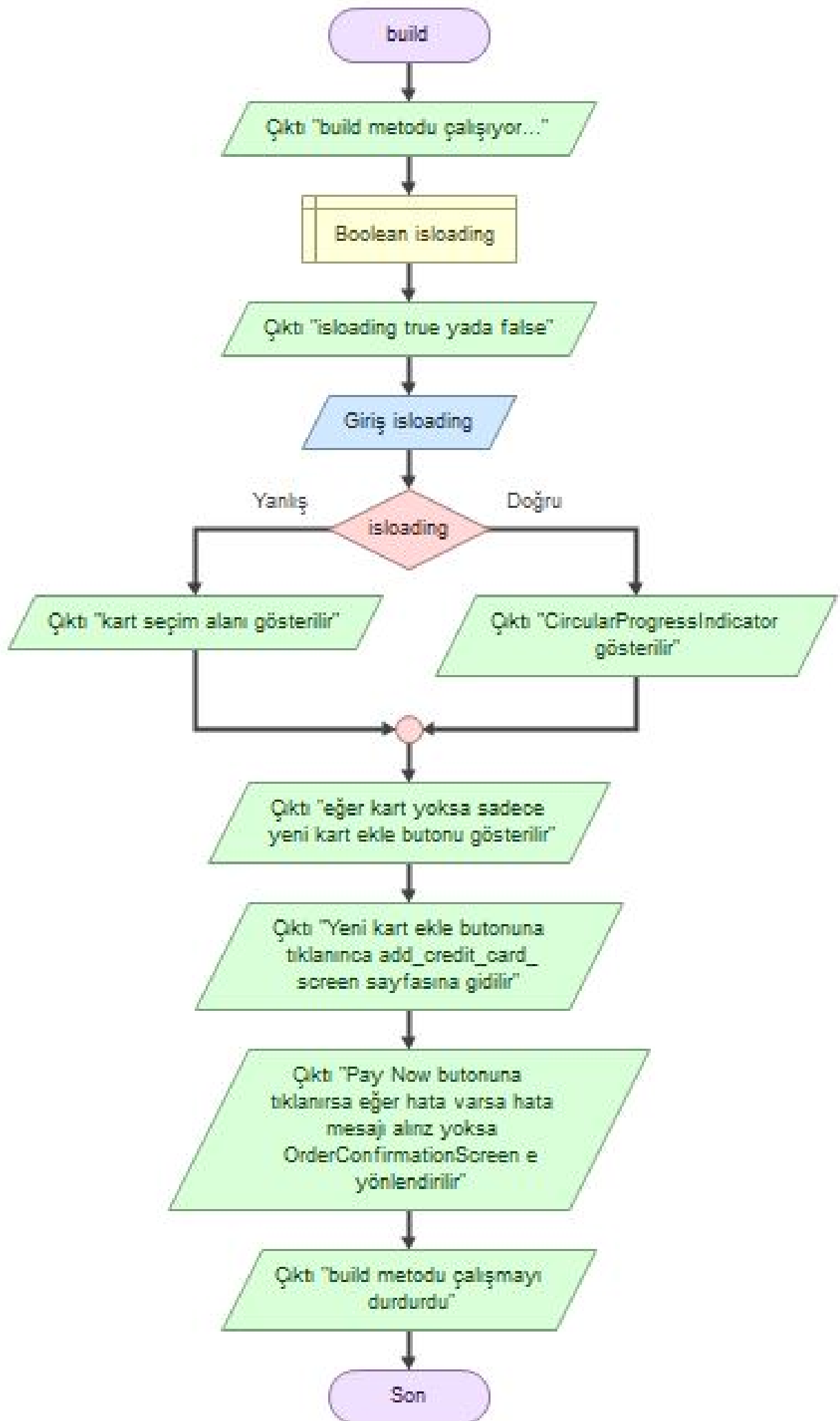


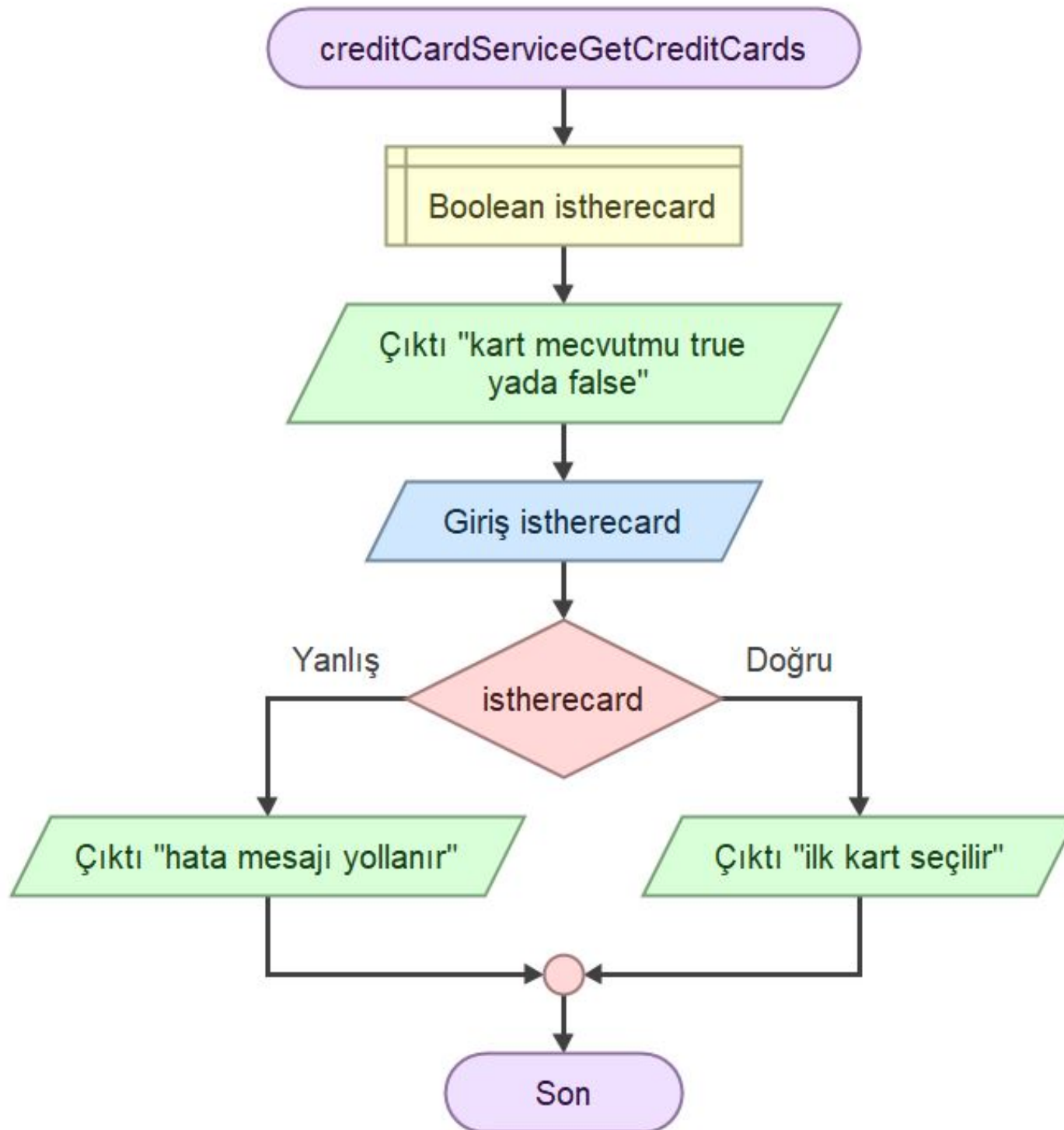


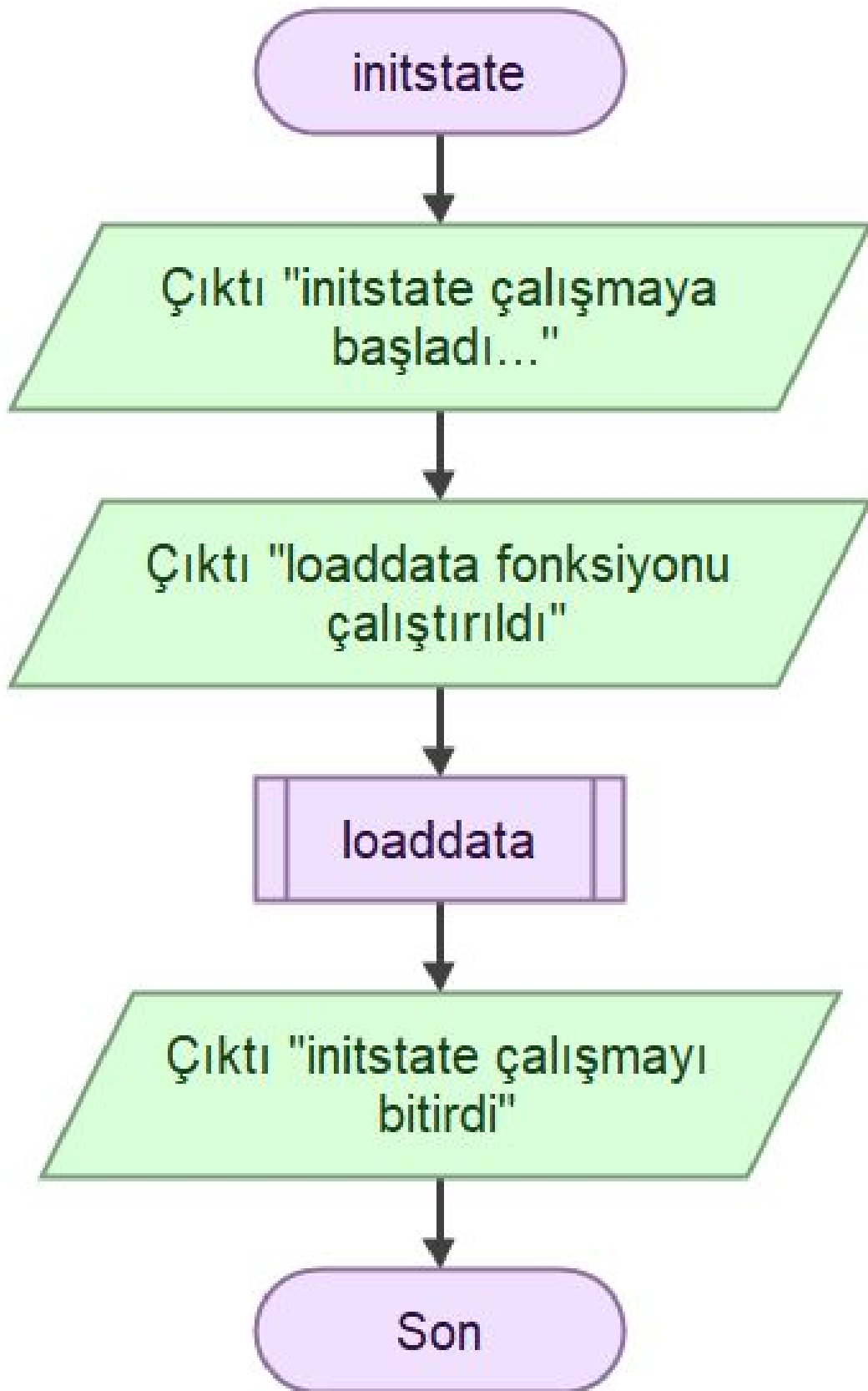












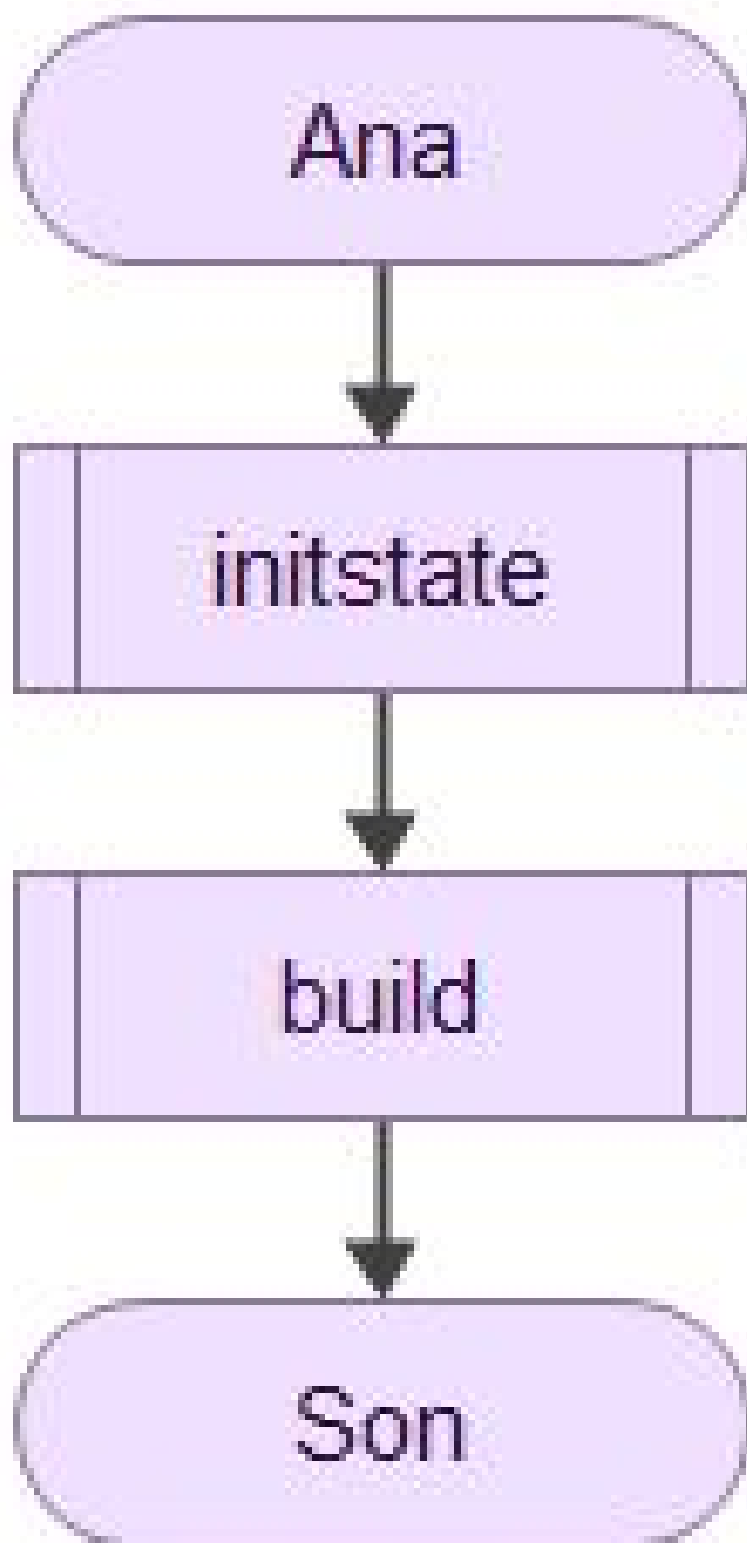
loaddata



creditCardServiceGetCreditCards



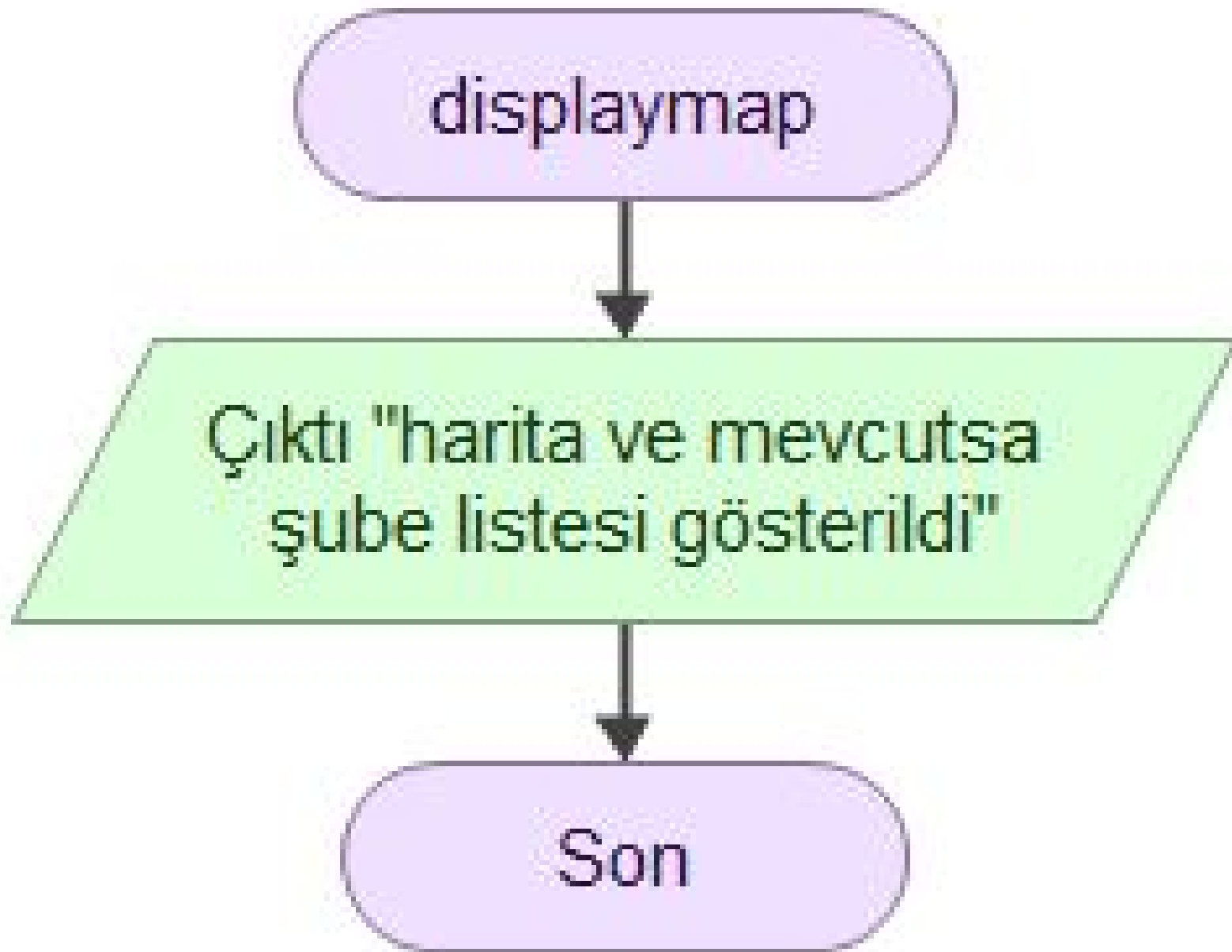
Son



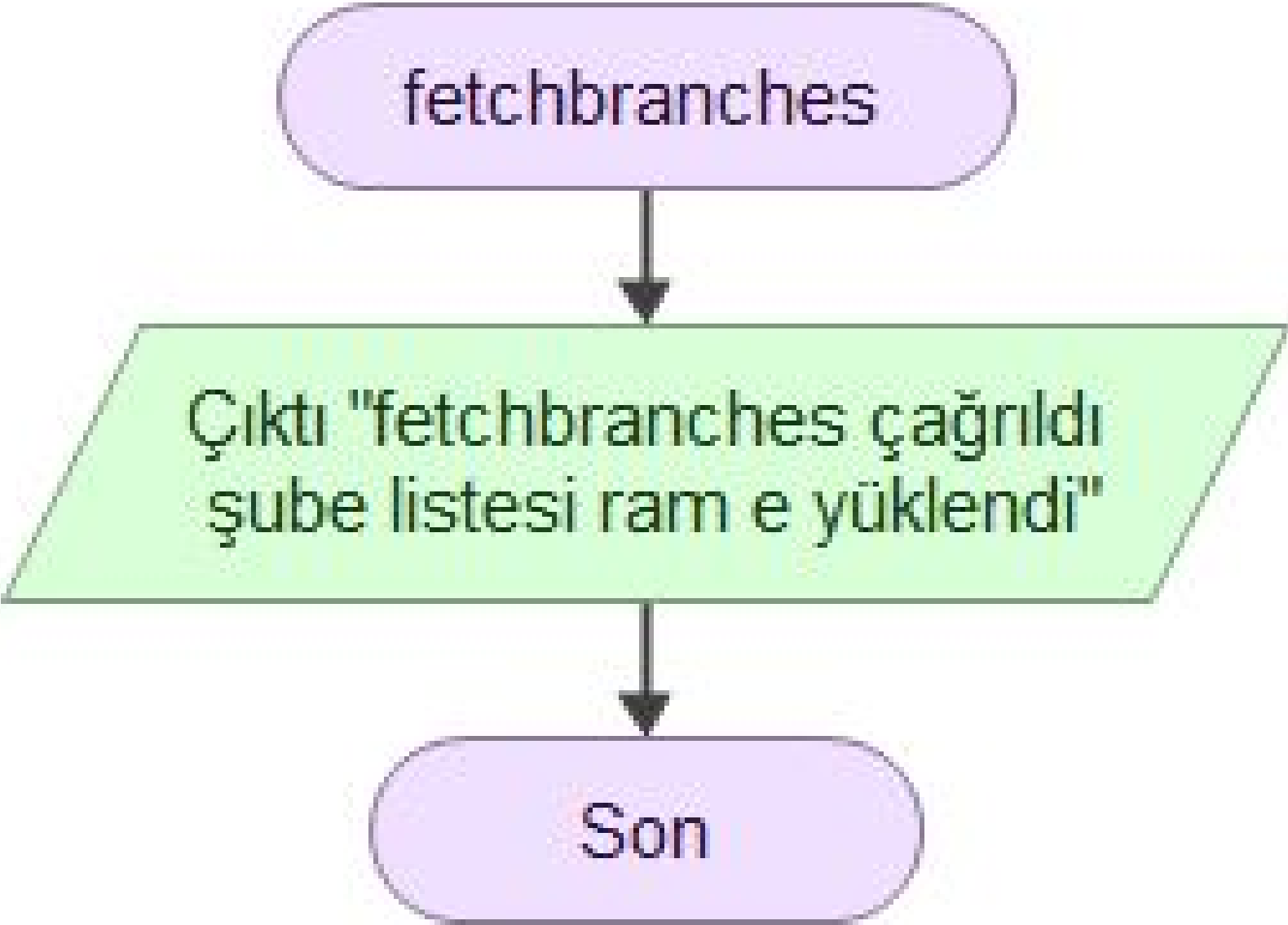
displaymap

Çıktı "harita ve mevcutsa
şube listesi gösterildi"

Son



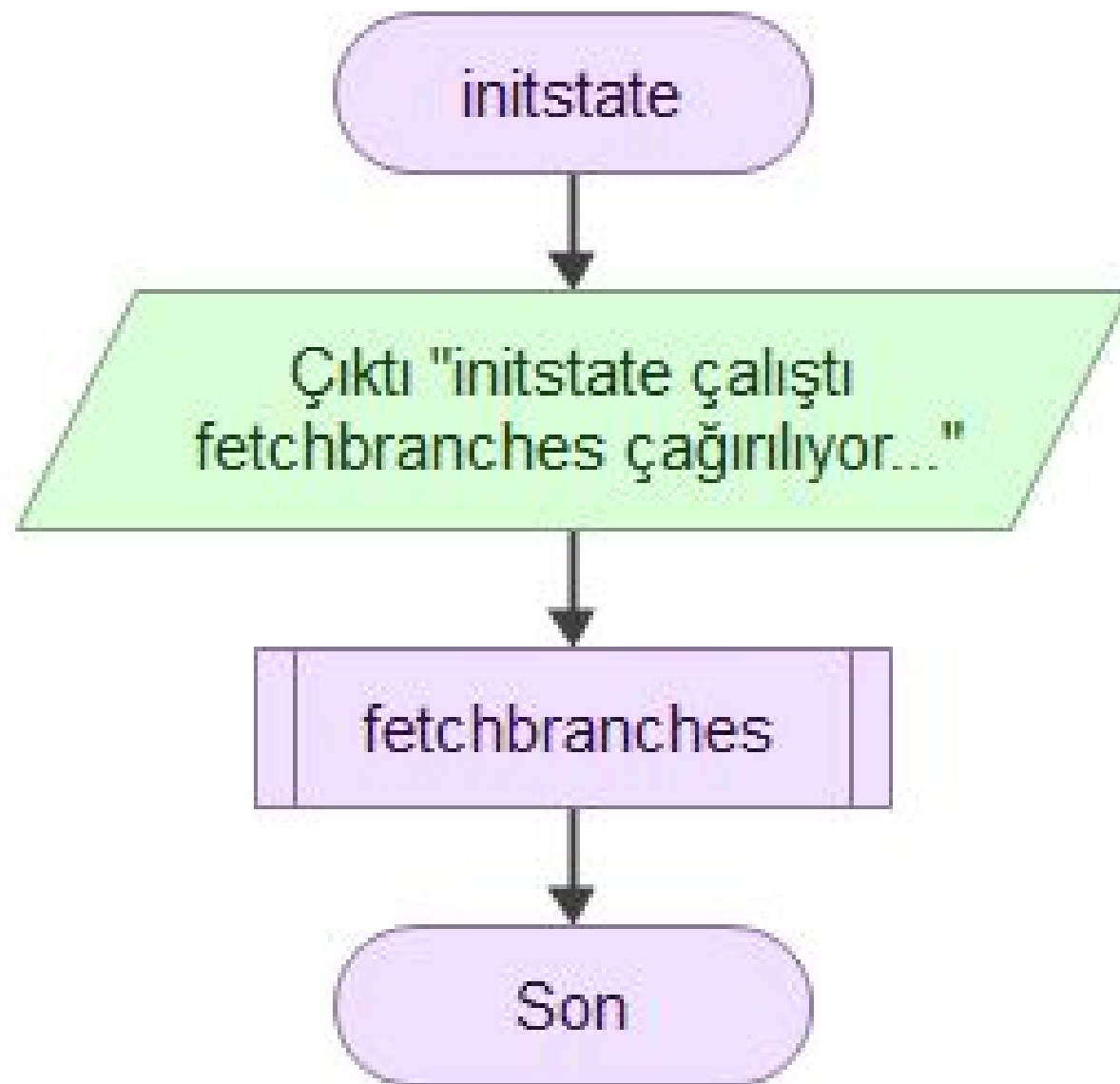
fetchbranches



```
graph TD; A(fetchbranches) --> B[/Çıktı "fetchbranches çağrıldı şube listesi ram e yüklendi"/]; B --> C(Son);
```

Çıktı "fetchbranches çağrıldı
şube listesi ram e yüklendi"

Son



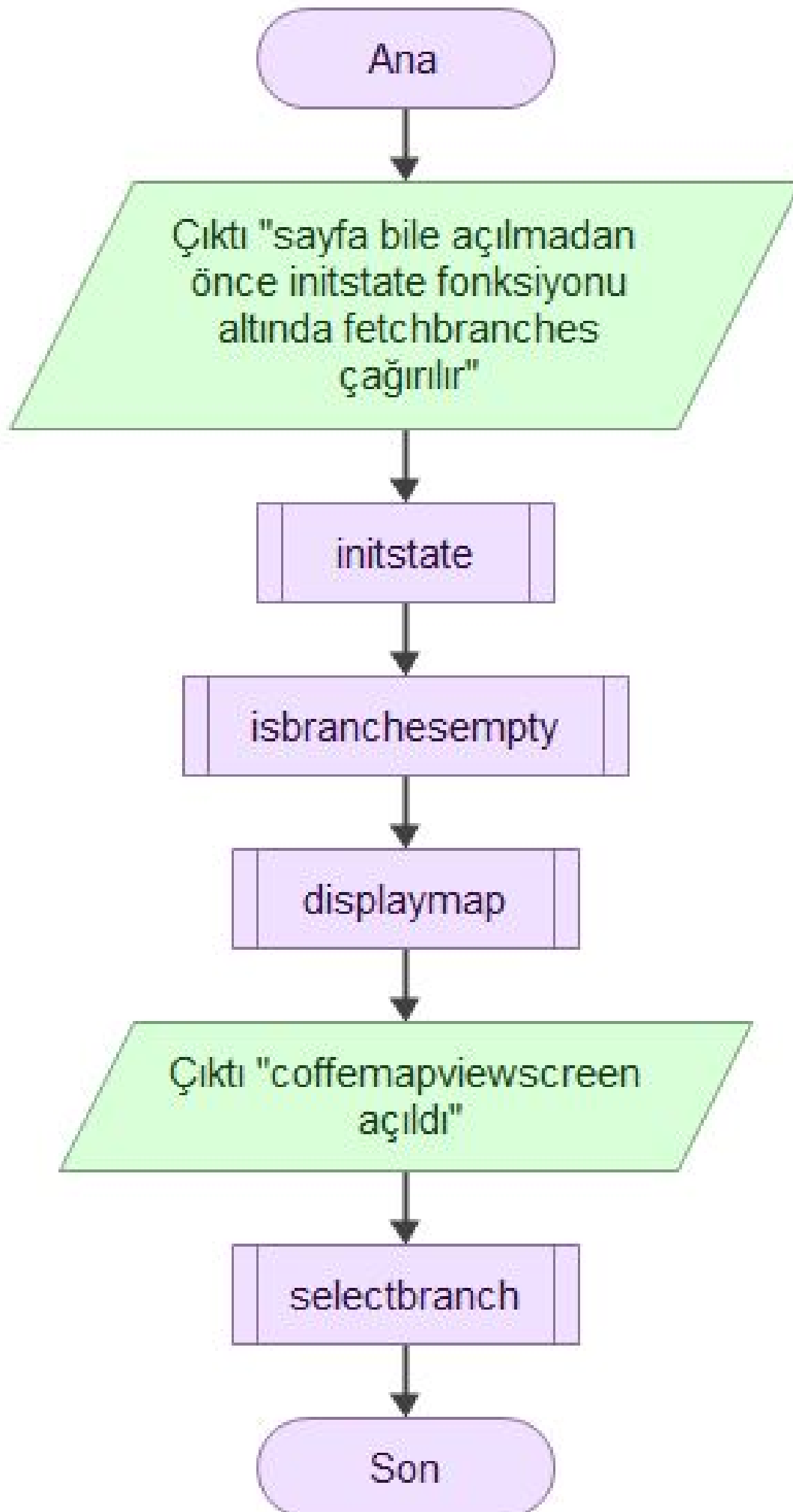
isbranchesempty



Çıktı "fetchbranches
fonksiyonunu dinleyen
consumer yapısı initState
metodunun altındaki
fetchbranches metodu
çağırıldığı için çalıştı eğer hiç
şube yoksa
circularprogressindicator
(yükleme ekranı gösteren
widget) gösterdi şube
varsada şubeleri listeledi"



Son



selectbranch

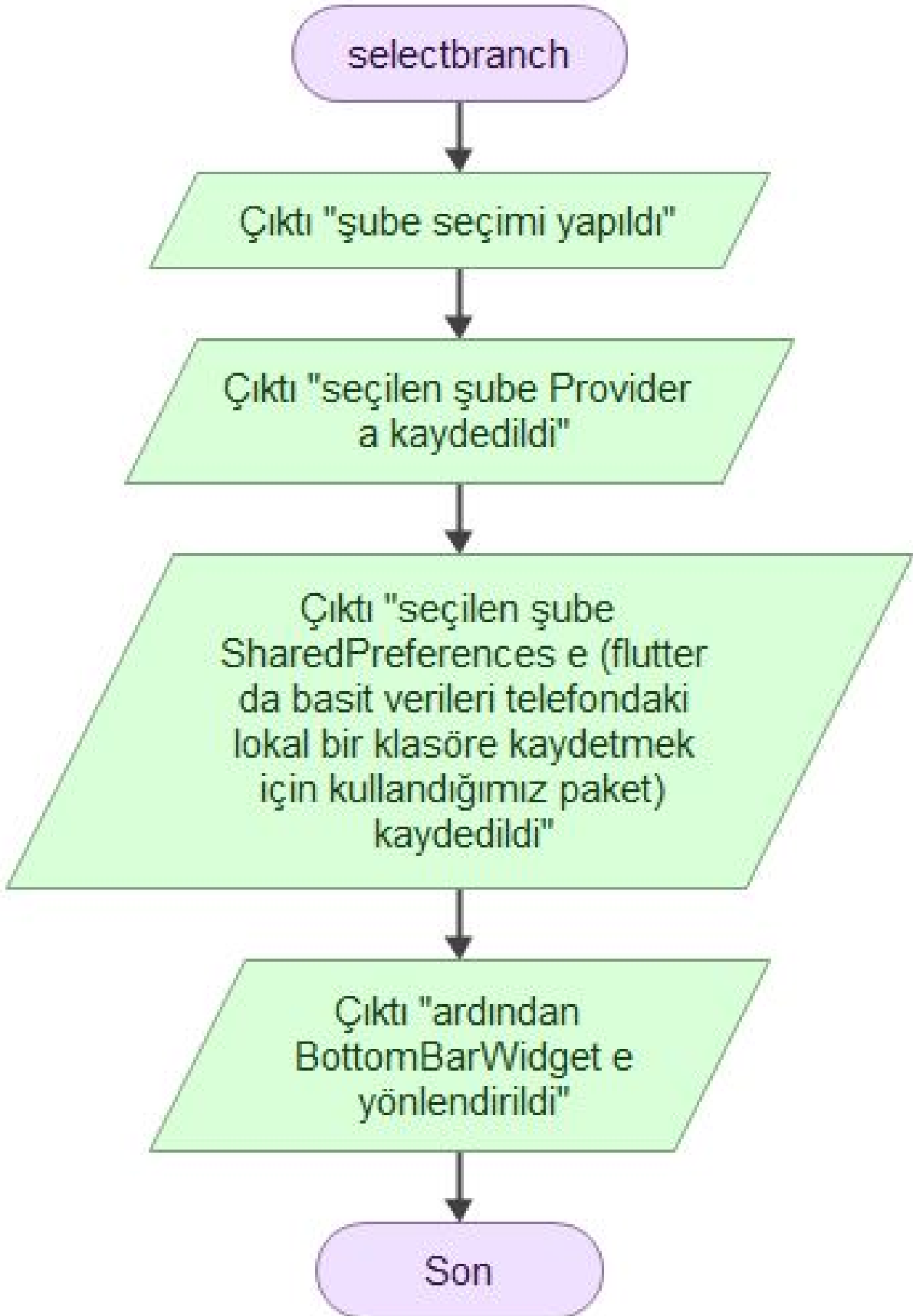
Çıktı "şube seçimi yapıldı"

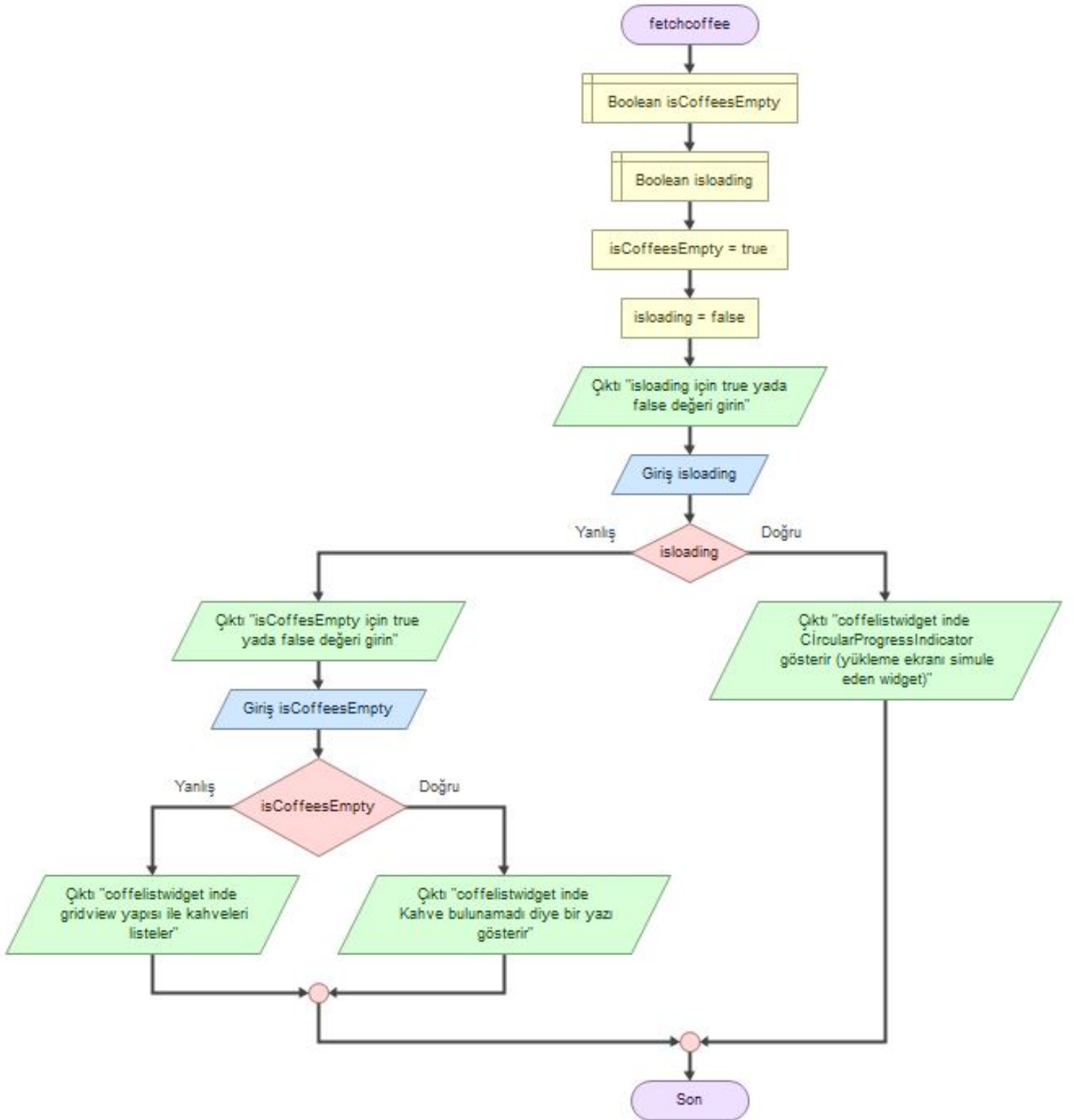
Çıktı "seçilen şube Provider
a kaydedildi"

Çıktı "seçilen şube
SharedPreferences e (flutter
da basit verileri telefondaki
lokal bir klasöre kaydetmek
için kullandığımız paket)
kaydedildi"

Çıktı "ardından
BottomBarWidget e
yönlendirildi"

Son







cofeelistwidget



Çıktı "cofeelistwidget akış
şeması"



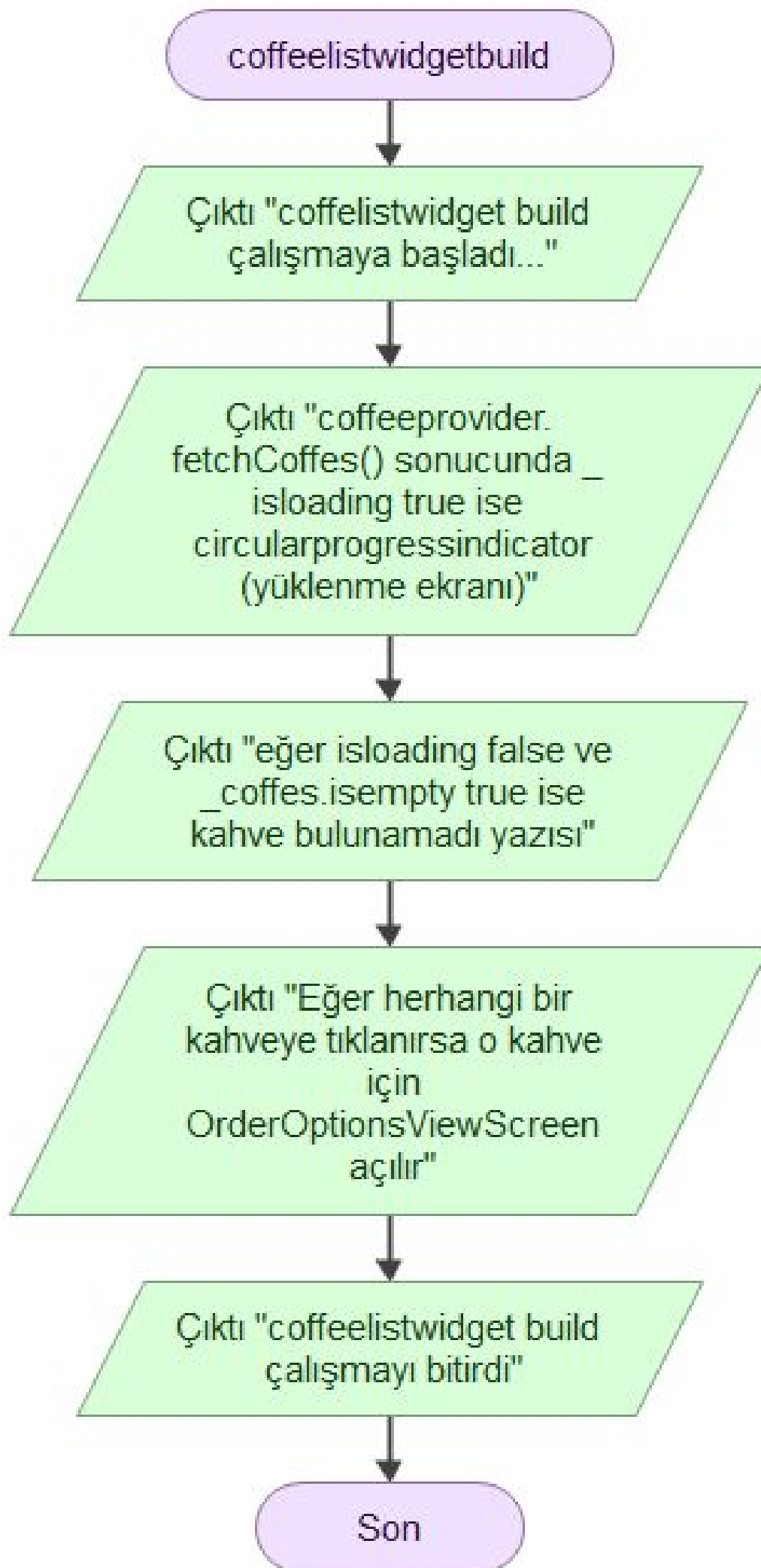
cofeelistwidgetinitstate

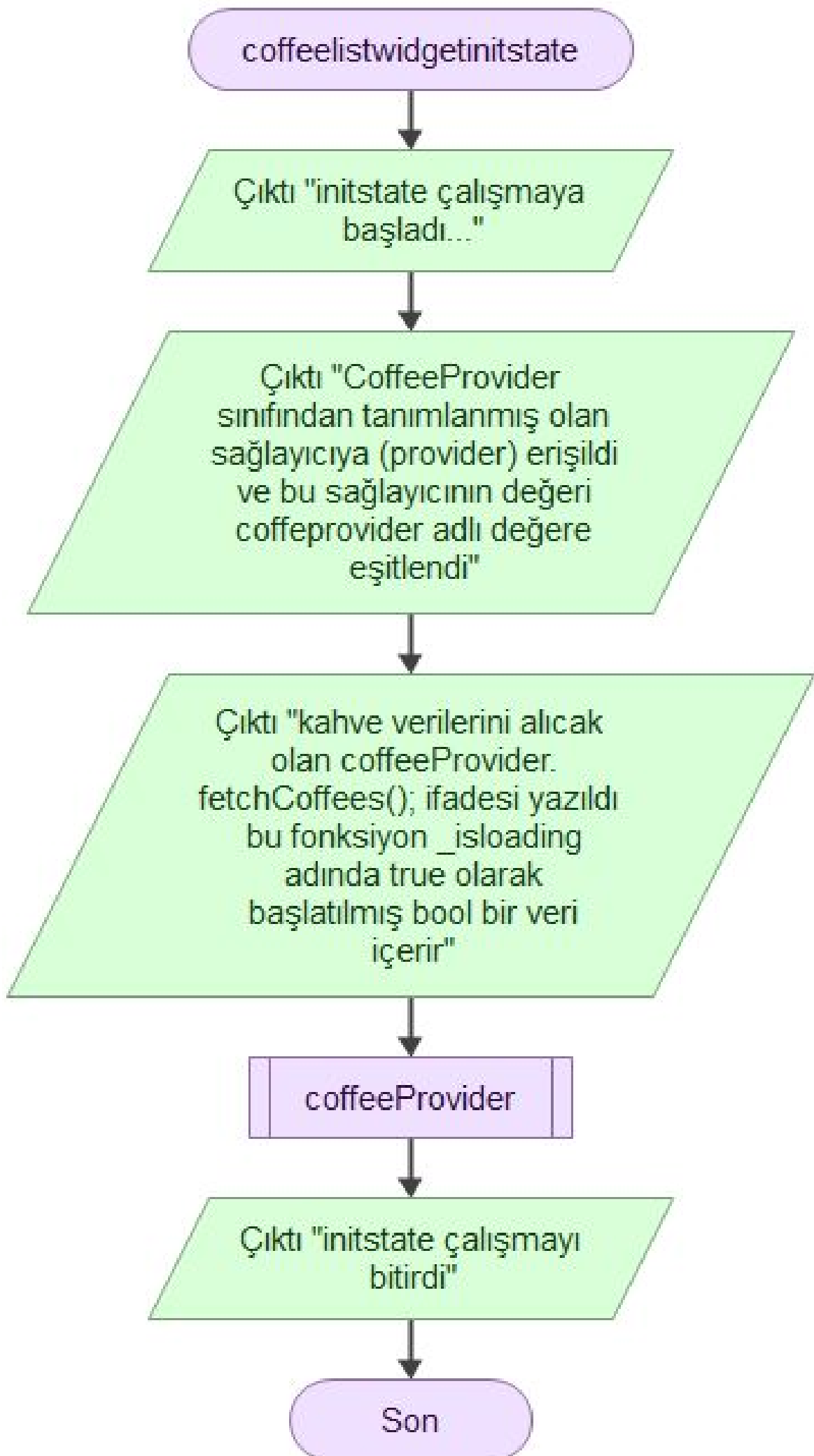


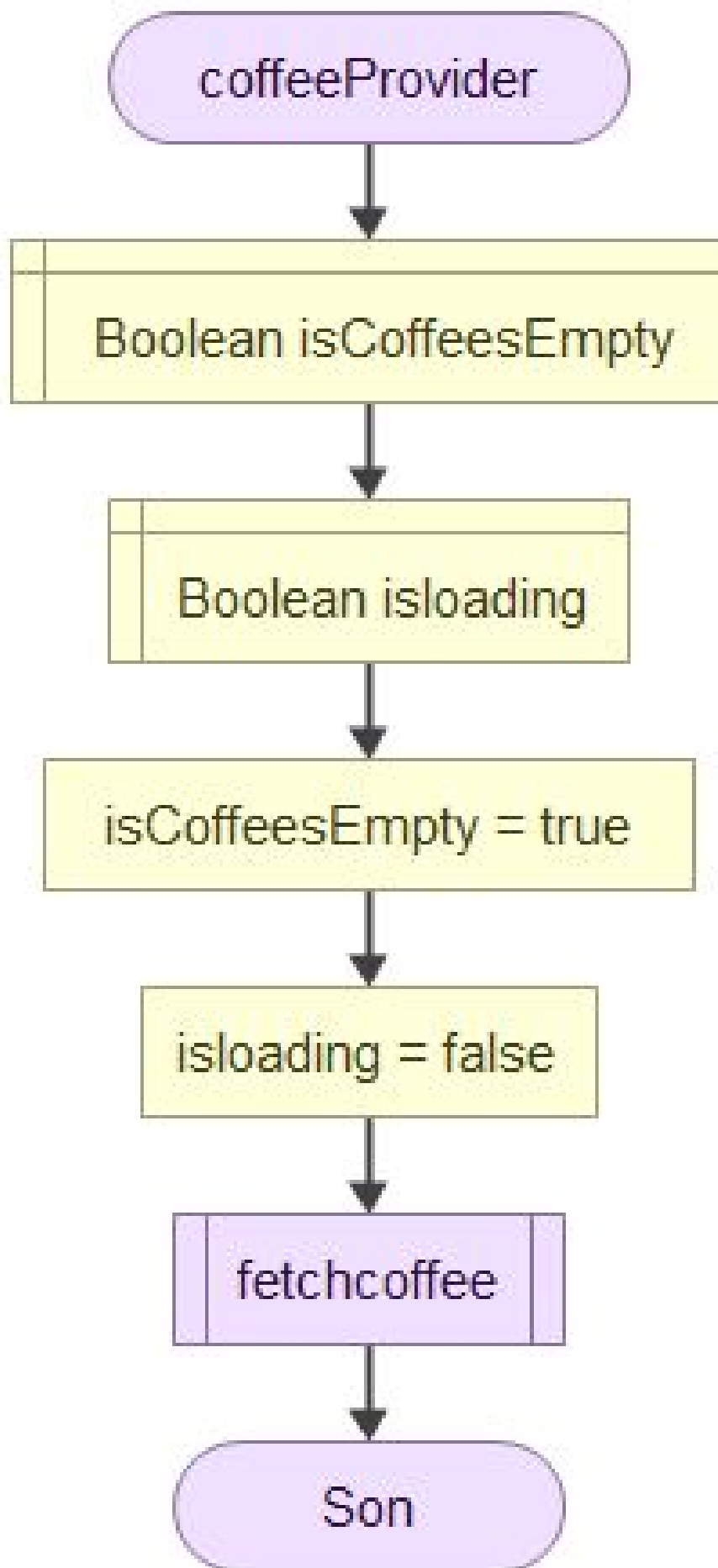
cofeelistwidgetbuild

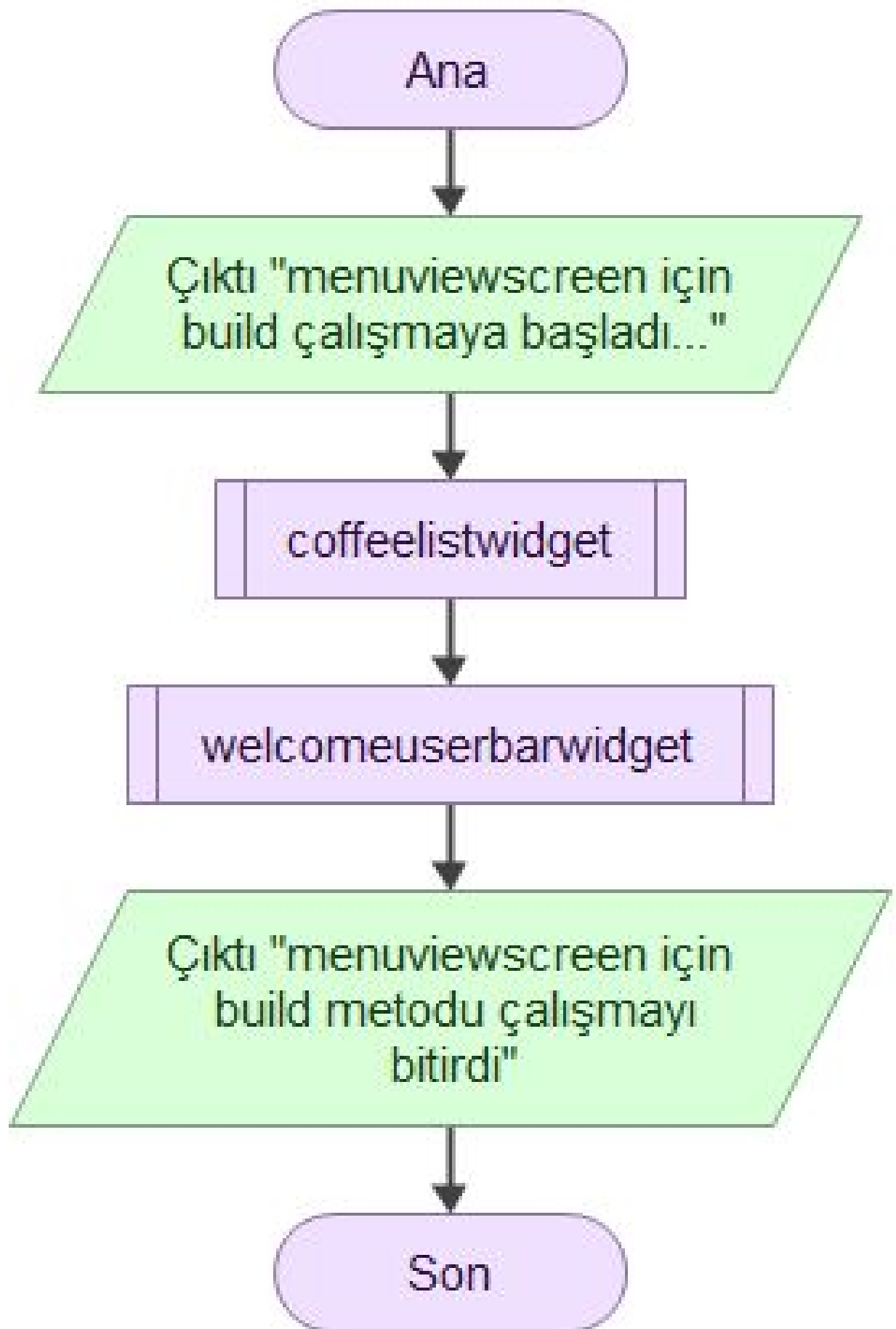


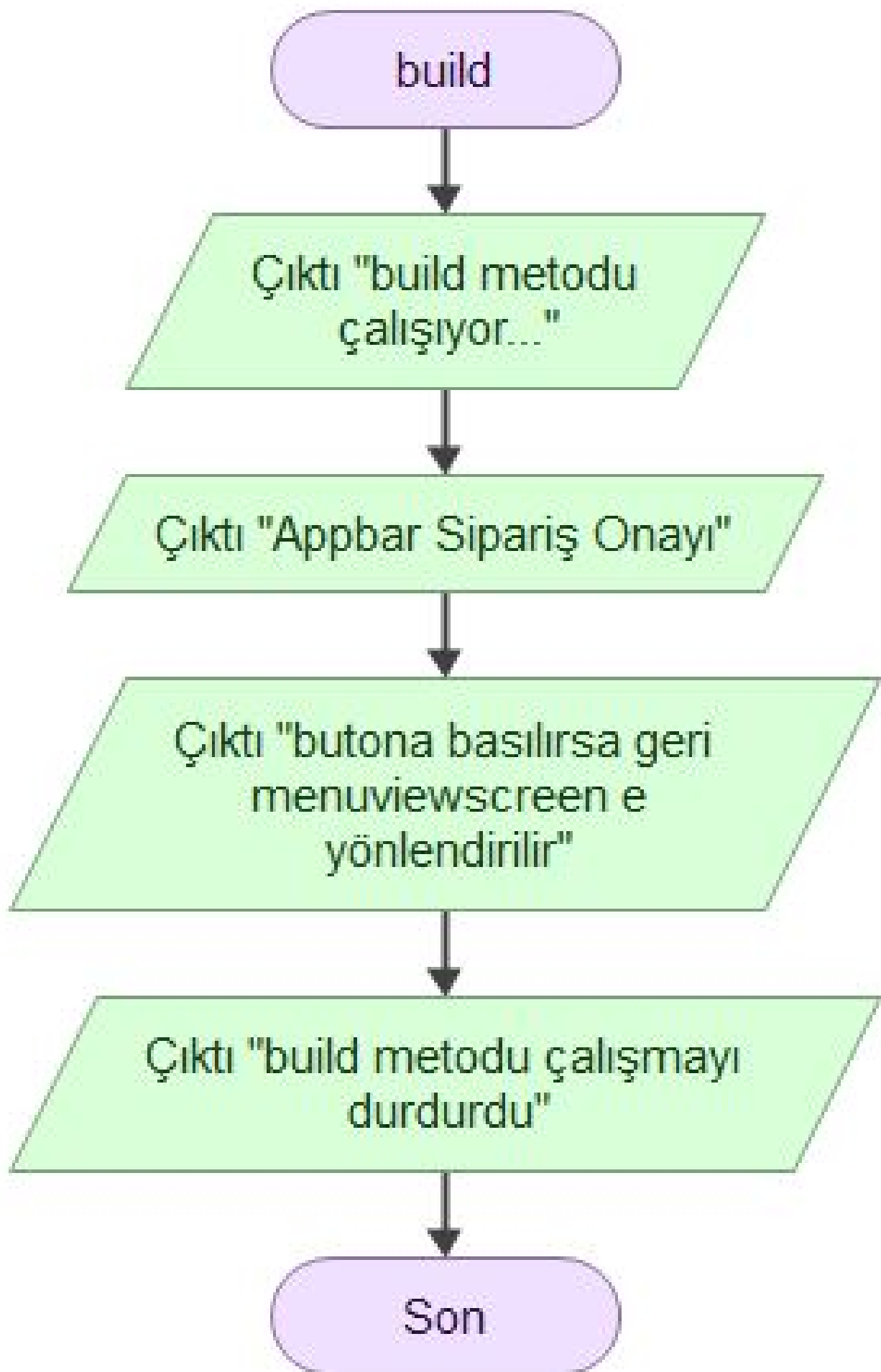
Son

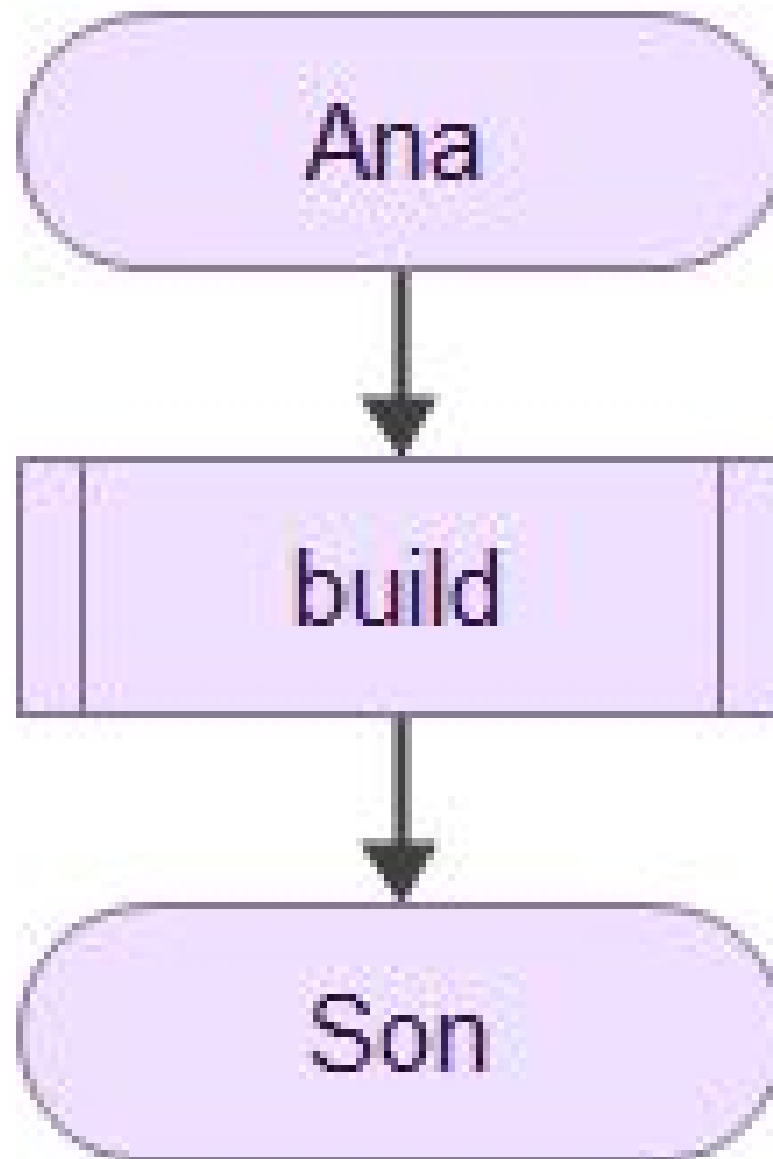












addToCart



```
graph TD; A([addToCart]) --> B[/Çıktı "seçilen değerler bir cartItem nesnesine kaydedilir"/]; B --> C[/Çıktı "ardından Cartservice(). addItem fonksiyonu ile eklenen sipariş veritabanına yansıtılır"/]; C --> D[/Çıktı "cartservice sayfasına gidilir"/]; D --> E([Son]);
```

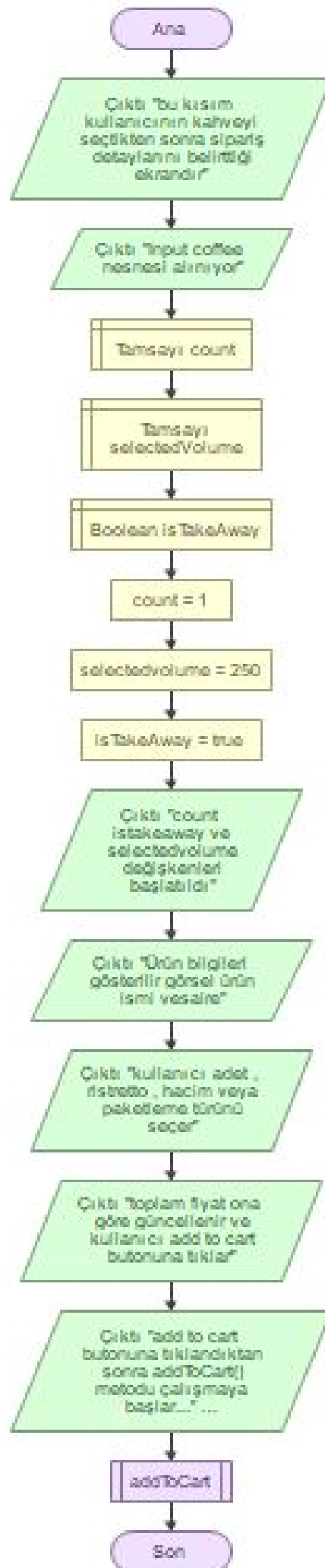
The flowchart illustrates the process of adding an item to a cart. It begins with a start node labeled 'addToCart'. This leads to a process block where the selected values are stored in a 'cartItem' object. The next step is another process block where the 'addItem' function of the 'Cartservice()' is used to save the order to the database. This is followed by a third process block indicating navigation to the 'cartservice' page. The process concludes at an end node labeled 'Son'.

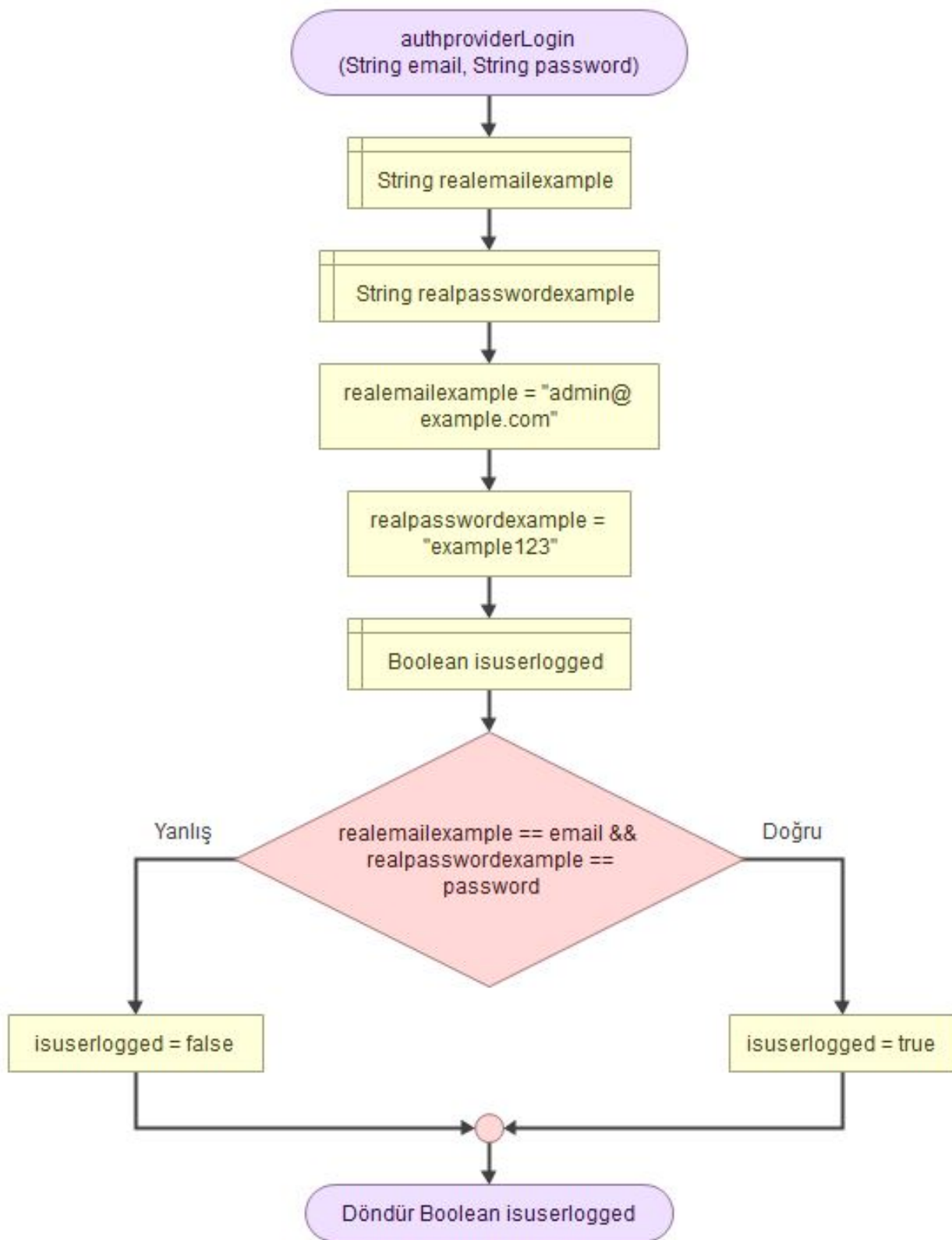
Çıktı "seçilen değerler bir
cartItem nesnesine
kaydedilir"

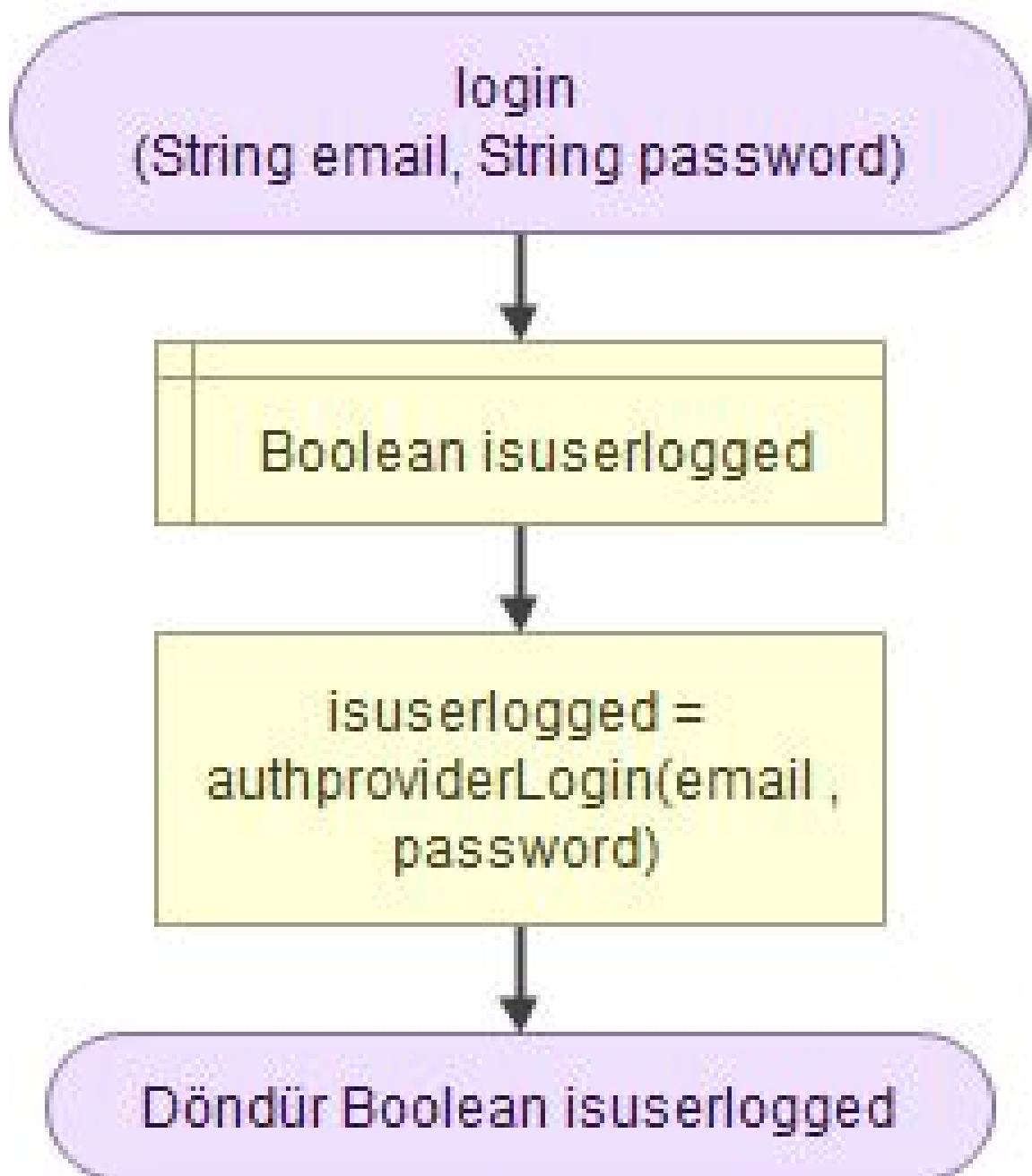
Çıktı "ardından Cartservice().
addItem fonksiyonu ile
eklenen sipariş veritabanına
yansıtılır"

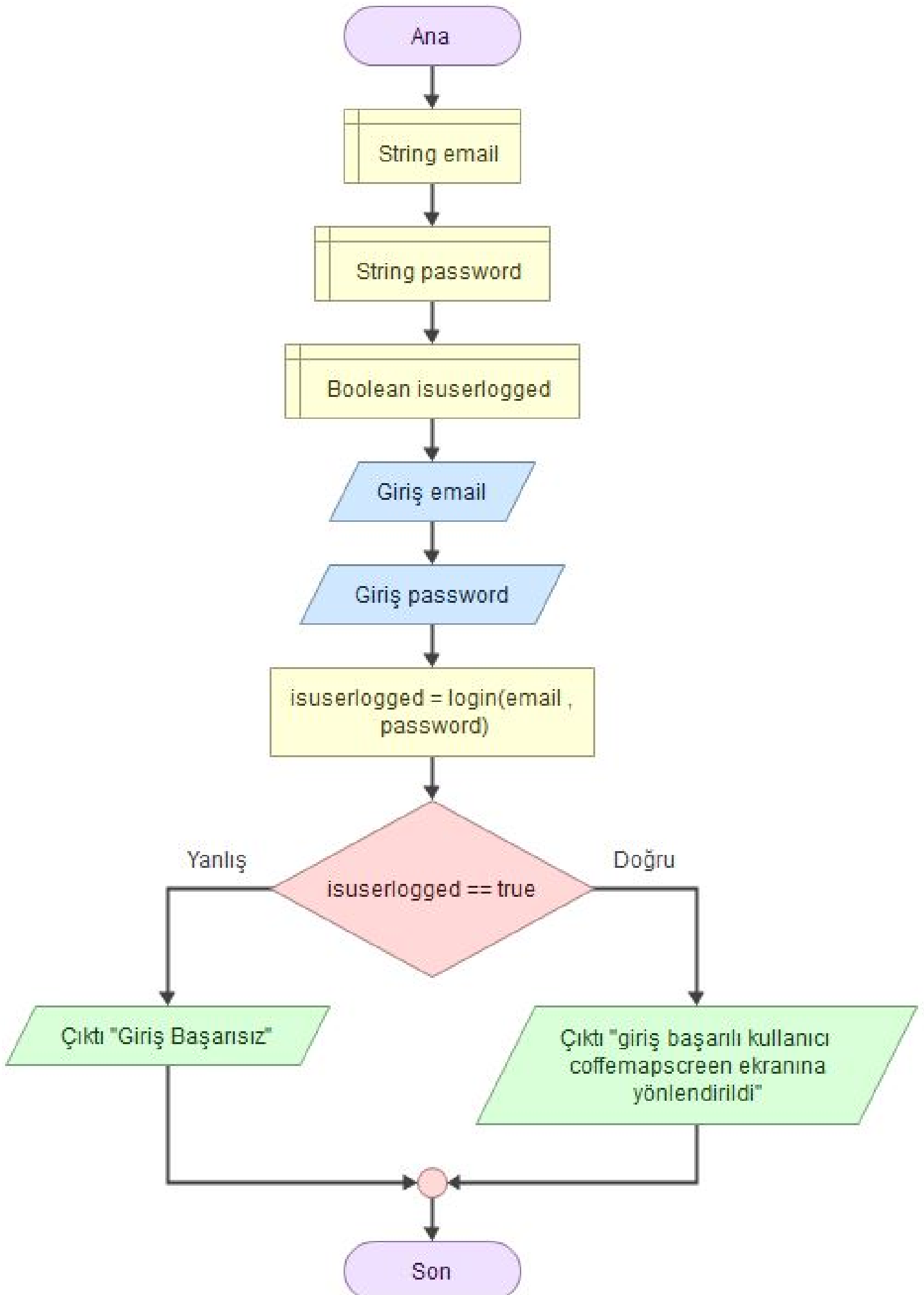
Çıktı "cartservice sayfasına
gidilir"

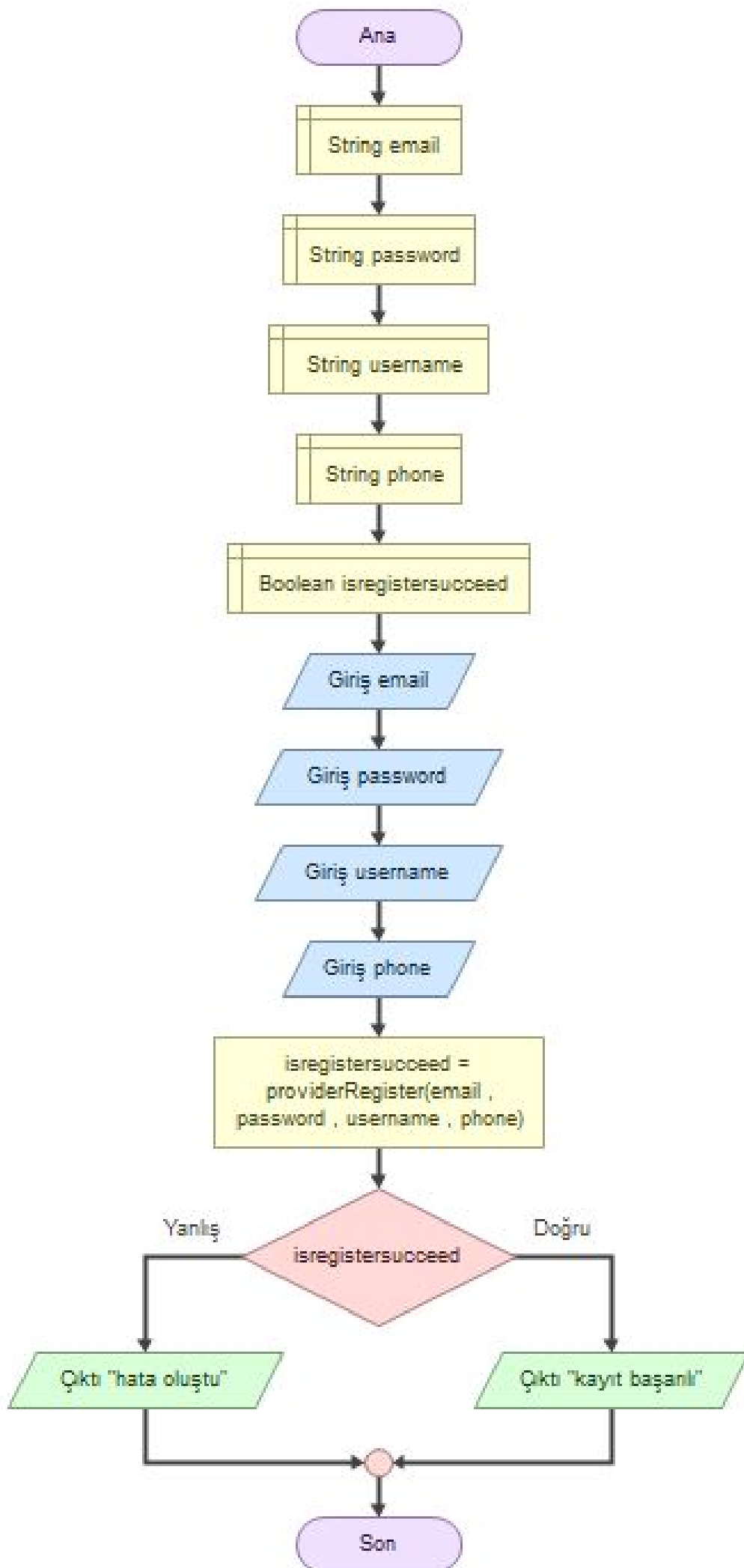
Son











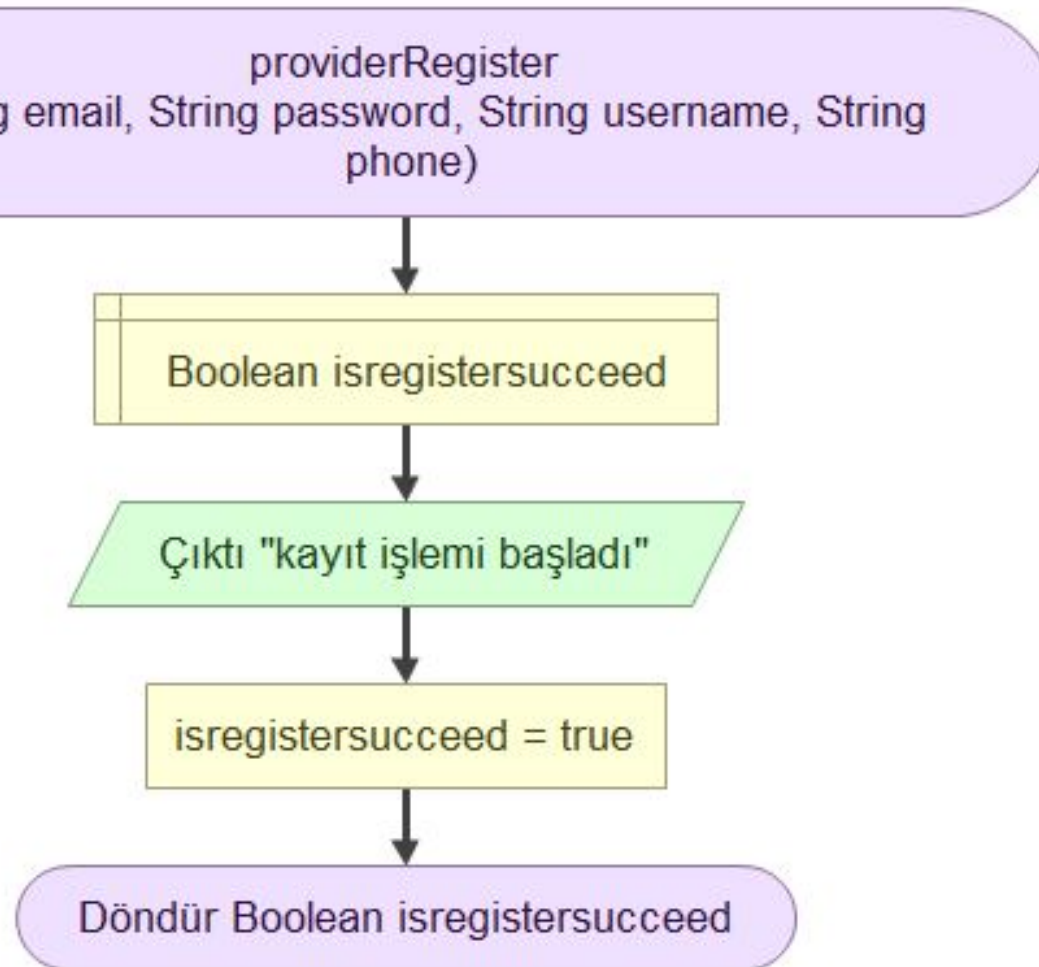
providerRegister
(String email, String password, String username, String
phone)

Boolean isregistersucceed

Çıktı "kayıt işlemi başladı"

isregistersucceed = true

Döndür Boolean isregistersucceed



register
(String email, String password, String username, String
phone)



```
graph TD; A([register  
(String email, String password, String username, String  
phone)]) --> B[ ]; B --> C[isregistersucceed =  
providerRegister(email ,  
password , username ,  
phone)]; C --> D([Döndür Boolean isregistersucceed]);
```

String isregistersucceed

isregistersucceed =
providerRegister(email ,
password , username ,
phone)

Döndür Boolean isregistersucceed