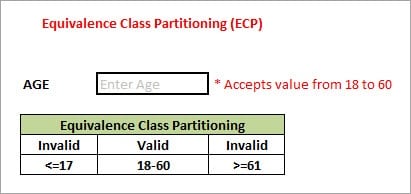
**4. Bölüm test örnekleri**

**Denklik Paylarına Ayırma**

**Amaç:** Benzer özelliklere sahip girdileri gruplandırarak test sayısını azaltmak.

**Örnek 1:** Bir kullanıcı kayıt formunda, yaş alanının 18-60 arasında geçerli olduğu belirtilsin.

* **Denklik Payları:**
  + Geçersiz (0 veya negatif)
  + Geçerli (18-60)
  + Geçersiz (60'tan büyük)
* **Test Senaryoları:**
  + Yaş olarak 0 gir.
  + Yaş olarak 18 gir.
  + Yaş olarak 50 gir.
  + Yaş olarak 60 gir.
  + Yaş olarak 11 gir.

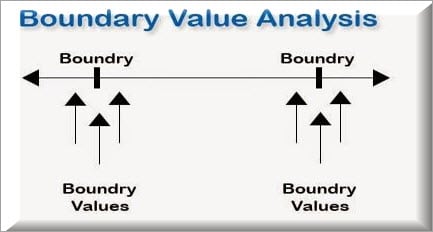


**Örnek 2:**

* **Bir parola doğrulama sisteminde:**
  + **Denklik Payları:**
    - Geçerli (8-15 karakter arasında, büyük/küçük harf, rakam ve özel karakter içeren)
    - Çok kısa (< 8 karakter)
    - Çok uzun (> 15 karakter)
    - Sadece harflerden oluşan
    - Sadece rakamlardan oluşan
    - Sadece özel karakterlerden oluşan
    - Boş
  + **Test Senaryoları:**
    - "12345678" gibi geçerli bir parola gir.
    - "a" gibi çok kısa bir parola gir.
    - "abcdefghijklmno" gibi çok uzun bir parola gir.
    - "password123" gibi harf ve rakam içeren bir parola gir.
    - "!@#$%^&\*" gibi sadece özel karakter içeren bir parola gir.
    - Boş bir alan bırak.

**Sınır Değer Analizi**

**Amaç:** Denklik paylarının sınır değerlerinde hataların daha sık ortaya çıkma olasılığından yararlanarak test etmek.



**Örnek 1:** Yukarıdaki yaş örneğini ele alalım.

* **Sınır Değerler:** 17, 18, 100, 101
* **Test Senaryoları:**
  + Yaş olarak 17 gir.
  + Yaş olarak 18 gir.
  + Yaş olarak 99 gir.
  + Yaş olarak 100 gir.
  + Yaş olarak 101 gir.

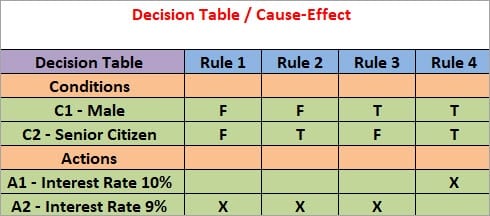
**Örnek 2:**

**Bir online mağazada ürün miktarı belirleme alanında:**

* **Sınır Değerler:** 0, 1, maksimum sipariş miktarı, maksimum sipariş miktarı + 1
* **Test Senaryoları:**
  + 0 ürün ekle.
  + 1 ürün ekle.
  + Maksimum sipariş miktarı kadar ürün ekle.
  + Maksimum sipariş miktarından 1 fazla ürün eklemeyi dene.
  + Negatif bir değer gir.

**Karar Tablosu Testi**

**Amaç:** Birden fazla koşulun bir araya gelmesiyle oluşan farklı durumları test etmek.



**Örnek 1:** Bir e-ticaret sitesinde, ürünün fiyatı 100 TL'den fazlaysa ve üyelik süresi 1 yıldan fazlaysa %10 indirim uygulanıyor.

* **Koşullar:**
  + Fiyat > 100 TL
  + Üyelik Süresi > 1 yıl
* **Sonuçlar:**
  + İndirim uygulanır
  + İndirim uygulanmaz

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek 2:

**Bir banka uygulamasında kredi kartı başvurusu:**

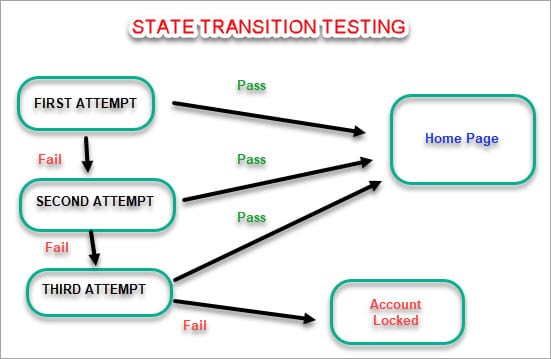
* **Koşullar:**
  + Yaş > 18
  + Gelir > asgari ücret
  + Kredi notu > belirli bir değer
* **Sonuçlar:**
  + Kredi kartı onaylanır
  + Kredi kartı reddedilir
* **Karar Tablosu:**

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Durum Geçişi Testi**

**Amaç:** Sistemin bir durumdan diğerine geçişlerini test etmek.



**Örnek 1:** Bir trafik lambasının durumları (kırmızı, sarı, yeşil).

* **Durumlar:** Kırmızı, Sarı, Yeşil
* **Geçişler:**
  + Kırmızıdan Sarıya
  + Sarıdan Yeşile
  + Yeşilden Kırmızıya
* **Test Senaryoları:**
  + Lambanın kırmızı iken sarıya dönmesini bekle.
  + Lambanın yeşil iken kırmızıya dönmesini bekle.

**Örnek 2:**

**Bir müzik çalar uygulamasında:**

* **Durumlar:** Çalma, Duraklatma, Önceki Şarkı, Sonraki Şarkı
* **Geçişler:**
  + Çalma durumundan duraklatmaya geçme
  + Duraklatma durumundan çalmaya geçme
  + Çalma durumundan önceki şarkıya geçme
  + ...
* **Test Senaryoları:**
  + Bir şarkıyı çal ve duraklat düğmesine bas.
  + Duraklatılmış bir şarkıyı çalmaya başlat.
  + Çalan bir şarkı sırasında önceki şarkı düğmesine bas.

**Ek Örnekler:**

* **Bir formda zorunlu alanların doldurulması:** Tüm zorunlu alanları boş bırakarak, sadece birini doldurarak, hepsini doldurarak farklı senaryolar oluşturulabilir.
* **Bir e-posta doğrulama sistemi:** Geçerli, geçersiz, tekrar eden e-posta adresleri ile testler yapılabilir.
* **Bir arama motoru:** Farklı anahtar kelimelerle, farklı sıralama kriterleriyle aramalar yapılarak sonuçların doğruluğu kontrol edilebilir.
* **Bir online oyun:** Oyunun farklı seviyelerinde, farklı karakterlerle oyun oynanarak hatalar tespit edilebilir.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Beyaz Kutu Testi:**

**Komut Testi Örnekleri**

**Komut testi (Statement Testing)**, beyaz kutu testinin temel bir tekniği olup, kodun her bir satırının en az bir kez çalıştırıldığını doğrulamaya yöneliktir. Bu sayede, kodun tüm satırlarının beklendiği gibi çalışıp çalışmadığı kontrol edilir.

**Örnek 1: Basit Bir Toplama İşlemi**

JavaScript

function topla(a, b) {

let toplam = a + b;

return toplam;

}

* **Komut Testi Senaryoları:**
  + topla(2, 3)
  + topla(-5, 7)
  + topla(0, 0)

Bu senaryolar, fonksiyonun her satırını en az bir kez çalıştırarak, toplama işleminin doğru yapıldığını doğrular.

**Örnek 2: Koşullu Durum**

JavaScript

function kontrolEt(sayi) {

if (sayi > 0) {

console.log("Sayı pozitiftir.");

} else {

console.log("Sayı negatif veya sıfırdır.");

}

}

* **Komut Testi Senaryoları:**
  + kontrolEt(5)
  + kontrolEt(-2)
  + kontrolEt(0)

Bu senaryolar, if-else bloğunun her iki kolunun da çalıştırıldığını gösterir.

**Örnek 3: Döngü**

JavaScript

function faktoriyel(sayi) {

let sonuc = 1;

for (let i = 1; i <= sayi; i++) {

sonuc \*= i;

}

return sonuc;

}

* **Komut Testi Senaryoları:**
  + faktoriyel(0) // Döngü hiç çalışmaz
  + faktoriyel(3) // Döngü 3 kez çalışır
  + faktoriyel(5) // Döngü 5 kez çalışır

Bu senaryolar, döngünün farklı tekrar sayılarında çalıştırıldığını gösterir.

**Örnek 4: Bir E-ticaret Sitesindeki Sepet İşlemi**

JavaScript

function hesaplaToplam(urunler) {

let toplam = 0;

for (let urun of urunler) {

toplam += urun.fiyat \* urun.miktar;

}

return toplam;

}

* **Komut Testi Senaryoları:**
  + Boş bir ürün listesi ile çağır
  + Tek bir ürünle çağır
  + Birden fazla ürünle çağır

**Komut Testinin Sınırlamaları**

* **Tam Kapsamı Sağlamaz:** Kodun her bir dalının (branch) veya koşulun (condition) doğru çalıştığını garanti etmez.
* **Hataları Tam Olarak Tespit Etmeyebilir:** Bazı hatalar, tüm satırlar çalıştırıldığında bile ortaya çıkmayabilir (örneğin, off-by-one hataları).

**Diğer Beyaz Kutu Testi Teknikleriyle Birlikte Kullanım**

Komut testi, diğer beyaz kutu test teknikleri (dal testi, karar tablosu testi vb.) ile birlikte kullanıldığında daha kapsamlı bir test süreci oluşturulur. Bu sayede, kodun hem yapısal hem de mantıksal olarak doğru çalıştığı doğrulanabilir.

**Özetle,** komut testi, kodun temel bir kapsamını sağlamak için önemli bir tekniktir. Ancak, yazılımın kalitesini artırmak için diğer beyaz kutu test teknikleriyle birlikte kullanılması önerilir.

**Dal Testi**

Bir yazılımdaki karar yapıları (if, else, switch gibi) programın akışını belirler. Dal testi, bu karar yapıları içindeki her bir olası yolu en az bir kez takip ederek, kodun tüm dallarının doğru çalıştığını kontrol eder.

**Örnek:**

JavaScript

function hesapla(sayi) {

if (sayi > 0) {

return "Pozitif";

} else if (sayi < 0) {

return "Negatif";

} else {

return "Sıfır";

}

}

Bu basit JavaScript fonksiyonunda üç farklı dal bulunmaktadır:

* sayi > 0 olduğu durum
* sayi < 0 olduğu durum
* sayi 0'a eşit olduğu durum

Dal testi için bu üç durumu da kapsayan test verileri oluşturulmalıdır:

* **Test Verisi 1:** sayi = 5 (Pozitif)
* **Test Verisi 2:** sayi = -3 (Negatif)
* **Test Verisi 3:** sayi = 0 (Sıfır)

**Dal Testi Örnekleri**

**1. Bir E-ticaret Sitesindeki Sepet İşlemi:**

* **Karar Yapıları:** Ürün miktarı sıfırdan büyük mü? Kullanıcı üye mi? Kupon kodu geçerli mi?
* **Dal Testi:**
  + Ürün eklemeden sepete gitmeye çalış
  + Üye olmadan alışveriş yapmaya çalış
  + Geçersiz bir kupon kodu gir
  + Ürün ekle, üye ol ve geçerli bir kupon kodu kullan

**2. Bir Banka Uygulamasındaki Para Çekme İşlemi:**

* **Karar Yapıları:** Hesapta yeterli para var mı? Günlük çekme limiti aşıldı mı? Kartın bloke durumu nedir?
* **Dal Testi:**
  + Hesaptaki paradan fazla çekmeye çalış
  + Günlük çekme limitini aşacak şekilde çekim yap
  + Bloke olan bir kartla çekim yap
  + Yeterli para varken, limit içinde ve aktif bir kartla çekim yap

**3. Bir Oyun Uygulamasındaki Seviye Geçişi:**

* **Karar Yapıları:** Oyuncu belirli bir puanı geçti mi? Belirli bir görevi tamamladı mı?
* **Dal Testi:**
  + Oyuncu seviye atlama puanına ulaşmadan bir üst seviyeye geçmeye çalışsın
  + Gerekli görevi tamamlamadan seviye atlamaya çalışsın
  + Hem puanı geçsin hem de görevi tamamlasın

**Tecrübeye Dayalı Test Tekniklerine Örnekler**

Tecrübeye dayalı test teknikleri, yazılım testçisinin geçmiş deneyimleri, sektör bilgisine ve sezgisine dayalı olarak hataları tespit etme yöntemleridir. Bu teknikler, yazılımın farklı alanlarında ortaya çıkabilecek potansiyel sorunları önceden tahmin etmeye yardımcı olur. İstediğiniz tekniklere örnekler verelim:

**Hata Tahminleme**

Hata tahminleme, testçinin geçmiş deneyimlerine ve yazılımın yapısına dayanarak potansiyel hataları önceden belirlemesi ve bu hataları hedef alan test senaryoları oluşturmasıdır.

* **Örnekler:**
  + **Veri giriş alanları:** Geçersiz karakter girişi, boş alan bırakma, özel karakter kullanımı gibi hataların olabileceğini öngörerek testler tasarlanır.
  + **Hesaplamalar:** Toplama, çıkarma, çarpma gibi işlemlerde yuvarlama hataları, sıfır bölme gibi durumların kontrol edilmesi.
  + **Veritabanı işlemleri:** Veri kaybı, veri tutarlılığı, veri tekrarı gibi sorunların olabileceğini düşünerek testler yapılır.
  + **Performans:** Yüksek yük altında sistemin yavaşlaması, çökmesi gibi durumların test edilmesi.
  + **Güvenlik:** SQL injection, cross-site scripting gibi güvenlik açıklarının olabileceğini öngörerek testler tasarlanır.

**Keşif Testi (Exploratory Testing)**

Keşif testi, testçinin yazılımı keşfederken aynı zamanda hataları bulmaya çalıştığı bir test tekniğidir. Bu teknik, önceden belirlenmiş bir test planı olmadan, testçinin sezgisine ve deneyimine dayanarak yapılır.

* **Örnekler:**
  + **Rastgele veri girişi:** Farklı türde ve miktarda verileri sisteme girerek beklenmedik davranışları gözlemlemek.
  + **Arayüzü keşfetme:** Farklı butonlara basmak, menüleri gezmek, linklere tıklamak gibi işlemlerle arayüzün bütünlüğünü ve tutarlılığını kontrol etmek.
  + **Sınır koşullarını test etme:** Minimum ve maksimum değerler, sınır durumları gibi özel durumlarda sistemin nasıl davrandığını gözlemlemek.
  + **Negatif testler:** Sistemin hata vereceği beklenen durumlarda sistemi zorlamak.

**Kontrol Listesine Dayalı Test**

Kontrol listesi, testçinin belirli bir alanda kontrol etmesi gereken noktaları içeren bir belgedir. Bu liste, geçmiş deneyimlere, standartlara ve yazılımın özelliğine göre hazırlanır.

* **Örnekler:**
  + **Erişilebilirlik kontrol listesi:** Engelli kullanıcıların siteyi kullanabilmesi için gerekli kriterlerin kontrol edilmesi (örneğin, ekran okuyucularla uyumluluk, kontrast oranları).
  + **Güvenlik kontrol listesi:** SQL injection, cross-site scripting, kimlik doğrulama gibi güvenlik açıklarının kontrol edilmesi.
  + **Performans kontrol listesi:** Sayfa yükleme süreleri, yanıt süreleri, kaynak kullanımı gibi performans metriklerinin ölçülmesi.
  + **Kullanılabilirlik kontrol listesi:** Arayüzün anlaşılırlığı, tutarlılığı, kullanıcı dostu olup olmadığı gibi faktörlerin kontrol edilmesi.

**Örnek: Bir E-ticaret Sitesi Ürün Sayfası Kontrol Listesi**

|  |  |  |
| --- | --- | --- |
| **Kontrol Noktası** | **Açıklama** | **Beklenen Sonuç** |
| Ürün Adı | Ürün adı doğru ve açıklayıcı mı? | Evet, ürünün adını doğru olarak yansıtmalı. |
| Ürün Görseli | Ürün görseli net ve yüksek çözünürlüklü mü? | Evet, ürünün özelliklerini net bir şekilde göstermeli. |
| Ürün Fiyatı | Fiyat doğru ve güncel mi? | Evet, fiyat etiketinde belirtilen fiyatla aynı olmalı. |
| Ürün Açıklaması | Ürün açıklaması detaylı ve anlaşılır mı? | Evet, ürünün özelliklerini ve kullanım alanlarını açıkça belirtmeli. |
| Stok Durumu | Stok bilgisi doğru mu? | Evet, stokta olup olmadığı doğru olarak gösterilmeli. |
| Sepete Ekle Butonu | Buton çalışıyor mu? Sepete ürün eklendiğinde sayı artıyor mu? | Evet, ürün sepete eklenmeli ve sepet sayısı güncellenmeli. |
| İndirim veya Kampanya Bilgileri | İndirim veya kampanya varsa doğru şekilde gösteriliyor mu? | Evet, geçerli indirimler ve kampanyalar belirtilmeli. |
| Benzer Ürünler Bölümü | Benzer ürünler öneriliyor mu? | Evet, ürünle ilgili benzer ürünler listelenmeli. |
| Müşteri Yorumları Bölümü | Müşteri yorumları görüntüleniyor mu? | Evet, eğer varsa müşteri yorumları gösterilmeli. |
| Mobil Uyumluluk | Mobil cihazlarda düzgün görüntüleniyor mu? | Evet, tüm mobil cihazlarda sorunsuz çalışmalı. |

**Bu tekniklerin Kombinasyonu**

Genellikle bu teknikler tek başına değil, bir arada kullanılır. Örneğin, bir testçi, hata tahminleme ile potansiyel sorunları belirleyebilir, keşif testi ile yazılımı daha derinlemesine inceleyebilir ve kontrol listesi ile de eksik kalmadan testleri tamamlayabilir.

**Tecrübeye dayalı test tekniklerinin avantajları:**

* **Esneklik:** Önceden belirlenmiş bir plan olmadan, testçinin yaratıcılığına ve sezgisine dayalı olarak testler yapılabilmesi.
* **Hızlı sonuç:** Kritik hataların hızlı bir şekilde tespit edilmesi.
* **Kapsamlılık:** Farklı açılardan yazılımın test edilmesi.

**Dezavantajları:**

* **Subjektiflik:** Test sonuçları testçinin deneyimine ve bilgisine bağlı olarak değişebilir.
* **Tekrarlanabilirlik:** Aynı testin farklı testçiler tarafından aynı sonuçları vermesi zor olabilir.
* **Tam kapsamlılık sağlamayabilir:** Tüm olası hataların tespit edilmesi garanti edilemez.