

T.C.

MANİSA CELAL BAYAR ÜNVERSİTESİ

KIRKAĞAÇ MESLEK YÜKSEKOKULU

BİLGİSAYAR PROGRAMCILIĞI

Ders Programı Hazırlama Otomasyonu

Hazırlayanlar

231809087 ENES ERMIŞ

231809015 MEHMET POLAT ALEMDAR

231809003 HİLAL AŞGİT

231809084 HASAN AKBUDAK

Danışman

Öğr.Gör. LEVENT KARAKUZ

İçindekiler Tablosu

| | |
|---|----------|
| Giriş | 3 |
| Uygulamayı kurma: | 3 |
| IntelliJ IDEA Kurulumu: | 3 |
| Uygulamayı Açma: | 3 |
| Proje Açma: | 3 |
| Bağımlılıkları Yükleme:..... | 3 |
| Main Class'ı Başlatma: | 3 |
| Mysql Ayarlama: | 3 |
| Uygulama ne işe yarıyor ? | 4 |
| Hızlı ve Verimli Program Oluşturma: | 4 |
| Çalışmaları Önleme:..... | 4 |
| En Uygun Atamaları Yapma: | 4 |
| Esneklik ve Uyum: | 4 |
| Teklif edilen Sistem | 4 |
| 1. Kısıtlı Yetkili Kullanıcı | 4 |
| 2. Tam Yetkili Kullanıcı | 5 |
| Otomasyonumuz | 6 |
| Kullanılabilirlik | 6 |
| Güvenilirlik..... | 6 |
| Performans | 6 |
| Desteklenebilirlik | 6 |
| İmplementasyon | 6 |
| Arayüz..... | 6 |
| Gizlilik Gereksinimi..... | 6 |
| Aktörler..... | 6 |
| Olaylar | 7 |
| Senaryolar | 7 |
| Veri Tabanı Diyagramı: | 8 |
| Program Aşamaları Kodlarla..... | 9 |
| Çalıştırma: | 9 |
| Giriş Yapma: | 10 |
| Menü Oluşturma :..... | 12 |
| Field Oluşturma:..... | 16 |
| ComboBox Oluşturma: | 17 |
| Tablo Özelleştirme: | 18 |

| | |
|------------------------------|----|
| Butonlar: | 21 |
| Geri Git: | 21 |
| Veriyi Sil: | 23 |
| Veriyi Kayıt Et: | 23 |
| Veriyi Güncelle: | 29 |
| Cıktı Al: | 32 |
| Özel Cıktı Al: | 35 |
| Algoritma Aşamaları Kodlarla | 39 |
| 1. Bölümlerin Müsaitliği | 40 |
| 2. Dersliklerin Müsaitliği | 41 |
| 3. Öğretmen Müsaitliği | 43 |
| 4. Ders Bilgileri | 45 |
| Programı Başlatmak | 47 |
| Alogirtma Akış Diyagramı | 54 |

Giriş

Uygulamayı kurma:

IntelliJ IDEA Kurulumu:

- Öncelikle IntelliJ IDEA'nın resmi web sitesinden uygun sürümü indirin ve kurulumu gerçekleştirin. ([İndirmek için tıkla](#))
- Kurulum sihirbazını takip ederek temel ayarları seçip, IntelliJ IDEA'nın bilgisayarınıza başarılı bir şekilde kurulmasını sağlayın.

Uygulamayı Açma:

- Kurulumun tamamlanmasının ardından IntelliJ IDEA uygulamasını başlatın.
- Sol üst köşede bulunan "File" menüsüne tıklayın. ([Github İçin Tıkla](#))

Proje Açma:

- "File" menüsünden "Open" seçeneğine tıklayın ve bilgisayarınızda bulunan proje dosyanızı seçin.

Bağımlılıkları Yükleme:

- Projenizi başarıyla açtıktan sonra, bağımlılıkları yüklemek için "Ctrl + Shift + O" tuş kombinasyonunu kullanın ve IntelliJ IDEA'nın sağ alt köşesinde bulunan "Load Maven Changes" seçeneğine tıklayın.

Main Class'ı Başlatma:

- Sağ alt köşede bulunan "Run" menüsünden "Main Class" seçeneğine gidin.
- Ardından "Public Class Main" başlığını bulun ve başlatmak için uygun seçeneği seçin.

Mysql Ayarlama:

Otomasyonu sorunsuz bir şekilde kullanmak ve veritabanı işlemlerini gerçekleştirmek için MySQL bağlantı bilgilerinizi doğru bir şekilde ayarlamak oldukça önemlidir. Aşağıda, bağlantı bilgilerini düzenlemek için adım adım bir rehber bulabilirsiniz:

Daha sonrasında Fun -> Database -> Config dosyasında database ayarlarını ayarlamamız için ufak bir kod satırı mevcuttur.

```
public class config {  
    6 usages  
    public static DriverManagerDataSource dataSource;  
    public static JdbcTemplate jdbcTemplate;  
  
    static {  
        // DataSource konfigürasyonu  
        dataSource = new DriverManagerDataSource();  
        dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");  
        dataSource.setUrl("jdbc:mysql://{{Ba\u0111lant\u0111Adresi}}:{PortNumaras\u0111}/{DataBaseAd\u0111}");  
        dataSource.setUsername("{Kullan\u0111ıcıAd\u0111}");  
        dataSource.setPassword("{Sifre}");  
        jdbcTemplate = new JdbcTemplate(dataSource);  
    }  
}
```

Burada {} parantezlerin içine yazdiğim yerler Mysql bilgilerinizi güncelleneniz gerekmektedir.

Uygulama ne işe yarıyor ?

Modern eğitim ortamlarında öğretmenlerin en büyük zorluklarından biri, öğretmen ve ders kaynaklarını etkili bir şekilde yöneterek, öğrencilere en iyi eğitimi sunmaktadır. Bu noktada, ders programı oluşturmak sıkılıkla zaman alıcı ve karmaşık bir süreç olabilir. Ancak, şimdi bu zorluğu ortadan kaldırın bir çözüm var: Akıllı Ders Programı Otomasyonu!

Bu benzersiz uygulama, öğretmenlere öğretmen bilgileri, ders içerikleri, okul zamanı ve bölüm detayları gibi kritik verileri girdikten sonra, ders çakışmalarını önleyen ve en uygun şekilde böümlere atama yapabilen bir akıllı sistem sunuyor. İşte bu uygulamanın sunduğu faydalar:

Hızlı ve Verimli Program Oluşturma:

Geleneksel yöntemlerle saatler süren ders programı oluşturma sürecini, bu otomasyon sayesinde sadece birkaç tıklama ile tamamlayabilirsiniz. Bu, öğretmenlere zaman kazandırarak daha önemli görevlere odaklanma imkanı sunar.

Çakışmaları Önleme:

Uygulama, dersler arasındaki çakışmaları etkili bir şekilde engeller. Bu, öğrencilere daha düzenli bir program sunmanın yanı sıra, öğretmenlere de daha etkili bir şekilde ders vermelerine olanak tanır.

En Uygun Atamaları Yapma:

Dersleri, öğretmenlerin uzmanlık alanlarına ve böümlere en uygun şekilde atayarak, öğrencilerin daha iyi bir eğitim almasını sağlar. Bu sayede, her öğrenci kendi ilgi ve yeteneklerine uygun derslere katılabilir.

Esneklik ve Uyum:

Uygulama, değişen okul koşullarına ve öğretmen programlarındaki güncellemelere hızlı bir şekilde adapte olabilir. Bu da öğretmenlere esneklik sağlayarak, herhangi bir aksaklığa karşı hazırlıklı olmalarını sağlar.

Bu Akıllı Ders Programı Otomasyonu ile, eğitim kurumları daha hızlı, daha düzenli ve daha verimli bir şekilde ders programları oluşturabilir. Bu da öğrencilerin ve öğretmenlerin en iyi şekilde faydalananmasını sağlayarak eğitim kalitesini artırır. Zamanınızı daha etkili bir şekilde yönetmek ve daha iyisini başarmak için şimdi bu benzersiz uygulamayı keşfedin!"

Teklif edilen Sistem

Sistemde 2 Adet kullanıcı vardır. Tam Yetkili ve Kısıtlı yetkili olmak üzere şimdilik detaylıca anlatalım..

1. Kısıtlı Yetkili Kullanıcı

Kısıtlı yetkili kullanıcılar öğretmenlerdir ve öğretmenler programda sadece kendi bilgilerini görebilmektedir. Ayrıca çıktı olarak hem tablo halinde hemde ders programı halinde de çıktı alabilirler...

| İventkarakuz - Öğretmen Ders Programı | | | | | | | | |
|---------------------------------------|--------------|---------------------|---------------|----------|------------|-------------|-------------|-------------------|
| ID | Episode | Lessons | Teacher | Day | LessonTime | Classroom | Lesson_Code | Classroom_Numb... |
| 818 | Bilgisayar 1 | Veri Tabanı | İeventkarakuz | Çarşamba | 4 | Laboratuvar | blg-02/1 | 3 |
| 819 | Bilgisayar 1 | Veri Tabanı | İeventkarakuz | Çarşamba | 5 | Laboratuvar | blg-02/1 | 3 |
| 820 | Bilgisayar 1 | Veri Tabanı | İeventkarakuz | Çarşamba | 6 | Laboratuvar | blg-02/1 | 3 |
| 821 | Bilgisayar 1 | Veri Tabanı | İeventkarakuz | Cuma | 1 | Laboratuvar | blg-02/2 | 4 |
| 822 | Bilgisayar 1 | Veri Tabanı | İeventkarakuz | Cuma | 2 | Laboratuvar | blg-02/2 | 4 |
| 823 | Bilgisayar 1 | Veri Tabanı | İeventkarakuz | Cuma | 3 | Laboratuvar | blg-02/2 | 4 |
| 785 | Bilgisayar 2 | Görsel Programla... | İeventkarakuz | Cuma | 4 | Laboratuvar | blg-7/1 | 2 |
| 786 | Bilgisayar 2 | Görsel Programla... | İeventkarakuz | Cuma | 5 | Laboratuvar | blg-7/1 | 2 |
| 787 | Bilgisayar 2 | Görsel Programla... | İeventkarakuz | Cuma | 6 | Laboratuvar | blg-7/1 | 2 |

Kısıtlı yetkili kullanıcılar örneğin bu şekilde verilerini görebilirler.

2. Tam Yetkili Kullanıcı

Tam yetkili kullanıcılar menülerde hersey yapabilecek kullanıcılardır. Ufak bir ekran görüntüsünden neler yapabileceklerini gösterelim...



Şimdi sadece bu menüye göre yorumlarsak kapataslak Tam yetkili kullanıcılar,

1. Unvan İşlemleri (Ekleme, Düzelme, Silme, Çıktı Alma)
2. Öğretmen İşlemleri (Ekleme, Düzelme, Silme, Çıktı Alma)
3. Müsaitlik İşlemleri (Ekleme, Düzelme, Silme, Çıktı Alma)
4. Derslik İşlemleri (Ekleme, Düzelme, Silme, Çıktı Alma)
5. Öğretmenlere ders aktarma İşlemleri (Ekleme, Düzelme, Silme, Çıktı Alma)
6. Bölümlere ders aktarma İşlemleri (Ekleme, Düzelme, Silme, Çıktı Alma)
7. Okulun Acık olacağı günleri belirleme (Ekleme, Düzelme, Silme, Çıktı Alma)
8. Okulun açılacağı saatı, ders süresini, teneffüs süresini, öğlen molasını, öğlen mola süresini, okulun kapatılacağı saatı belirleme işlemleri.
9. Bütün programları tabloda listeleyip gerekli İşlemleri (Ekleme, Düzelme, Silme, Çıktı Alma)
10. Sadece Öğretmene göre gerekli İşlemleri (Ekleme, Düzelme, Silme, Çıktı Alma)
11. Sadece Bölümüne göre gerekli İşlemleri (Ekleme, Düzelme, Silme, Çıktı Alma)
12. Ders Programını sıfırda oluşturma.

Tam Yetkili bir kullanıcı buradaki tüm bilgileri eksiksiz bir şekilde girdiği zaman ders programını hazırlayabilir. Eksik bilgilerde hata uyarı mesajlarımız mevcuttur.

Otomasyonumuz

Kullanılabilirlik

Projede, ders programı hazırlama sürecini hızlandırmak için kullanıcıların ihtiyaç duyduğu ana işlevlere hızlı erişim sağlayan menüler bulunmalıdır. Bu menüler, kullanıcıların istedikleri işlemlere kolayca ulaşmalarını sağlamalı ve zaman kaybını minimuma indirmelidir. Oluşturulan program istenildiği gibi çıktı alabilir ve basit bir arayüzü sayesinde kullanıcılarla güçlü zorluklar yaratmamaktadır. Ayrıca en üst düzeyde optimize sağlanmış ve geliştirilebilmesi çok basit bir şekilde kodlanmıştır.

Güvenilirlik

Ders programı otomasyonunda verilerin korunması ve güvenliği oldukça önemlidir. Yaşanabilecek 3. Parti uygulamala açıklarının ve illegal girişleri engellemek için ufak güvenilir kod bütünlüğü kullanarak bu açıkları kapattık. Mesela login sistemindeki sql saldırınızı engelledik. Yada DDOS saldırınızı azaltmak için programda ufak engeller ekledik. Onun dışında database'nizi kodumuzun içerisinde güvenilir bir şekilde gömdüğümüz için bizlere güvenebilirsiniz. Ama en önemli yaşayacağınız en ufak sorunda yanınızda bulunacağımızı bilmelisiniz.

Performans

Performans olaraktan verileri sürekli databaseden çekip yaşayacağımız ağ trafiğinden dolayı verilerimizi nodelere kayıt edip en son işlemleri veritabanından yapıyoruz bu sayede daha az dışardan bağlantı cekerek performansımızı güncelndirdik Java 17 ile en düşük sistem gereksinimlerinde bile çalışacak uygun kod bloklarını ve döngülerini en az kullanacak şekilde programı oluşturduk.

Desteklenebilirlik

Projede kullanılan veritabanının transaction ve foreign key desteğinin olması gereklidir. Programın kurulduğu platformun da Java destekli olmalı ve Java programlarını çalıştırılmalıdır. Özgür yazılım lisansı sayesinde yasal gereksinimleri bulunmamakta

Implementasyon

Sistem için verilen veritabanlarını dikkatli bir şekilde bağlaştırdı UML Diyagramımızı oluşturduk aşağıda diyagramımız mevcut ve buna göre gerekli java kodlarında bağlantılarımızı destekledik.

Arayüz

Otomasyonda kullanıcı arabirimini pencerelerdir. Tek bir pencerede yer alacak programda, kullanıcıyı login sayfası karşılar. Login kullanıcı, rolüne göre kendine ait bilgilere ulaşır. Görsel özellikler için gereksinim dökümanı ayrıca hazırlanacaktır. Kullanıcı düğmelere basarak veya formları doldurarak işlem yapacaktır. Ara yüzün çıktısı ekran, girdileri ise Mouse ve klavyedir.

Gizlilik Gereksinimi

Kullanıcıların sınırlanılması, sadece kendilerine ait bölümleri görebilmesi ve gerektiğinde bu ayarların değiştirilmesi yüksek derecede önemlidir. Veritabanında, dataların güvenliği için ek bir özellik bulunmuyor. Sunucunun başına geçen veya dosyaları çalan birisi kayıtlara ulaşabilir. Sunucu güvenliği işletim sisteminin kendi güvenliği ile eş. Program açılırken kullanıcı adı ve şifre soracak. Sistem bu bilgileri kullanarak izinlerini kontrol edecektir. Kullanıcılar veritabanına başka bir programla bağlanıp bilgileri görebilir. Programda kullanılan gizlilik sadece verinin bütünlüğü ve iş bölümü için. İstenmeyen kullanıcıların sisteme bağlanması engellemek işletim sisteminin sorumluluğundadır.

Aktörler

| Aktör | Tanım |
|-----------------|---|
| Tam Yetkili | Sistem içerisinde tamamen bütün yetkinin ait olduğu ve sistem içerisindeki bütün düzenlemeleri yapan kullanıcıdır. Kısaca sistemdeki YÖNETİCİ'dir |
| Kısıtlı Yetkili | Sadece ders programını görüntüler ve sistemde çıktı alabilir. Kısaca yetkisi olmayan Öğretmenlerdir. |
| Sistem | Ders programı oluşturma, çakışmaları kontrol etme, otomatik derslik ve laboratuvar atama gibi görevleri gerçekleştiren bir sistem aktöründür. Çakışmaları kontrol eder ve yöneticiye uyarılar verir, ancak müdahale etmez. |

| | |
|--|--|
| | Yönetici, uyarıları değerlendirip istediği şekilde değişiklik yapabilir. |
|--|--|

Olaylar

Sistemin haftalık ders programı hazırlama otomasyonu olaylarını şu şekilde sıralayabiliriz:

Hoca Bilgisi Girişi:

Yönetici, hocaların bilgilerini (unvanları ile birlikte) sisteme girebilir.

Ders Bilgisi Girişi:

Yönetici, derslerin bilgilerini (ders kodu, adı, ders saatı) sisteme girebilir.

Hoca-Ders Atama:

Kullanıcı, hangi hoca hangi derse gireceğini sisteme girebilir.

Bir hoca birden fazla derse atanabilir.

Bir derse birden fazla hoca atanabilir.

Şube ve Program Ayarları:

Yönetici, bir şube için günlük ders saatlerini, başlangıç saatini, ders süresini, teneffüs süresini, öğle tatili süresini belirleyebilir.

Çakışma Kontrolü:

Sistem, aynı gün ve saatte ismi farklı birden fazla derse atanmış bir hocanın durumunu kontrol eder.

Eğer çakışma varsa, yöneticiye uyarı verir.

Müsaitlik Girişi:

Hocalar, müsait oldukları günleri ve saatleri sisteme girebilir.

Derslik ve Laboratuvar Atama:

Sistem, bir dersin laboratuvara yapılması gerekiyorsa otomatik olarak laboratuvar ataması yapar.

Eğer müsait bir laboratuvar bulunamıyorsa, sistem yöneticiye müsait derslikleri önerir.

Otomatik Derslik Atama:

Sistem, bir dersin öğrenci sayısına göre en uygun dersliğe otomatik olarak atama yapar, en az boş yer bırakılan dersliği sefer.

Çakışma İzni:

Yönetici, aynı gün ve saatte ismi farklı birden fazla derse atanmış bir hocanın durumu hakkında uyarı alındığında, sistemin önerdiği çözümü değerlendirip onay verebilir.

Program Çıktıları:

Kullanıcı, bölüm, sistem ve hoca bazında ayrı ayrı haftalık program çıktıları alabilir.

Manuel Değişiklikler:

Yönetici, ders programında istediği şekilde manuel değişiklikler yapabilir.

Sistem, kriterlere uygun olmayan durumlar hakkında yöneticiyi uyarır, ancak müdahale etmez.

Senaryolar

Hoca Ekleme:

Yönetici, yeni bir hoca ekleyerek unvan, ad, soyad gibi bilgileri sisteme kaydeder.

Ders Ekleme:

Yönetici, yeni bir ders ekleyerek ders kodu, ders adı, ders saatı gibi bilgileri sisteme kaydeder.

Hoca-Ders Ataması Yapmak:

Kullanıcı, belirli bir hoca için hangi dersin atanacağını seçer ve sisteme kaydeder.

Şube Programı Ayarlamak:

Yönetici, bir şube için günlük ders saatleri, başlangıç saati, teneffüs süresi gibi program ayarlarını düzenler.

Çalışma Kontrolü:

Sistem, bir hoca için çakışan ders atamalarını kontrol eder ve yöneticiye uyarı verir.

Müsaitlik Bilgisi Girmek:

Hoca, kendi müsait olduğu günleri ve saatleri sisteme girer.

Derslik ve Laboratuvar Ataması:

Sistem, bir dersin laboratuvara yapılması gerekiyorsa otomatik olarak laboratuvar ataması yapar.

Otomatik Derslik Atama:

Sistem, derslerin öğrenci sayılarına göre en uygun dersliklere otomatik olarak atama yapar.

Çalışma İzni Verme:

Yönetici, çakışan ders atamaları konusunda sistem tarafından verilen uyarılara izin verir.

Program Çıktısı Almak (Bölüm):

Kullanıcı, belirli bir bölüm için haftalık ders programını sisteme sorgular ve çıktı alır.

Program Çıktısı Almak (Sistem):

Kullanıcı, genel sistem haftalık ders programını sorgular ve çıktı alır.

Program Çıktısı Almak (Hoca):

Kullanıcı, belirli bir hoca için haftalık ders programını sisteme sorgular ve çıktı alır.

Manuel Değişiklik Yapmak:

Yönetici, ders programında manuel değişiklik yapar ve sistem tarafından verilen uyarıları değerlendirir.

Ders Sayısı Azaltmak:

Yönetici, bir dersin haftalık saat sayısını azaltır ve sistemden olası çalışma uyarılarını değerlendirir.

Derslik Kapasitesini Güncellemek:

Yönetici, bir dersliğin kapasitesini günceller ve sistemin otomatik atama algoritmasını etkiler.

Hoca Bilgisi Güncellemek:

Yönetici, bir hocanın bilgilerini günceller ve sistemdeki ilgili atamalara yansıtır.

Ders Bilgisi Güncellemek:

Yönetici, bir dersin bilgilerini günceller ve sistemdeki ilgili atamalara yansıtır.

Derslik Bilgisi Ekleme:

Yönetici, yeni bir derslik ekler ve sistemde otomatik atama algoritmasını etkiler.

Tüm Ders Atamalarını Silmek:

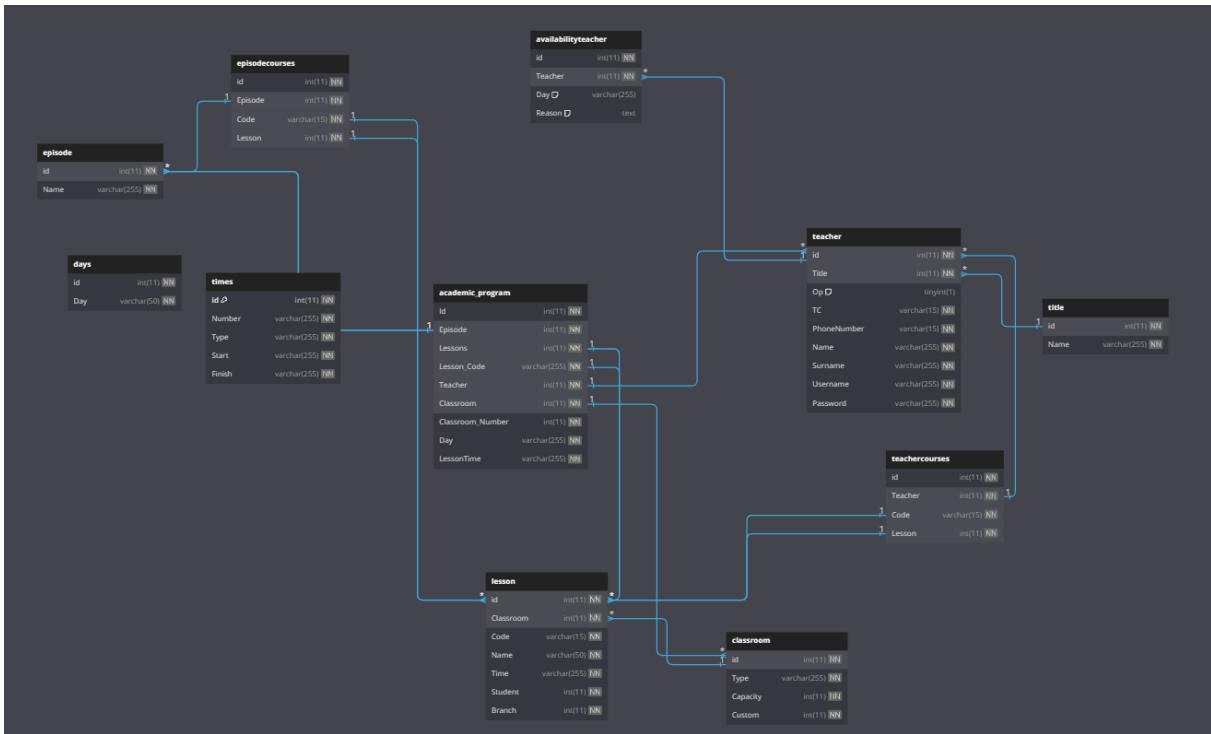
Yönetici, tüm ders atamalarını siler ve sistemde temiz bir başlangıç yapar.

Çalışma Kontrolsüz İzin:

Yönetici, çakışan ders atamalarına sistem uyarı olmadan izin verir.

Veri Tabanı Diyagramı:

Veritabanında kurduğumuz bağlantılar aşağıda bir görselde belirtilmiştir. Belirtilen görselde kurulan bağlantılar (FOREIGN KEY) güncelleme işlemlerinde veri güncellemesi için ayrıntılı kullanılmıştır.



Program Aşamaları Kodlarla

Çalıştırma:

```

public class Main {
    public static void main(String[] args) {

        notExits.main();
        // Database bağlantı menüsü açar.

        // FrameName Menü adı
        // SizeX ve SizeY Menülerin boyutu
        // String dizisinin içinde olacak label ve fieldlar.
        // rows ise field alt alta kaçar sıralanacak.
        // Process işlem kodu
        // FrameType buton ve işlem türü
        // TabloSql ise tabloda sql sorusu

        ArrayList deneme = new ArrayList();

        deneme.add("Username");
        deneme.add("Password");

        createFrame.main( frameName: "Kullanıcı giriş ekranı", frameSizeX: 400 , frameSizeY: 175, deneme,
            rows: 3,
            frameType: "Login", tabloSql: " " );

    }
}

```

Program ilk olarak bu class'tan başmaktadır. NotExits.Main methoduna ilk olarak gidecektir. Gittikten sonra Mevcut olmayan Tableleri oluşturmamızı sağlıyor.

```

public class notExists {
    
    // Mevcut olmayan table'leri oluşturma
    public static void main() {
        title.main();
        teacher.main();
        days.main();
        availabilityTeacher.main();
        episode.main();
        classroom.main();
        lesson.main();
        teacherCourses.main();
        times.main();
        episodeCourses.main();
        academicProgram.main();
    }
}

```

Buradan tek tek table'leri oluşturuyoruz. Mesela örnek bir methodu açalım.

```

public class lesson {
    public static void main() {
        String sql = "CREATE TABLE IF NOT EXISTS Lesson (" +
            + "id INT AUTO_INCREMENT PRIMARY KEY," +
            + "Classroom INT NOT NULL," +
            + "Code VARCHAR(15) NOT NULL UNIQUE," +
            + "Name VARCHAR(50) NOT NULL," +
            + "Time VARCHAR(255) NOT NULL," +
            + "Student INT NOT NULL," +
            + "Branch INT NOT NULL," +
            + "FOREIGN KEY (Classroom) REFERENCES Classroom(id) ON DELETE CASCADE," +
            + "UNIQUE (Code, Name)" +
            + ")";
        jdbcTemplate.execute(sql);
    }
}

```

Buradan bu şekilde databasesleri oluşturuyoruz. Gördüğünüz method lesson.main() methodudur. Bu method sayesinde databasemizde eğer Lesson tablosu yoksa oluşturacaktır. Bütün tabloları bu şekilde oluşturdukten sonra bir menü açmamız lazım. Menü açılmasını diğer başlıkta daha detaylı anlatacağım o yüzden geçiyorum.

Giriş Yapma:

Uygulamayı çalıştırınca login ekranına atar login ekranında ise kullanıcının olup olmadığını tabloda o kullanıcı adına ve şifresine sahip 1 kullanıcı var mı yok mu diye bir sorgu yapıp kullanıcının doğruluğunu aldım.

```

// Eğer username ve password 0 tane varsa haliyle kullanıcı olmaz.
// o yüzden böyle bir login sistemi hazırladım.
1 usage
public static boolean verify(String username, String password) {
    String sql = "SELECT COUNT(*) FROM Teacher WHERE Username = ? AND Password = ?";
    return 0 != jdbcTemplate.queryForObject(sql, Integer.class, username, password);
}

```

Bu kod parçasında da gördüğünüz üzere...

```

2 usages
public static JButton loginButton() {

    JButton button = new JButton( text: "Giriş Yap");

    button.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {

            // Field verilerdeki bilgileri alıyoruz.

            String username = findTextFieldEntry( fieldName: "Username").getTextField().getText();
            String password = findTextFieldEntry( fieldName: "Password").getTextField().getText();

            // Boş değilse...
            if (!username.isEmpty() || !password.isEmpty()) {
                // Burada böyle username ve password var mı diye kontrol sağlıyoruz.
                if (verify(username, password)) {
                    frame.dispose();

                    switch (getOpValueByUsername(username)) {
                        case 0:

```

Burada buttona bir eylem aktardım ve buton ismini Giriş yap olarak değiştirdim. Açılan pencereden username ve password bilgilerini alıp boş olup olmadığını kontrolunu sağladım. Daha sonrasında bir verify böyle bir kullanıcı olup olmadığına baktım. Bundan sonra artık bu kullanıcının op (Yetkisine) kontrol etmem gereklili. Kullanıcıyı yoksa yönetici mi diye bunuda getOpValueByUsername methoduyla yani

```

1 usage
public static int getOpValueByUsername(String username) {
    String sql = "SELECT Op FROM Teacher WHERE Username = ?";
    try {
        return jdbcTemplate.queryForObject(sql, Integer.class, username);
    } catch (EmptyResultDataAccessException e) {
        return 0;
    }
}

```

Bu kod parçasından alıyorum. Eğer 1 ise bu kullanıcı Tam Yetkili eğer 0 ise bu kullanıcı Kısıtlı yani bir öğretmen olduğunu gösteriyorum.

```

switch (getOpValueByUsername(username)) {
    case 0:
        // Pencere oluşturuyoruz. ve öğretmen penceresinde hiçbir eylem yapamayacak. CreateFrame de belirttim.

        String usernameSql = "SELECT id FROM Teacher Where Username = ?";
        int id = jdbcTemplate.queryForObject(usernameSql, Integer.class, username);
        System.out.println(id);
        ArrayList fieldName = new ArrayList();
        fieldName.add("jComboBoxEpisode");
        fieldName.add("jComboBoxLessons");
        fieldName.add("jComboBoxTeacher");
        fieldName.add("jComboBoxDay");
        fieldName.add("jComboBoxLessonTime");
        fieldName.add("jComboBoxClassroom");
        fieldName.add("Lesson_Code");
        fieldName.add("Classroom_Number");

        String tabloSql = "Select id, Episode, Lessons, Teacher, Day, LessonTime, Classroom, Lesson_Code, Classroom_Number" +
                " from academic_program Where Teacher = '" + id + "' ORDER BY Episode";
        createFrame.main( frameName: username + " - Öğretmen Ders Programı", frameSizeX: 1000, frameSizeY: 600, fieldName, rows: 4,
                frameType: "loginUsername", tabloSql );
        break;
    case 1:
        // tam yetki tüm konfigurasyonları yapabileceği menüye atıyorum.
        opMenu.opPermission();
        break;
    }
} else {
    showMessage.main("Kullanıcı adı veya şifreniz hatalı.");
}
} else {
    showMessage.main("Lütfen boş bırakmayın.");
}

```

Buradan 0 (Kısıtlı Yetki), 1 (Tam Yetki) yetki türüne göre pencerelere atıyorum. Ve gerekli hata mesajlarım ekranı göründüğü gibi eklenmiştir.

Menü Oluşturma :

Asıl konumuzdan birisi menü oluşturma, menü oluşumlarını en basit olacak şekilde ayarlamaya çalıştım. Şimdi ufak bir kod parçasından ilerleyerek nasıl oluşturulmasına bakalım...

Bilgilendirme (?)

Label Penceredeki Yazı

Field Pencerede yazı girilen kutu

ComboBox Pencerede seçim yaptıran kutu

```

frame.dispose();
ArrayList fieldName = new ArrayList();
fieldName.add("jComboBoxTitle");
fieldName.add("jComboBoxOp");
fieldName.add("TC");
fieldName.add("PhoneNumber");
fieldName.add("Name");
fieldName.add("Surname");
fieldName.add("Username");
fieldName.add("Password");
createFrame.main( frameName: "Ünvan İşlemleri", frameSizeX: 800, frameSizeY: 600, fieldName,
        rows: 4, frameType: "Teacher", tabloSql: "Select * from Teacher ORDER BY Username");

```

Burada öğretmenler penceresini oluşturmuşuz. İlk önce mevcut frame kapatmak için frame.dispose() methodunu çağirdım. Daha sonrasında fieldName adında bir ArrayList oluştururdum. Oluşturduğum Arryliste Eklemelemeleri belirtiyorum. Mesela öğretmenler tablosunda Title(Ünvan), Op(Yetkisi),TC,PhoneNumber(Telefon Numarası), Name(isim), Surname(Soyisim),(Username) kullanıcı adı, password(Sifre) bilgilerini istiyorum. İngilizce olmasına bakmayın databasenizde table isimlerini yazmanız lazım. Ben tercihen ingilizce kullandığım için ingilizce yazdım. Şimdi burada şunu dile getiriyoruz herhangi bir kısıtlama olmadığı zaman bu Fieldleri oluştur ve Labelerininide haliyle otomatik oluşturacak. Fakat biz bunu Field degilde comboBox olmasını istersek eğer kelimenin başına jComboBox yazmak zorundayız.

Bu ArrayList sayesinde

| | | | |
|----------|---------------------|-------------|-----------|
| Title | Araştırma Görevlisi | Op | Tam Yetki |
| TC | | PhoneNumber | |
| Name | | Surname | |
| Username | | Password | |

Böyle bir görüntüye ulaşabiliyoruz. Pencerenin en üst kısmı şunda bu şekilde oluşturulmuştur. Daha sonrasında FrameName yani pencerenin sol üst kısmında gözükecek isim, Pencerenin büyüğünü gelirsektçe genişlik olarak frameSizeX, yükseklik olarak frameSizeY değerlerini giriyoruz. Daha sonrasında Arrylistimizi ekliyoruz. Rows yani kaç stün olacak 4 stün olarak yapılmasını istedik ve bu şekilde max 4 stün olacak şekilde bir görüntü verdi. Daha sonrasında bu olaya bir FrameType isim belirliyoruz. Daha sonra aşağıda oluşacak Table için sql sorgumuzu yazıyoruz.

| ID | Title | Op | TC | PhoneNumber | Name | Surname | Username | Password |
|----|-----------------|---------------|-------------|-------------|-----------------|-----------|------------------|-----------------|
| 26 | Öğretim Gör... | Kısıtlı Yetki | 89012321098 | 89012321098 | Aslıhan | Salın | aslıhansalin | 21312312 |
| 28 | Öğretim Gör... | Kısıtlı Yetki | 01232109876 | 01232109876 | Ayşe | Kik | Aysekik | awe312 |
| 27 | Profesör | Kısıtlı Yetki | 90123210987 | 90123210987 | Cihançır | Güven | cihanginguven | 123123123 |
| 23 | Doç Dr. | Kısıtlı Yetki | 56789012321 | 56789012321 | Derya | Kayma | deryakayma | deryakayma |
| 41 | Öğretim Göre... | Kısıtlı Yetki | 56789012345 | 56789012345 | Didem | Çavuşoğlu | didemcavuso... | didemcavuso... |
| 24 | Öğretim Göre... | Kısıtlı Yetki | 67890123210 | 67890123210 | Elif | Baş | elifbas | elifbas |
| 1 | Öğretim Gör... | Tam Yetki | 12345678901 | 12345678901 | Engin | Uçar | enginucar | enginucar |
| 22 | Öğretim Gör... | Kısıtlı Yetki | 45678901232 | 45678901232 | Ergün | Metin | ergunmetin | ergunmetin |
| 25 | Öğretim Gör... | Kısıtlı Yetki | 78901232109 | 78901232109 | Gözde | Yılmaz | gozdeyilmaz | gozdeyilmaz |
| 21 | Öğretim Gör... | Kısıtlı Yetki | 34509876543 | 34509876543 | Hilal | Cura | hilalcura | hilalcura |
| 39 | Öğretim Göre... | Kısıtlı Yetki | 89012345678 | 89012345678 | İdris | Yağmur | idrisyagmur | idrisyagmur |
| 19 | Öğretim Gör... | Kısıtlı Yetki | 12345098765 | 12345098765 | İsmail Altuğ | Baysak | ismailaltugba... | ismailaltugb... |
| 42 | Öğretim Göre... | Kısıtlı Yetki | 78901234567 | 78901234567 | Kübra Göksu ... | Özbek | kubragoksuk... | kubragoksuk... |
| 40 | Öğretim Gör... | Kısıtlı Yetki | 23456789012 | 23456789012 | Levent | Karakuz | leventkarakuz | leventkarakuz |
| 12 | Öğretim Gör... | Kısıtlı Yetki | 45678901234 | 45678901234 | Mesude | Uçar | mesudeucar | mesudeucar |
| 20 | Öğretim Gör... | Kısıtlı Yetki | 23450987654 | 23450987654 | Muhammed ... | Vahşi | muhammed... | muhammed... |
| 3 | Öğretim Gör... | Kısıtlı Yetki | 34567890123 | 34567890123 | Murat | Albayrak | muratalbayrak | muratalbayrak |
| 17 | Öğretim Gör... | Kısıtlı Yetki | 90123456789 | 90123456789 | Mustafa | Çelik | mustafacelik | mustafacelik |
| 14 | Öğretim Göre... | Kısıtlı Yetki | 67890123456 | 67890123456 | Sezai | Bahar | sezaibahar | sezaibahar |
| 18 | Öğretim Gör... | Kısıtlı Yetki | 01234567890 | 01234567890 | Uğur | Bilgen | ugurbilgen | ugurbilgen |

ArrayList ve Sql sorgumuz bağlantılı sırasıyla belirtilmeli dir. Mesela table'mızde ilk gelen Arrylist ilk eklenen, Tabloda da ilk gözüken olur. Sırasını değiştirmek isterseniz * ifadesi yerine tek tek sırasını yazmak zorundasınız.

Şuan güncel olarak

Öğretmen İşlemleri

| Title | Araştırma Görevlisi | Op | Tam Yetki | | | | | |
|----------|---------------------|---------------|-------------|-------------|-----------------|-----------|------------------|-----------------|
| TC | | PhoneNumber | | | | | | |
| Name | | Surname | | | | | | |
| Username | | Password | | | | | | |
| ID | Title | Op | TC | PhoneNumber | Name | Surname | Username | Password |
| 26 | Öğretim Gör... | Kısıtlı Yetki | 89012321098 | 89012321098 | Aslıhan | Salın | aslihansalin | 21312312 |
| 28 | Öğretim Gör... | Kısıtlı Yetki | 01232109876 | 01232109876 | Ayşe | Kik | Ayşekik | awe312 |
| 27 | Profesör | Kısıtlı Yetki | 90123210987 | 90123210987 | Cihangir | Güven | cihanginguven | 123123123 |
| 23 | Doç Dr. | Kısıtlı Yetki | 56789012321 | 56789012321 | Derya | Kayma | deryakayma | deryakayma |
| 41 | Öğretim Göre... | Kısıtlı Yetki | 56789012345 | 56789012345 | Didem | Çavuşoğlu | didemcavuso... | didemcavuso... |
| 24 | Öğretim Göre... | Kısıtlı Yetki | 67890123210 | 67890123210 | Elif | Baş | elifbas | elifbas |
| 1 | Öğretim Gör... | Tam Yetki | 12345678901 | 12345678901 | Engin | Uçar | enginucar | enginucar |
| 22 | Öğretim Göre... | Kısıtlı Yetki | 45678901232 | 45678901232 | Ergün | Metin | ergunmetin | ergunmetin |
| 25 | Öğretim Gör... | Kısıtlı Yetki | 78901232109 | 78901232109 | Gözde | Yılmaz | gozdeyilmaz | gozdeyilmaz |
| 21 | Öğretim Gör... | Kısıtlı Yetki | 34509876543 | 34509876543 | Hilal | Cura | hilalcura | hilalcura |
| 39 | Öğretim Göre... | Kısıtlı Yetki | 89012345678 | 89012345678 | İdris | Yağmur | idrisyagmur | idrisyagmur |
| 19 | Öğretim Gör... | Kısıtlı Yetki | 12345098765 | 12345098765 | İsmail Altuğ | Baysak | ismailaltugba... | ismailaltugb... |
| 42 | Öğretim Gör... | Kısıtlı Yetki | 78901234567 | 78901234567 | Kübra Göksu ... | Özbek | kubragoksuk... | kubragoksuk... |
| 40 | Öğretim Gör... | Kısıtlı Yetki | 23456789012 | 23456789012 | Levent | Karakuz | leventkarakuz | leventkarakuz |
| 12 | Öğretim Gör... | Kısıtlı Yetki | 45678901234 | 45678901234 | Mesude | Uçar | mesudeucar | mesudeucar |
| 20 | Öğretim Göre... | Kısıtlı Yetki | 23450987654 | 23450987654 | Muhammed ... | Vahşi | muhammed... | muhammed... |
| 3 | Öğretim Gör... | Kısıtlı Yetki | 34567890123 | 34567890123 | Murat | Albayrak | muratalbayrak | muratalbayrak |
| 17 | Öğretim Gör... | Kısıtlı Yetki | 90123456789 | 90123456789 | Mustafa | Çelik | mustafacelik | mustafacelik |
| 14 | Öğretim Göre... | Kısıtlı Yetki | 67890123456 | 67890123456 | Sezai | Bahar | sezaibahar | sezaibahar |
| 18 | Öğretim Gör... | Kısıtlı Yetki | 01234567890 | 01234567890 | Üğur | Bilgen | ugurbilgen | ugurbilgen |

Bu fotoğrafta gözükecek kısma kadar geldik. Şimdi daha kodlama kısımlarına geçip anlatacağım.

```
19 usages
public class CreateFrame {

    public static void main(String frameName, int frameSizeX, int frameSizeY, ArrayList fieldNames, int rows, String frameType, String tabloSql) {

        frame = new JFrame(frameName);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(frameSizeX, frameSizeY);
        frame.setLocationRelativeTo(null);

        JPanel panel = new JPanel(new GridLayout(rows, cols));
        JPanel buttonPanel = new JPanel(new FlowLayout());
```

CreateFrame.main() methodu parametreleri ile bu şekilde çalışır. Frame verilen bilgilere göre oluşturur. Daa sonrasında sırasıyla ilk önce almış olduğumuz ArryListlerimizi

```

// Arrylistin içine Combobox, Field ve Label oluşturur.
for (Object field : fieldNames) {
    String fieldName = (String) field;

    // Burada oluşturmasını istemediğimiz FrameType'lerde durum belirtiyoruz. Mesela EpisodeCourses ve TeacherCourses Type'lerinde Code
    // Varsa oluşturmasını ve LoginUsername Type'sinde de oluşturmasını istiyoruz.
    if ((frameType.equals("episodeCourses") || frameType.equals("teacherCourses")) && fieldName.equals("Code"))
        || frameType.equals("loginUsername")) {

        // Eğer başında jComboBox varsa....
    } else if (fieldName.startsWith("jComboBox")) {
        // jComboBox siliyorum...
        String methodName = fieldName.substring( beginIndex: 9 );
        // JComboBox oluşturuyorum...
        JComboBox comboBox = new JComboBox(<>(generated(methodName).toArray(new String[0])));
        // ComboBoxEntry içine ekliyoruz.
        ComboBoxEntry comboBoxEntry = new ComboBoxEntry(fieldName, comboBox);
        comboBoxEntries.add(comboBoxEntry);
        // Panel'e ekliyoruz. Labelini
        panel.add(new JLabel(fieldName.substring( beginIndex: 9 )));
        // Daha sonra ComboBox'u
        panel.add(comboBox);
    } else {
        // Entry eklemeye çalışıyorum
        JTextField textField = new JTextField();
        TextFieldEntry entry = new TextFieldEntry(fieldName, textField);
        textFieldEntries.add(entry);

        // Panel'e ekleme
        panel.add(new JLabel(fieldName));
        panel.add(textField);
    }
}

```

Buradan özelliğine göre çözümleyip menüye ekliyoruz. Detayları başlıklarda anlatacağım... Şimdi butonları nasıl menüye ekleyeceğimizi göstereceğim... CreateFrame classında

```

// LoginUsername = Normal öğretmen girdiğinde butonları eklememesi için...
if (!frameType.equals("loginUsername")) {

    // Eğer Academic program öğretmen veya Bölümü görevse.... Özel çıktı buton ekle
    if (frameType.equals("academic_program_episode") || frameType.equals("academic_program_teacher")) {
        JButton excelCKT = excelCKT(frameType, frameName);
        buttonPanel.add(excelCKT);
    }

    // Normal çıktı butonu ekleme
    JButton pdfButton = excelcikti(fieldNames, frameType, tabloSql);
    buttonPanel.add(pdfButton);

    // Eğer Gün, Bölüm ve Öğretmene ders atama değilse güncelleme butonu ekle
    if (!frameType.equals("episodeCourses") && !frameType.equals("teacherCourses") && !frameType.equals("Days")) {
        JButton updateButton = updateButton(fieldNames, frameType, tabloSql);
        buttonPanel.add(updateButton);
    }

    // Kayıt etme butonu
    JButton saveButton = saveButton(fieldNames, frameType, tabloSql);
    buttonPanel.add(saveButton);

    // Silme Butonu
    JButton removeButton = removeButton(fieldNames, frameType, tabloSql);
    buttonPanel.add(removeButton);

    // Geri Butonu
    JButton backButton = backButton();
    buttonPanel.add(backButton);

    // Eğer Gün ise ileri git butonu ekle
    if (frameType.equals("Days")) {
        JButton nextButton = nextButton();
        buttonPanel.add(nextButton);
    }
}

```

Bu şekilde butonları kısıtlayarak veya normal bir şekilde ekleyebilirsiniz..

Field Oluşturma:

Field oluşturmamızda çok fazla detayımız yok aslında

```
/** TextFieldEntry sınıfı, bir metin alanı ve alan adını içeren ögeleri temsil eder.*/
14 usages
public class TextFieldEntry {
    2 usages
    private String fieldName;
    3 usages
    private JTextField textField;
    /** TextFieldEntry sınıfının yapıcı metodudur.
     * @param fieldName Metin alanının adını temsil eder.
     * @param textField JTextField türündeki metin alanını temsil eder.*/
    1 usage
    public TextFieldEntry(String fieldName, JTextField textField) {
        this.fieldName = fieldName;
        this.textField = textField;
    }
    /**
     * Metin alanının adını getiren metod.
     * @return Metin alanının adı.
     */
    >    public String getFieldName() { return fieldName; }
    /** Metin alanını getiren metod.
     * @return JTextField türündeki metin alanı.*/
    4 usages
    >    public JTextField getTextField() { return textField; }
    /** Metin alanının değerini ayarlayan metod.
     * @param value Ayarlanacak metin alanı değeri.*/
    1 usage
    >    public void setTextFieldValue(String value) { textField.setText(value); }
    /** Verilen alan adına sahip TextFieldEntry ögesini bulan ve döndüren statik metod.
     * @param fieldName Bulunmak istenen metin alanının adı.
     * @return Eslesen metin alanı bulunursa TextFieldEntry ögesi, aksi takdirde null.*/
    5 usages
    public static TextFieldEntry findTextFieldEntry(String fieldName) {
        for (TextFieldEntry textFieldEntry : textFieldEntries) {
            if (textFieldEntry.getFieldName().equals(fieldName)) {
                return textFieldEntry;
            }
        }
        return null;
    }
}
```

ComboBox Oluşturma:

ComboBox oluşturma methodumuz aslında biraz uzun çünkü veri kaydetmesi mevcut...

```
/** ComboBoxEntry sınıfı, bir JComboBox ögesini ve adını içeren öğeleri temsil eder. */
13 usages
public class ComboBoxEntry {
    2 usages
    private String comboBoxName;
    3 usages
    private JComboBox<String> comboBox;
    /** ComboBoxEntry sınıfının yapıcı metodudur.
     * @param comboBoxName JComboBox'in adını temsil eder.
     * @param comboBox JComboBox türündeki öğeyi temsil eder.*/
    1 usage
    public ComboBoxEntry(String comboBoxName, JComboBox<String> comboBox) {
        this.comboBoxName = comboBoxName;
        this.comboBox = comboBox;
    }
    /** JComboBox'in adını getiren metod.
     * @return JComboBox'in adı */
    1 usage
    public String getComboBoxName() { return comboBoxName; }
    /** JComboBox'i getiren metod.
     * @return JComboBox türündeki öğe.*/
    2 usages
    public JComboBox<String> getComboBox() { return comboBox; }
    /** JComboBox'in seçili değerini ayarlayan metod.
     * @param value Ayarlanacak seçili değer.*/
    1 usage
    public void setComboBoxSelectedValue(String value) { comboBox.setSelectedItem(value); }

    /** Verilen comboBoxName'e sahip ComboBoxEntry ögesini bulan ve döndüren statik metod.
     * @param comboBoxName Bulunmak istenen JComboBox'in adı.
     * @return Eşleşen JComboBox bulunursa ComboBoxEntry ögesi, aksi takdirde null. */
    2 usages
    public static ComboBoxEntry findComboBoxEntry(String comboBoxName) {
        for (ComboBoxEntry comboBoxEntry : comboBoxEntries) {
            if (comboBoxEntry.getComboBoxName().equals(comboBoxName)) {
                return comboBoxEntry;
            }
        }
        return null;
    }
}
```

Print Screen

Bu şekilde öğeleri oluşturduktan sonra,

```
// JComboBox oluşturuyoruz...
JComboBox comboBox = new JComboBox<>(generated(methodName).toArray(new String[0]));
```

Bu kısımda Combobox oluşturkenki generated methoduna gidiyoruz..

```
public class comboBox {

    // Field Name göre mesela JComboBoxOp olan fieldName op olarak buradan bize gönderecek.
    // ve bizde buna ait en uygun yapıyı Switch ile bulacağız...
    2 usages
    public static ArrayList generated(String fieldName) {
        switch (fieldName) {
            case "Op":
                return permission();
            case "Branch":
                return branch();
            case "LessonTime":
                return lessonTime();
            case "Title":
                return arrayList( Data: "Name", Table: "Title");
            case "Teacher":
                return arrayList( Data: "Username", Table: "Teacher");
            case "Episode":
                return arrayList( Data: "Name", Table: "Episode");
            case "Lessons":
                return arrayList( Data: "Name", Table: "Lesson");
            case "Classroom":
                return arrayList( Data: "Type", Table: "Classroom");
            case "Lesson":
                return lessonCode();
            case "Day":
                return daysName();
            default:
                return new ArrayList<>();
        }
    }
}
```

Buradan ilk önce gelen veriyi alacağız ve return olarak bir liste göndereceğiz. Buradan Swich içerisinde comboBox içerisindeki veriyi oluşturabiliriz.

Tablo Özelleştirme:

Menümüzde olacak tablo hakkında kullandığım kod blokları ve açıklamaları,

```

public class tabloModel {
    /**
     * CustomTableModel sınıfı, özel tablo modellerini temsil eder ve DefaultTableModel'i genişletir.
     */
    2 usages
    public static class CustomTableModel extends DefaultTableModel {
        /**
         * CustomTableModel sınıfının yapıcı metodudur.
         * @param columnNames Tablo sütun isimlerini içeren vektör.
         * @param rowCount Satır sayısını temsil eden bir tamsayı.
         */
        1 usage
        public CustomTableModel(Vector<String> columnNames, int rowCount) {
            super(columnNames, rowCount);
        }
        /**
         * Belirtilen hücrenin düzenlenebilir olup olmadığını belirleyen metod.
         * @param row Hücrenin bulunduğu satır indeksi.
         * @param column Hücrenin bulunduğu sütun indeksi.
         * @return Hücrenin düzenlenebilir olup olmadığını gösteren boolean değer.
         */
        @Override
        public boolean isCellEditable(int row, int column) {
            return false;
        }
    }
}

```

Ve tekrardan createFrame methodunu çağırırken frame'lerden FieldNames arraylistini kullanarak burada tablo oluşturur ve sonuç olarak

| ID | Title | Op | TC | PhoneNumber | Name | Surname | Username | Password |
|----|-------|----|----|-------------|------|---------|----------|----------|
|----|-------|----|----|-------------|------|---------|----------|----------|

Bu kısmı kodladık.. şimdi içine bilgi ekleme kısmına geliyoruz...

```
5 usages
public class reloadTablo {
    /**
     * Yeniden yükleme işlemlerini gerçekleştiren bir yardımcı sınıf.
     */
    10 usages
    public static void reload(ArrayList fieldNames, String tabloSql) {

        // Sütun isimlerini al
        ArrayList columnList = getColumnList.main(fieldNames);
        // SQL tablosundan veri al
        ArrayList loadArrayList = returnListe.main(tabloSql);
        // Tablo modelinin satır sayısını sıfırla
        tableModel.setRowCount(0);
        // Verileri tabloya ekle
        for (int i = 0; i < loadArrayList.size(); i = i + columnList.size()) {
            Vector<Object> row = new Vector<Object>();
            for (int j = 0; j < columnList.size(); j++) {
                String field = (String) columnList.get(j);
                row.add(tableComboBox.main(field, loadArrayList.get(j + i)));
            }
            tableModel.addRow(row);
        }
    }
}
```

Bu kod bloğu sayesinde artık tablomuzun güncel görüntüsü

Öğretmen İşlemleri

| Title | Araştırma Görevlisi | | Op | | | Tam Yetki | | |
|-------|---------------------|---------------|-------------|-------------|-----------------|-----------|------------------|-----------------|
| | | | | PhoneNumber | | | | |
| | | | | Surname | | | | |
| | | | | Password | | | | |
| ID | Title | Op | TC | PhoneNumber | Name | Surname | Username | Password |
| 26 | Öğretim Gör... | Kısıtlı Yetki | 89012321098 | 89012321098 | Aslıhan | Salın | aslihansalin | 21312312 |
| 28 | Öğretim Gör... | Kısıtlı Yetki | 01232109876 | 01232109876 | Ayşe | Kik | Ayşekik | awe312 |
| 27 | Profesör | Kısıtlı Yetki | 90123210987 | 90123210987 | Cihangir | Güven | cihanginguven | 123123123 |
| 23 | Doç Dr. | Kısıtlı Yetki | 56789012321 | 56789012321 | Derya | Kayma | deryakayma | deryakayma |
| 41 | Öğretim Göre... | Kısıtlı Yetki | 56789012345 | 56789012345 | Didem | Çavuşoğlu | didemcavuso... | didemcavuso... |
| 24 | Öğretim Göre... | Kısıtlı Yetki | 67890123210 | 67890123210 | Elif | Baş | elifbas | elifbas |
| 1 | Öğretim Gör... | Tam Yetki | 12345678901 | 12345678901 | Engin | Uçar | enginucar | enginucar |
| 22 | Öğretim Gör... | Kısıtlı Yetki | 45678901232 | 45678901232 | Ergün | Metin | ergunmetin | ergunmetin |
| 25 | Öğretim Gör... | Kısıtlı Yetki | 78901232109 | 78901232109 | Gözde | Yılmaz | gozdeyilmaz | gozdeyilmaz |
| 21 | Öğretim Gör... | Kısıtlı Yetki | 34509876543 | 34509876543 | Hilal | Cura | hilalcura | hilalcura |
| 39 | Öğretim Gör... | Kısıtlı Yetki | 89012345678 | 89012345678 | İdris | Yağmur | idrisyagmur | idrisyagmur |
| 19 | Öğretim Gör... | Kısıtlı Yetki | 12345098765 | 12345098765 | İsmail Altuğ | Baysak | ismailaltugba... | ismailaltugb... |
| 42 | Öğretim Gör... | Kısıtlı Yetki | 78901234567 | 78901234567 | Kübra Göksu ... | Özbek | kubragoksuk... | kubragoksuk... |
| 40 | Öğretim Gör... | Kısıtlı Yetki | 23456789012 | 23456789012 | Levent | Karakuz | leventkarakuz | leventkarakuz |
| 12 | Öğretim Gör... | Kısıtlı Yetki | 45678901234 | 45678901234 | Mesude | Uçar | mesudeucar | mesudeucar |
| 20 | Öğretim Gör... | Kısıtlı Yetki | 23450987654 | 23450987654 | Muhammed ... | Vahşi | muhammed... | muhammed... |
| 3 | Öğretim Gör... | Kısıtlı Yetki | 34567890123 | 34567890123 | Murat | Albayrak | muratalbayrak | muratalbayrak |
| 17 | Öğretim Gör... | Kısıtlı Yetki | 90123456789 | 90123456789 | Mustafa | Çelik | mustafacelik | mustafacelik |
| 14 | Öğretim Gör... | Kısıtlı Yetki | 67890123456 | 67890123456 | Sezai | Bahar | sezaibahar | sezaibahar |
| 18 | Öğretim Gör... | Kısıtlı Yetki | 01234567890 | 01234567890 | Uğur | Bilgen | ugurbilgen | ugurbilgen |

bu şekilde oluşturulmuştur. Şimdi sırayla butonlara gelelim...

Butonlar:

Şimdi size sırayla butonların nasıl çalıştığını anlatacağım. Sanırım en uzun kısımlara gelmiş bulunmaktaiz.

Geri Git:

Aslında en kısa butonlarımızdan birtanesini şuanda size anlatacağım..

```
2 usages
public class backButton {
    3 usages
    public static JButton backButton() {
        JButton button = new JButton( text: "Geri Git");
        button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Frameyi Kapatıp...
                frame.dispose();
                // Yeni pencereyi açıyoruz..
                opPermission();
            }
        });
        return button;
    }
}
```

Veriyi Sil:

```
public class removeButton {  
    /**  
     * Veriyi silme işlemlerini gerçekleştirilen JButton'i oluşturan metod.  
     * @param fieldNames tablo alan isimlerini içeren ArrayList.  
     * @param process ilgili işlemi temsil eden bir String.  
     * @param tabloSql Verilerin alındığı SQL sorgusunu içeren bir String.  
     * @return Veriyi silme işlemlerini gerçekleştirilen JButton.  
     */  
    2 usages  
    public static JButton removeButton(ArrayList fieldNames, String process, String tabloSql) {  
        JButton button = new JButton( text: "Veriyi Sil");  
        button.addActionListener(new ActionListener() {  
            @Override  
            public void actionPerformed(ActionEvent e) {  
                // Sql Sorgumuz  
                String sql = "DELETE FROM " + tableName(process) + " WHERE id = ?";  
  
                // Secilen stün alıyoruz  
                int selectedRow = table.getSelectedRow();  
  
                // Eğer Secildiyse...  
                if (selectedRow != -1) {  
                    // Stün 0. indexi id olduğu için 0. indexi alıyoruz..  
                    int id = Integer.parseInt(tableView.getValueAt(selectedRow, column: 0).toString());  
                    //Sql sorgumuzu çalıştırıldı.  
                    jdbcTemplate.update(sql, id);  
                    showMessage.main("Veri tablodan silindi.");  
                    reload(fieldNames, tabloSql);  
                } else {  
                    showMessage.main("Lütfen tablodan bir veri seçin.");  
                }  
            }  
        });  
        return button;  
    }  
  
    /**  
     * İşlem adına göre tablo adını döndüren özel bir metod.  
     * @param Process işlem adına temsil eden bir String.  
     * @return işlem adına göre belirlenen tablo adı.  
     */  
    1 usage  
    private static String tableName(String Process) {  
  
        if (Process.equals("academic_program_episode") || Process.equals("academic_program_teacher")) {  
            Process = "academic_program";  
        }  
        return Process;  
    }  
}
```

Veriyi Kayıt Et:

Veriyi kayıt etmek için belirli şartlar ve olaylar vardır. Sırayla inceleyelim..

```

    * Veri kaydetme islemlerini gerçeklestiren JButton'i olusturan metod.
    * @param fieldNames Tablo alan isimlerini içeren ArrayList.
    * @param frameType Pencere türünü temsil eden bir String.
    * @param tabloSql Verilerin olduğu SQL sorgusunu içeren bir String.
    * @return Veri kaydetme islemlerini gerçeklestiren JButton.
*/
2 usages
public static JButton saveButton(ArrayList fieldNames, String frameType, String tabloSql) {
    JButton button = new JButton(text: "Kayıt Et");
    button.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {

            // Burada field içindeki verileri fieldData Kayıt ediyoruz.
            ArrayList fieldData = new ArrayList<>();

            // TextFieldEntry listesini döngüye alarak text alanlarındaki verileri al
            for (TextFieldEntry textFieldEntry : textFieldEntries) {
                String value = textFieldEntry.getTextField().getText();
                fieldData.add(value);
            }

            // ComboBoxEntry listesini döngüye alarak combo box alanlarındaki verileri al
            for (ComboBoxEntry comboBoxEntry : comboBoxEntries) {
                String value = comboBoxEntry.getComboBox().getSelectedItem().toString();
                // Özel durumlar için kontrol ekleyebilirsiniz
                if ((frameType.equals("teacherCourses") || frameType.equals("episodeCourses")) && value.startsWith("(")) {
                    String[] slot = value.split(regex: " ", limit: 2);
                    System.out.println(fieldData.get(0));
                    fieldData.add(fieldData.get(0));
                    fieldData.set(0, slot[0].substring(1, slot[0].length() - 1));
                    fieldData.add(slot[1]);
                    continue;
                }
                fieldData.add(value);
            }
            // Kayıt etme islemiini gerçeklestiren metod
            saveData(fieldData, fieldNames, frameType, tabloSql);
        }
    });
    return button;
}

```

Print Screen

Şimdi saveData methoduna gelelim...

```

1 usage
private static void saveData(ArrayList fieldData, ArrayList fieldNames, String frameType, String tabloSql) {

    // İlk önce Arrylistimizi düzenliyoruz ve sıralıyoruz (önce field sonra jComboBox)
    ArrayList fieldOrder = getFieldOrder(fieldNames);

    // Buradan kontroller yapıyoruz, Eklenmesinde sakincalar varmı diye bakıyoruz..
    if (valid(fieldData, fieldOrder, frameType, id: 0, update: false)) {

        // Arrylist içindeki verileri düzenleyeceğiz mesela Tam Yetki comboBox var ama biz bunu
        // Veritabanında 1 olarak kayıt edeceğiz bunları düzenleyeceğ.. .
        dataComboBox.main(fieldData, fieldOrder);

        // Kaydi tamamlıyoruz ve sql sorgusunu göndermek için hazırladık verileri...
        executeInsertQuery(fieldData, fieldOrder, frameType);

        // Tabloyu yeniliyoruz..
        reload(fieldNames, tabloSql);
    }
}

```

getFieldOrder methodumuz..

```
public class getFieldOrder {

    /**
     * Alan sıralamasını düzenleyen metod.
     * @param fieldNames Tablo alan isimlerini içeren ArrayList.
     * @return Düzenlenmiş alan sıralamasını içeren ArrayList.
     */
    4 usages
    public static ArrayList getFieldOrder(ArrayList fieldNames) {

        // Buradan önce Field isimlerini alıyoruz daha sonra comboBox verilerinin ismini alıyoruz.
        // Bunun amacı jComboBox yazısından kurtulmak ve sıralamak.

        ArrayList<String> fieldOrder = new ArrayList<>();

        // Field isimlerini ekleyen döngü
        for (Object fieldName : fieldNames) {
            String field = (String) fieldName;
            if (!field.startsWith("jComboBox")) {
                fieldOrder.add(field);
            }
        }

        // ComboBox verilerinin isimlerini ekleyen döngü
        for (Object fieldName : fieldNames) {
            String field = (String) fieldName;
            if (field.startsWith("jComboBox")) {
                fieldOrder.add(field);
            }
        }

        return fieldOrder;
    }
}
```

valid (Kontrol) methodumuzda bütün kontrolleri sağlayacağız... Fakat eğer öğe eklicecekse False öğe düzenlicksek true olmalı buna lütfen dikkat edin...

```
4 usages
public static boolean valid(ArrayList fieldData, ArrayList fieldOrder, String frameType, Object id, Boolean update) {

    for (int i = 0; i < fieldData.size(); i++) {

        String label = (String) fieldData.get(i);
        String field = (String) fieldOrder.get(i);

        // Boş olup olmadığını kontrol edeceğiz
        if (label.isEmpty() && !field.startsWith("jC")) {
            showMessage.main(field + " Metin boş bırakılamaz.");
            return false;
        }

        // Kelime uzunluğunu kontrolü
        if (label.length() > 35) {
            showMessage.main(field + " En fazla 35 Karakter kullanılabilir..");
            return false;
        }
    }
}
```

Mesela burada kelime uzunluğu ve boş olup olmadığını kontrol ediyoruz....

```

switch (frameType) {
    case "Title":
        // Başka ünvan varmı diye kontrol ediyoruz
        if (countRecords(frameType, field, label)) {
            showMessage.main(field + " zaten kullanılıyor.");
            return false;
        }
        break;
    case "Teacher":
        if (field.equals("TC") || field.equals("PhoneNumber")) {
            // uzunluğu 11 Rakamı diye kontrol ediyoruz
            if (label.length() != 11) {
                showMessage.main(field + " lütfen 11 rakamlı girin. (" + label.length() + ")");
                return false;
                // Baskası kullanıyorum diye bakiyoruz..
            } else if (countRecords(frameType, field, label)) {
                showMessage.main(field + " zaten kullanılıyor.");
                return false;
                // Harf giriip girmediğini kontrol ediyoruz...
            } else if (!containsNumericalDigit(label)) {
                showMessage.main(field + " bilgisine lütfen sadece rakam girin.");
                return false;
            }
        }
        if (field.equals("Username")) {
            // Kullanıcı adının daha önce kullanılmıştılarına bakıyoruz..
            if (countRecords(frameType, field, label)) {
                showMessage.main(field + " zaten kullanılıyor.");
                return false;
            }
        }
        break;
}

```

Mesela bunun gibi birçok kontrolleri buradan (Valid methodu) gerçekleştireceğiz...

Şimdi sonraki olayımız DataComboBox methodu burası çok önemli çünkü Foreing key olan verileri comboboxta adını gösteriyorduk fakat bunları id olarak kayıt edeceğimiz için verileri toplamamız lazım...

```
/**  
 * DataComboBox sınıfı, JComboBox alanlarındaki verileri düzenleyen yardımcı bir sınıfır.  
 */  
4 usages  
public class dataComboBox {  
  
    /**  
     * JComboBox alanlarındaki verileri düzenleyen metod.  
     * @param fieldData Kaudedilecek veriyi içeren ArrayList.  
     * @param fieldOrder Düzelenmiş alan sıralamasını içeren ArrayList.  
     */  
    public static void main(ArrayList fieldData, ArrayList fieldOrder) {  
  
        for (int i = 0; i < fieldOrder.size(); i++) {  
            String fieldName = (String) fieldOrder.get(i);  
            String value = (String) fieldData.get(i);  
            switch (fieldName) {  
                case "jComboBoxOp":  
                    fieldData.set(i, value.equals("Tam Yetki") ? "1" : "0");  
                    break;  
                case "jComboBoxBranch":  
                    fieldData.set(i, value.equals("Bölünecek") ? "2" : "1");  
                    break;  
                case "jComboBoxTitle":  
                case "jComboBoxClassroom":  
                case "jComboBoxEpisode":  
                case "jComboBoxLesson":  
                case "jComboBoxLessons":  
                case "jComboBoxTeacher":  
                    handleComboBoxField(fieldName, value, fieldData, i);  
                    break;  
            }  
        }  
    }  
}
```

Buradan özellikle toplayabiliriz bu verileri....

handleComboBoxField methoduna gelecek olursak...

```
/** JComboBox alanlarındaki verileri düzenleyen özel bir metod.
 * @param fieldName JComboBox alanının adını temsil eden bir String.
 * @param value JComboBox'ten alınan değeri temsil eden bir String.
 * @param fieldData Kullanılabilecek veriyi içeren ArrayList.
 * @param index ArrayList üzerindeki indeks değerini temsil eden bir tamsayı. */
1 usage
private static void handleComboBoxField(String fieldName, String value, ArrayList fieldData, int index) {
    String columnName;
    String whereName;
    switch (fieldName) {
        case "jComboBoxTitle":
        case "jComboBoxClassroom":
            columnName = fieldName.equals("jComboBoxTitle") ? "Title" : "Classroom";
            whereName = fieldName.equals("jComboBoxTitle") ? "Name" : "Type";
            break;
        case "jComboBoxEpisode":
        case "jComboBoxLesson":
            columnName = fieldName.equals("jComboBoxEpisode") ? "Episode" : "Lesson";
            whereName = fieldName.equals("jComboBoxEpisode") ? "Name" : "Code";
            if (fieldName.equals("jComboBoxLesson")) {
                value = (String) fieldData.get(0);}
            break;
        case "jComboBoxLessons":
            columnName = "Lesson";
            whereName = "Name";
            break;
        case "jComboBoxTeacher":
            columnName = "Teacher";
            whereName = "Username";
            break;
        default:
            return;
    }
    String[] parcalar = value.split( regex: "/");
    String sql = "SELECT id FROM " + columnName + " WHERE " + whereName + " = '" + parcalar[0] + "'";
    int result = jdbcTemplate.queryForObject(sql, Integer.class);
    fieldData.set(index, result);
}
```

Şimdi herşeyi düzenlediğimize göre bir Sql sorgusuna ihtiyacımız var bunuda artık

```

/**
 * GenerateInsertQuery sınıfı, veri ekleme sorgusunu oluşturan yardımcı bir sınıfır.
 * Veri ekleme sorgusunu oluşturan metod.
 * @param fieldOrder Düzenlenmiş alan sıralamasını içeren ArrayList.
 * @param process Flagili işlemi temsil eden bir String.
 * @return Oluşturulan veri ekleme sorgusu.
 */
1 usage
private static String generateInsertQuery(ArrayList<String> fieldOrder, String process) {
    // Burada ufak bir kısıtlama yapıyoruz..
    if (process.equals("academic_program_episode") || process.equals("academic_program_teacher")) {
        process = "academic_program";
    }
    // Sql sorgusunun ilk başını belirliyoruzz...
    StringBuilder sqlQuery = new StringBuilder("INSERT INTO " + process + "(");
    // Eğer bir Field ise düz ekle ve sonuna (,) koy mesela TC, gibi..
    for (String fieldName : fieldOrder) {
        if (!fieldName.startsWith("jComboBox")) {
            sqlQuery.append(fieldName).append(",");
        }
    }
    // Eğer jComboBoxName ise ilk 9 harfi yok et ve Name olarak yazdır...
    for (String fieldName : fieldOrder) {
        if (fieldName.startsWith("jComboBox")) {
            sqlQuery.append(fieldName.substring( beginIndex: 9)).append(",");
        }
    }
    // Oluşan son verideki son harfi sil
    sqlQuery.deleteCharAt( index: sqlQuery.length() - 1);
    // Oluşan veriye bunu ekle
    sqlQuery.append(") VALUES(");
    // Fieldorder kadar ?, koy ve döndür
    for (int i = 0; i < fieldOrder.size(); i++) {
        sqlQuery.append("?,");
    }
    // Son harfi sil virgül olacağlı için
    sqlQuery.deleteCharAt( index: sqlQuery.length() - 1);
    // Parantezi kapat.
    sqlQuery.append(")");
    return sqlQuery.toString();
}

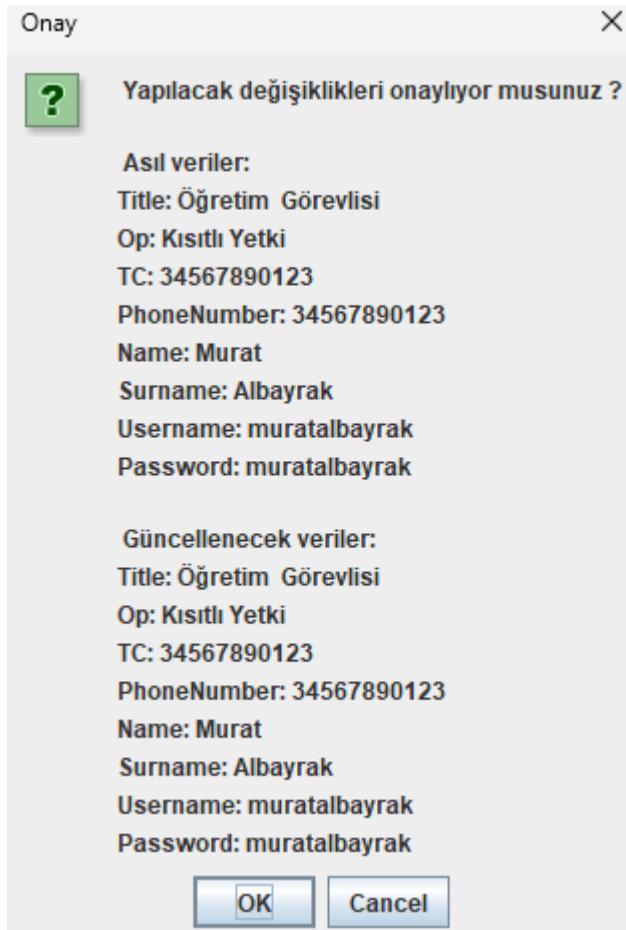
```

Buradan gerekli sql sorgusunu oluşturabiliriz ve artık veriyi kayıt edebiliriz...

Veriyi Güncelle:

Kayıt ederken aslında buradaki çoğu şeyi yaptığımız için güncellemenin üstünde çok fazla durmayacağım...

```
/**  
 * Veri kaydetme işlemlerini gerçekleştiren JButton'i oluşturan metod.  
 *  
 * @param fieldNames Tablo alan isimlerini içeren ArrayList.  
 * @param frameType Pencere türünü temsil eden bir String.  
 * @param tabloSql Verilerin alındığı SQL sorusunu içeren bir String.  
 * @return Veri güncelleme işlemlerini gerçekleştiren JButton.  
 */  
2 usages  
public static JButton updateButton(ArrayList fieldNames, String frameType, String tabloSql) {  
    JButton button = new JButton( text: "Güncelle");  
    button.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            // Buradan seçilen row kayıt ediyoruz..  
            int selectedRow = table.getSelectedRow();  
            // Eğer secildiyse...  
            if (selectedRow != -1) {  
                // Id değerini alıyoruz..  
                Object memberId = tableView.getModel().getValueAt(selectedRow, column: 0).toString();  
                // Burada field içindeki verileri fieldData Kayıt ediyoruz.  
                ArrayList fieldData = new ArrayList<>();  
                // TextFieldEntry listesini döngüye alarak text alanlarındaki verileri al  
                for (TextFieldEntry textFieldEntry : textFieldEntries) {  
                    String value = textFieldEntry.getTextField().getText();  
                    fieldData.add(value);  
                }  
                // ComboBoxEntry listesini döngüye alarak combo box alanlarındaki verileri al  
                for (ComboBoxEntry comboBoxEntry : comboBoxEntries) {  
                    String value = comboBoxEntry.getComboBox().getSelectedItem().toString();  
                    fieldData.add(value);  
                }  
                updateData(fieldData, fieldNames, frameType, memberId, tabloSql);  
            } else {  
                showMessage.main("Lütfen güncelenecek veriyi seçiniz.");  
            }  
        }  
    });  
    return button;  
}
```



Uyariya basıldığı zaman böyle bir yazı mesajı gelmesi lazım ki kullanıcı bilgileri kontrol edip onaylasın...

```

private static boolean showAlertly(ArrayList<Object> fieldData, ArrayList fieldNames, String frameType) {
    // Burada güncelleme ile ilgili mesaj yayınlanması içindir..
    // Mesajın ilk kısmını belirtiyorum...
    alertMesage = alertMesage + "Yapılacak değişiklikleri onaylıyor musunuz ?\n\n Asıl veriler:";

    int sayac = 0;

    // Asıl tablodan verileri çekerek belirtiyorum.
    int selectedRow = table.getSelectedRow();
    for (int i = 0; i < fieldNames.size(); i++) {
        String field = (String) fieldNames.get(i);
        if (field.startsWith("jCom")) {
            alertMesage = alertMesage + "\n" + field.substring( beginIndex: 9 ) + ":" + tableModel.getValueAt(selectedRow, column: i + 1).toString();
            sayac++;
        } else {
            alertMesage = alertMesage + "\n" + field + ":" + tableModel.getValueAt(selectedRow, column: i + 1).toString();
        }
    }

    // Güncellenecek verileri listeliyorum .
    alertMesage = alertMesage + "\n\n Güncellenecek veriler:";

    int c = sayac;
    for (int i = 0; i < fieldNames.size(); i++) {
        String field = (String) fieldNames.get(i);
        if (field.startsWith("jC")) {
            alertMesage = alertMesage + "\n" + field.substring( beginIndex: 9 ) + ":" + fieldData.get(fieldNames.size() - sayac);
            sayac--;
        } else {
            if (!frameType.equals("Lesson")) {
                alertMesage = alertMesage + "\n" + field + ":" + fieldData.get(i - c);
            } else {
                alertMesage = alertMesage + "\n" + field + ":" + fieldData.get(i);}}
```

// Ve bir uyarı penceresi açıyorum uyarı penceresine göre durum bildiriyorum .

```

    int result = JOptionPane.showConfirmDialog( parentComponent: null, alertMesage, title: "Onay", JOptionPane.OK_CANCEL_OPTION);
    boolean confirmed = false;
    if (result == JOptionPane.OK_OPTION) {
        confirmed = true;
    }
    return confirmed;
}

```

Onun içinde böyle bir kod blogu kullandım... sql sorgumuzda

```
1 usage
private static String generateInsertQuery(ArrayList<String> fieldOrder, String frameType) {

    // Ufak kısıtlamalarımız..
    if (frameType.equals("academic_program_episode") || frameType.equals("academic_program_teacher")){
        frameType = "academic_program" ;
    }

    // Sql sorgusunun ilk başını belirliyoruz...
    StringBuilder sqlQuery = new StringBuilder("UPDATE " + frameType + " SET ");

    // FieldNamesini ve = ? ardından bunu koyması beliriyoruz..
    for (String fieldName : fieldOrder) {
        if (!fieldName.startsWith("jComboBox")) {
            sqlQuery.append(fieldName).append(" = ?, ");
        }
    }

    for (String fieldName : fieldOrder) {
        if (fieldName.startsWith("jComboBox")) {
            sqlQuery.append(fieldName.substring( beginIndex: 9)).append(" = ?, ");
        }
    }

    // Daha sonra son 2 verisini silip Boşluk ve virgülü..
    sqlQuery.deleteCharAt( index: sqlQuery.length() - 2);

    // Nereden düzelteceğini gösteriyorum...
    sqlQuery.append("WHERE id = ?");

    return sqlQuery.toString();
}
```

bu sefer bu şekilde oluşturuyoruz...

Cıktı Al:

Cıktı alma işleminde normal cıktıda tablo gibi bir cıktı alabilmemizi sağlıyor. Aslında tablodaki veriyi cıktı olarak alıyoruz burada....
Gerekli kodları inceleyelim..

```
/*
 * ExcelOutputButton sınıfı, Excel çıktısı alma işlemelerini gerçekleştiren bir JButton oluşturur.
 */
1 usage
public class excelcikti {

    /**
     * Excel çıktısı alma işlemelerini gerçekleştiren JButton'i oluşturan metod.
     * @param fieldNames Tablo alan isimlerini içeren ArrayList.
     * @param frameType Pencere türünü temsil eden bir String.
     * @param tabloSql Verilerin alındığı SQL sorusunu içeren bir String.
     * @return Excel çıktısı alma işlemelerini gerçekleştiren JButton.
     */
2 usages
public static JButton excelcikti(ArrayList fieldNames, String frameType, String tabloSql) {
    JButton button = new JButton( text: "Çıktı Al");
    button.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {

            JFileChooser fileChooser = new JFileChooser();
            fileChooser.setDialogTitle("Dosya Yolu Seç");
            fileChooser.setCurrentDirectory(new File(System.getProperty("user.home")));

            int userSelection = fileChooser.showSaveDialog( parent: null);

            try {

                if (userSelection == JFileChooser.APPROVE_OPTION) {
                    // Seçilen dosyanın yolunu belirle
                    Path outputPath = Path.of( first: fileChooser.getSelectedFile().toPath() + ".xlsx");
                    // Eğer dosya dizini yoksa oluştur
                    if (!Files.exists(outputPath.getParent())) {
                        Files.createDirectories(outputPath.getParent());
                    }
                    // Yeni bir Excel isbook'u oluştur
                    Workbook workbook = new XSSFWorbook();
                    Sheet sheet = workbook.createSheet(frameType);
                }
            }
        }
    });
}
```

```

// Başlık satırını oluşturun ve stil uygula
Row headerRow = sheet.createRow( rownum: 0);
CellStyle headerCellStyle = workbook.createCellStyle();
Font headerFont = workbook.createFont();
headerFont.setBold(true); // Kalın yazı için
headerCellStyle.setFont(headerFont);

// Kolon isimlerini başlık satırına ekle
ArrayList<String> columnList = getColumnList.main(fieldNames);
for (int i = 0; i < columnList.size(); i++) {
    Cell Cell = headerRow.createCell(i);
    Cell.setCellValue(columnList.get(i));
    Cell.setCellStyle(headerCellStyle);
}

// Tablodan verileri çek
ArrayList loadArrayList = returnListe.main(tabloSql);
int rowCount = 1;

// Verileri Excel sayfasına ekle
for (int i = 0; i < loadArrayList.size(); i = i + columnList.size()) {
    Row dataRow = sheet.createRow(rowCount++);
    for (int j = 0; j < columnList.size(); j++) {
        // Eğer bir ComboBox alındıysa, özel işlemleri uygula
        if (columnList.get(j).startsWith("jComboBox")) {
            dataRow.createCell(j).setCellValue(tableComboBox.main(columnList.get(j), loadArrayList.get(i + j)));
            continue;
        }
        dataRow.createCell(j).setCellValue(String.valueOf(loadArrayList.get(i + j)));
    }
}
}

```

```

// Dosyayı diske yaz
try (FileOutputStream fileOut = new FileOutputStream(outputPath.toFile())) {
    workbook.write(fileOut);
    showMessage.main("Excel dosyası oluşturuldu.");
    reload(fieldNames, tabloSql);
}
workbook.close();
} else {
    System.out.println("Dosya seçme işlemi iptal edildi.");
}
} catch (IOException ex) {
    throw new RuntimeException(ex);
}
}
});
return button;
}

```

Bu kod blogunda tablonun aynısını excelde bize yansıtacaktır... Örnek bir çıktı gösterim...

| A | B | C | D | E | F | G | H | I | |
|----|----|-------|----|-------------|-------------|----------------------|-----------|-------------------------|-------------------------|
| 1 | id | Title | Op | TC | PhoneNumber | Name | Surname | Username | Password |
| 2 | 26 | 4 | 0 | 89012321098 | 89012321098 | Aslıhan | Salın | aslıhansalin | 21312312 |
| 3 | 28 | 4 | 0 | 01232109876 | 01232109876 | Ayşe | Kik | Ayşekik | awe312 |
| 4 | 27 | 1 | 0 | 90123210987 | 90123210987 | Cihançır | Güven | cihanginguven | 123123123 |
| 5 | 23 | 7 | 0 | 56789012321 | 56789012321 | Derya | Kayma | deryakayma | deryakayma |
| 6 | 41 | 3 | 0 | 56789012345 | 56789012345 | Didem | Çavuşoğlu | didemcavusoglu | didemcavusoglu |
| 7 | 24 | 3 | 0 | 67890123210 | 67890123210 | Elif | Baş | elifbas | elifbas |
| 8 | 1 | 4 | 1 | 12345678901 | 12345678901 | Engin | Uçar | enginucar | enginucar |
| 9 | 22 | 3 | 0 | 45678901232 | 45678901232 | Ergün | Metin | ergunmetin | ergumetin |
| 10 | 25 | 4 | 0 | 78901232109 | 78901232109 | Gözde | Yılmaz | gozdeyilmaz | gozdeyilmaz |
| 11 | 21 | 4 | 0 | 34509876543 | 34509876543 | Hilal | Cura | hilalcura | hilalcura |
| 12 | 39 | 3 | 0 | 89012345678 | 89012345678 | İdris | Yağmur | idrisyagmur | idrisyagmur |
| 13 | 19 | 4 | 0 | 12345098765 | 12345098765 | İsmail Altuğ | Baysak | ismailaltugbaysak | ismailaltugbaysak |
| 14 | 42 | 3 | 0 | 78901234567 | 78901234567 | Kübra Göksu Köstepen | Özbek | kubragoksukostepenozbek | kubragoksukostepenozbek |
| 15 | 40 | 4 | 0 | 23456789012 | 23456789012 | Levent | Karakuz | leventkarakuz | leventkarakuz |
| 16 | 12 | 4 | 0 | 45678901234 | 45678901234 | Mesude | Uçar | mesudeucar | mesudeucar |
| 17 | 20 | 3 | 0 | 23450987654 | 23450987654 | Muhammed Mustafa | Vahşi | muhammedmustafabahsi | muhammedmustafabahsi |
| 18 | 3 | 4 | 0 | 34567890123 | 34567890123 | Murat | Albayrak | muratalbayrak | muratalbayrak |
| 19 | 17 | 4 | 0 | 90123456789 | 90123456789 | Mustafa | Çelik | mustafacelik | mustafacelik |
| 20 | 14 | 3 | 0 | 67890123456 | 67890123456 | Sezai | Bahar | sezaibahar | sezaibahar |
| 21 | 18 | 4 | 0 | 01234567890 | 01234567890 | Üğur | Bilgen | ugurbilgen | ugurbilgen |

Örnek çıktımız bu şekildedir.

Özel Çıktı Al:

Ders programı olarak çıktı almak istediğimiz zaman nasıl çıktı alabileceğimizi şimdi size anlatacağım... İlk öncelikle kendimize olası durumlara karşılık bir placeholders hazırlayacağız ve daha sonrasında verileri uygun Placeholderslere aktaracağız. Örnek taslağı aşağıda gösterevim.

Taslağın tamamını gösteremezsemde büyük bir kısmı aşağıda görüntülenmektedir.

| | {lesson1} | {lesson2} | {lessons} | {lesson4} | {lesson5} | {lesson6} |
|-----------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|--------------------------------|
| | {lesson1_hourse} | {lesson2_hourse} | {lesson3_hourse} | {lesson4_hourse} | {lesson5_hourse} | {lesson6_hourse} |
| Pazartesi | {lesson_1_1_lesson_Pazartesi} | {lesson_2_1_lesson_Pazartesi} | {lesson_3_1_lesson_Pazartesi} | {lesson_4_1_lesson_Pazartesi} | {lesson_5_1_lesson_Pazartesi} | {lesson_6_1_lesson_Pazartesi} |
| | {lesson_1_1_place_Pazartesi} | {lesson_2_1_place_Pazartesi} | {lesson_3_1_place_Pazartesi} | {lesson_4_1_place_Pazartesi} | {lesson_5_1_place_Pazartesi} | {lesson_6_1_place_Pazartesi} |
| | {lesson_1_1_teacher_Pazartesi} | {lesson_2_1_teacher_Pazartesi} | {lesson_3_1_teacher_Pazartesi} | {lesson_4_1_teacher_Pazartesi} | {lesson_5_1_teacher_Pazartesi} | {lesson_6_1_teacher_Pazartesi} |
| | {lesson_1_2_lesson_Pazartesi} | {lesson_2_2_lesson_Pazartesi} | {lesson_3_2_lesson_Pazartesi} | {lesson_4_2_lesson_Pazartesi} | {lesson_5_2_lesson_Pazartesi} | {lesson_6_2_lesson_Pazartesi} |
| | {lesson_1_2_place_Pazartesi} | {lesson_2_2_place_Pazartesi} | {lesson_3_2_place_Pazartesi} | {lesson_4_2_place_Pazartesi} | {lesson_5_2_place_Pazartesi} | {lesson_6_2_place_Pazartesi} |
| | {lesson_1_2_teacher_Pazartesi} | {lesson_2_2_teacher_Pazartesi} | {lesson_3_2_teacher_Pazartesi} | {lesson_4_2_teacher_Pazartesi} | {lesson_5_2_teacher_Pazartesi} | {lesson_6_2_teacher_Pazartesi} |
| SALI | {lesson_1_1_lesson_Sali} | {lesson_2_1_lesson_Sali} | {lesson_3_1_lesson_Sali} | {lesson_4_1_lesson_Sali} | {lesson_5_1_lesson_Sali} | {lesson_6_1_lesson_Sali} |
| | {lesson_1_1_place_Sali} | {lesson_2_1_place_Sali} | {lesson_3_1_place_Sali} | {lesson_4_1_place_Sali} | {lesson_5_1_place_Sali} | {lesson_6_1_place_Sali} |
| | {lesson_1_1_teacher_Sali} | {lesson_2_1_teacher_Sali} | {lesson_3_1_teacher_Sali} | {lesson_4_1_teacher_Sali} | {lesson_5_1_teacher_Sali} | {lesson_6_1_teacher_Sali} |
| | {lesson_1_2_lesson_Sali} | {lesson_2_2_lesson_Sali} | {lesson_3_2_lesson_Sali} | {lesson_4_2_lesson_Sali} | {lesson_5_2_lesson_Sali} | {lesson_6_2_lesson_Sali} |
| | {lesson_1_2_place_Sali} | {lesson_2_2_place_Sali} | {lesson_3_2_place_Sali} | {lesson_4_2_place_Sali} | {lesson_5_2_place_Sali} | {lesson_6_2_place_Sali} |
| | {lesson_1_2_teacher_Sali} | {lesson_2_2_teacher_Sali} | {lesson_3_2_teacher_Sali} | {lesson_4_2_teacher_Sali} | {lesson_5_2_teacher_Sali} | {lesson_6_2_teacher_Sali} |
| Çarşamba | {lesson_1_1_lesson_Carşamba} | {lesson_2_1_lesson_Carşamba} | {lesson_3_1_lesson_Carşamba} | {lesson_4_1_lesson_Carşamba} | {lesson_5_1_lesson_Carşamba} | {lesson_6_1_lesson_Carşamba} |
| | {lesson_1_1_place_Carşamba} | {lesson_2_1_place_Carşamba} | {lesson_3_1_place_Carşamba} | {lesson_4_1_place_Carşamba} | {lesson_5_1_place_Carşamba} | {lesson_6_1_place_Carşamba} |
| | {lesson_1_1_teacher_Carşamba} | {lesson_2_1_teacher_Carşamba} | {lesson_3_1_teacher_Carşamba} | {lesson_4_1_teacher_Carşamba} | {lesson_5_1_teacher_Carşamba} | {lesson_6_1_teacher_Carşamba} |
| | {lesson_1_2_lesson_Carşamba} | {lesson_2_2_lesson_Carşamba} | {lesson_3_2_lesson_Carşamba} | {lesson_4_2_lesson_Carşamba} | {lesson_5_2_lesson_Carşamba} | {lesson_6_2_lesson_Carşamba} |
| | {lesson_1_2_place_Carşamba} | {lesson_2_2_place_Carşamba} | {lesson_3_2_place_Carşamba} | {lesson_4_2_place_Carşamba} | {lesson_5_2_place_Carşamba} | {lesson_6_2_place_Carşamba} |
| | {lesson_1_2_teacher_Carşamba} | {lesson_2_2_teacher_Carşamba} | {lesson_3_2_teacher_Carşamba} | {lesson_4_2_teacher_Carşamba} | {lesson_5_2_teacher_Carşamba} | {lesson_6_2_teacher_Carşamba} |

Burada gördüğünüz gibi placeholderslerimi oluşturdum.. {lesson1} bu placeholdersim aynı şekilde pazartesi 1. Ders için {lesson_1_lesson_Pazartesi} gibi placeholderslerimiz mevcut. 1 Bölümde 2 şube açmaya izin verdiği için programımız bu şekilde hazırladım ama siz isterseniz bunu artırabilirsiniz....

Şimdi placeholdersleri nasıl kullanacağımızı anlatayım..

```
// Burada amacım FrameName (username) - Öğretmenin Ders Programı veya  
// Bilgisayar 1 - Bölümün ders programı olduğu için  
// Sadece Baş kısmını alıyorum...  
String[] Slot = frameName.split( regex: " - " );  
String columnData = Slot[0];  
String columnName = " ";  
  
// Şimdi burada Öğretmenin mi Bölümün mü bunu belirtiyorum ve Columnname'ni seçiyorum..  
if (frameType.equals("academic_program_episode")) {  
    String episodeSqlName = "SELECT id FROM Episode WHERE Name = ?";  
    columnData = jdbcTemplate.queryForObject(episodeSqlName, Integer.class, columnData) + " Ders Programı";  
    columnName = "Episode";  
  
} else {  
    String episodeSqlUsername = "SELECT id FROM Teacher WHERE Username = ?";  
    columnData = jdbcTemplate.queryForObject(episodeSqlUsername, Integer.class, columnData) + " Ders Programı";  
    columnName = "Teacher";  
}  
  
// Academic Nodemi çağırıyorum... ve tüm verileri ekliyorum...  
nodeAcademic nodeAcadem = new nodeAcademic();  
nodeAcadem.main(columnName, columnData);
```

İlk önce bir node oluşturmamız lazım.. ve bu node gerekli bilgileri işliyoruz.. Node içeriğimiz aşağıdaki gibidir...

```
public static void main(String columnName, String columData) {  
  
    // Ait olan stün sayısını alıyoruz...  
    String sql = "SELECT id FROM academic_program where " + columnName + " = ?";  
    ArrayList<Integer> academicId = (ArrayList<Integer>) jdbcTemplate.queryForList(sql, Integer.class, columData);  
  
    // Buradan Gerekli bilgileri node kayıt etmek için gerekli olan sql cümleleri  
    String academicEpisode = "SELECT e.name FROM academic_program a JOIN episode e ON a.episode = e.id WHERE a.id = ?";  
    String academicLesson = "SELECT e.name FROM academic_program a JOIN lesson e ON a.lessons = e.id WHERE a.id = ?";  
    String academicTeacher = "SELECT e.username FROM academic_program a JOIN teacher e ON a.teacher = e.id WHERE a.id = ?";  
    String academicClassroom = "SELECT e.type FROM academic_program a JOIN Classroom e ON a.Classroom = e.id WHERE a.id = ?";  
    String academicCode = "SELECT Lesson_Code from academic_program where id = ?";  
    String academicClassroomNumber = "SELECT Classroom_Number from academic_program where id = ?";  
    String academicDay = "SELECT Day from academic_program where id = ?";  
    String academicLessonTime = "SELECT LessonTime from academic_program where id = ?";  
  
    // Ait olan stün sayısı kadar döndür  
    for (Object id : academicId) {  
  
        // Gerekli bilgileri al..  
        Object episode = jdbcTemplate.queryForObject(academicEpisode, Object.class, id);  
        Object lesson = jdbcTemplate.queryForObject(academicLesson, Object.class, id);  
        Object lessonCode = jdbcTemplate.queryForObject(academicCode, Object.class, id);  
        Object teacher = jdbcTemplate.queryForObject(academicTeacher, Object.class, id);  
        Object classroom = jdbcTemplate.queryForObject(academicClassroom, Object.class, id);  
        Object classroomNumber = jdbcTemplate.queryForObject(academicClassroomNumber, Object.class, id);  
        Object day = jdbcTemplate.queryForObject(academicDay, Object.class, id);  
        Object lessonTime = jdbcTemplate.queryForObject(academicLessonTime, Object.class, id);  
  
        // Gerekli bilgileri kayıt et.  
        addNodeList(id, episode, lesson, lessonCode, teacher, classroom, classroomNumber, day, lessonTime);  
    }  
}
```

Bu şekilde node bilgiler çekiyoruz ve bunları kullanıyoruz...

```

try {
    // Dosya seçici oluşturuyor
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Save Excel File");
    FileNameExtensionFilter filter = new FileNameExtensionFilter( description: "Excel Files (*.xlsx)", ...extensions: "xlsx");
    fileChooser.setFileFilter(filter);

    // Kullanıcıdan kayıt konumunu seçmesini iste
    int userSelection = fileChooser.showSaveDialog( parent: null);

    if (userSelection == JFileChooser.APPROVE_OPTION) {
        // Seçilen dosyanın yolu belirleniyor
        java.io.File fileToSave = fileChooser.getSelectedFile();

        // Dosya uzantısı .xlsx değilse ekleniyor
        if (!fileToSave.getAbsoluteFilePath().toLowerCase().endsWith(".xlsx")) {
            fileToSave = new java.io.File( pathname: fileToSave.getAbsoluteFilePath() + ".xlsx");
        }

        // Excel dosyasını yükleyin
        FileInputStream fileInputStream = new FileInputStream( name: "src/main/resources/taslak.xlsx");
        Workbook workbook = new XSSFWorkbook(fileInputStream);

        // ilk sayfa (worksheet) seçin
        Sheet sheet = workbook.getSheetAt( index: 0);
    }
}

```

Gerekli excel olaylarını işliyoruz... şimdi sırada değiştirme işlemleri... Öncelikle replace işlemi yapmak için gerekli olan methodumuz.

```


/**
 * Excel sayfasındaki hücrelerde belirli bir metni arar ve bulduğum her hücrede bu metni başka bir metinle değiştirir.
 *
 * @param sheet      Değiştirilecek Excel sayfası
 * @param searchText  Değiştirilecek metni aramak için kullanılacak metin
 * @param replacementText  Değiştirilecek metni yerine kullanılacak yeni metin
 */
12 usages
private static void replaceTextInCells(Sheet sheet, String searchText, String replacementText) {
    // Her satır için döngü
    for (Row row : sheet) {
        // Her hücre için döngü
        for (Cell cell : row) {
            // Hücre tipi STRING ise devam et
            if (cell.getCellType() == CellType.STRING) {
                // Hücredeki metni al
                String cellValue = cell.getStringCellValue();
                // Eğer hücrede aranan metin varsa
                if (cellValue.contains(searchText)) {
                    // Hücredeki metni değiştir ve yeni metin ile güncelle
                    cell.setCellValue(cellValue.replace(searchText, replacementText));
                }
            }
        }
    }
}


```

Bu şekilde methodu çağrıp replace yapacağız...

İlk önce ders saatlerini ve kaçinci ders olduğunu replace yapalım...

| {lesson1} | {lesson2} | {lesson3} | {lesson4} | {lesson5} | {lesson6} |
|------------------|------------------|------------------|------------------|------------------|------------------|
| {lesson1_hourse} | {lesson2_hourse} | {lesson3_hourse} | {lesson4_hourse} | {lesson5_hourse} | {lesson6_hourse} |

Bunları replace çekip aşağıdaki gibi göstereceğiz..

| 1. Ders | 2. Ders | 3. Ders | 4. Ders | 5. Ders | 6. Ders |
|------------|--------------|---------------|---------------|---------------|---------------|
| 9:0 9:45 | 9:55 10:40 | 10:50 11:35 | 11:45 12:30 | 13:30 14:15 | 14:25 15:10 |

Bunun için gerekli kod satırı,

```

// Ders sayısını al
String lessonCountQuery = "SELECT COUNT(*) FROM times WHERE type = ?";
int lessonCount = jdbcTemplate.queryForObject(lessonCountQuery, Integer.class, ...args: "Ders");

String lessonStart = "SELECT Start FROM times WHERE type = 'Ders' AND Number = ?";
String lessonFinish = "SELECT Finish FROM times WHERE type = 'Ders' AND Number = ?";

for (int i = 1; i <= lessonCount; i++) {
    // {lesson1} olan placeholdersi 1. Ders olarak kayıt et
    replaceTextInCells(sheet, searchText: "{lesson" + i + "}", replacementText: i + ". Ders");

    int startLesson = jdbcTemplate.queryForObject(lessonStart, Integer.class, i);
    int finishLesson = jdbcTemplate.queryForObject(lessonFinish, Integer.class, i);

    // {lesson1_hourse} olan placeholdersi 9:0 | 9:45 kayıt et
    replaceTextInCells(sheet, searchText: "{lesson" + i + "_hourse}",
        replacementText: returnClock(startLesson) + " | " + returnClock(finishLesson));
}

}

```

Şimdi gelelim ders içerisinde güncelleme kodlarımıza bunlarda aşağıdaki gibi güncelleyeceğiz..

```

while (active != null) { // Node bitene kadar donecek dongümüz
    // Eğer lessonCode'de "/" işaretini varsa bu bölüme ayrılmış demektir.
    if (((String) active.lessonCode).contains("/")) {
        // ve bunu splint ile ayıryoruz mesela BL-50/2 ise bu 2. sube kayıt edilecek demektir...
        String[] parcalar = ((String) active.lessonCode).split(regex: "/");
        // Gerekli placeholdersleri güncelliyoruz...
        replaceTextInCells(sheet, searchText: "{lesson_" + active.lessonTime + "." + parcalar[1] + "_lesson_" + active.day + "}",
            replacementText: active.lesson + " " + active.lessonCode);
        replaceTextInCells(sheet, searchText: "{lesson_" + active.lessonTime + "." + parcalar[1] + "_place_" + active.day + "}",
            replacementText: active.classroom + " " + active.classroomNumber);
        replaceTextInCells(sheet, searchText: "{lesson_" + active.lessonTime + "." + parcalar[1] + "_teacher_" + active.day + "}"
            replacementText: active.teacher + " ");
    } else {
        // Eğer yoksa bu ortak derstir ve ortak dersleri tam olarak yazıyoruz 1 ve 2. sube aynı düzenle...
        replaceTextInCells(sheet, searchText: "{lesson_" + active.lessonTime + ".1_lesson_" + active.day + "}",
            replacementText: active.lesson + " " + active.lessonCode);
        replaceTextInCells(sheet, searchText: "{lesson_" + active.lessonTime + ".2_lesson_" + active.day + "}",
            replacementText: active.lesson + " " + active.lessonCode);
        replaceTextInCells(sheet, searchText: "{lesson_" + active.lessonTime + ".1_place_" + active.day + "}",
            replacementText: active.classroom + " " + active.classroomNumber);
        replaceTextInCells(sheet, searchText: "{lesson_" + active.lessonTime + ".2_place_" + active.day + "}",
            replacementText: active.classroom + " " + active.classroomNumber);
        replaceTextInCells(sheet, searchText: "{lesson_" + active.lessonTime + ".1_teacher_" + active.day + "}",
            replacementText: active.teacher + " ");
        replaceTextInCells(sheet, searchText: "{lesson_" + active.lessonTime + ".2_teacher_" + active.day + "}",
            replacementText: active.teacher + " ");
    }
    // Aktiviyi bir öncekine al... (Sondan başa gidiyoruz)
    active = active.last;
}

```

Artık son olarak tablo ismini belirteceğiz bu tablonun ne olduğunu bilmesi için..

```

// Tablo ismini Slottaki 0. Öğe ile değiştir.
replaceTextInCells(sheet, searchText: "{tablo_name}", Slot[0]);

```

Onuda böyle ufak bir kodla replace cekiyoruz... Son olarak doldurulmamış stünları kaldırmak için gerekli kodumuz

```
// Koseli paranter olan tüm stünleri sil..
deleteCellsStartingWith(sheet, prefix: "{}");
```

Çağrıdığımız methoddaki kod bloğunun içeriği...

```
/*
 * Excel sayfasındaki belirli bir önekle başlayan hücreleri siler.
 *
 * @param sheet Silme işlemi yapılacak Excel sayfası
 * @param prefix Silinecek hücrelerin öneki
 */

1 usage
private static void deleteCellsStartingWith(Sheet sheet, String prefix) {
    // Her satır için döngü
    for (Row row : sheet) {
        // Her hücre için döngü
        for (Cell cell : row) {
            // Hücre tipi STRING ise devam et
            if (cell.getCellType() == CellType.STRING) {
                // Hücredeki metni al
                String cellValue = cell.getStringCellValue();
                // Eğer hücredeki metin belirtilen önekle başlıyorsa
                if (cellValue.startsWith(prefix)) {
                    // Hücre içeriğini sil
                    cell.setCellValue("");
                }
            }
        }
    }
}
```

Algoritma Aşamaları Kodlarla

Şimdi şuna kadar yazılan kodlardaki herşeyi kapataslak anlatmış bulunmaktayız. Şuna kadar kurut işlemlerinin hepsinin nasıl yapılacağını ve yapıldığını gösterdim. Şimdi ders programı otomasyonunu çalıştırmayı anlatacağım.

Algoritmada yaşadığım en büyük sıkıntı Local makinemde çok fazla sorgu yaptığım için xamp'in firewallını çok fazla tetikledim. Bu yüzden algoritmada çok fazla sorgu yapmamak için gerekli bütün bilgileri bir Node'ye kayıt edeceğim.

1. Bölümlerin Müsaitliği

Algoritmada bölümlerin o gün o ders başka ders almaması için müsait olup olmadığını kontrol edeceğiz algoritma ilk olarak tüm gün müsait olarak düşünecek ve nodelere o şekilde kayıt edecek daha sonrasında ders atadıkça bu kaydı silecek.

```
16 usages
public class availablesEpisode {

    // Node sınıfı için değişkenler
    public Object Episode;
    public ArrayList Days;
    3 usages
    public availablesEpisode next;
    8 usages
    public availablesEpisode last;

    // Yarıcıl metod
    2 usages
    public availablesEpisode(Object Episode, ArrayList Days) {
        this.Episode = Episode;
        this.Days = Days;
        this.next = null;
        this.last = null;
    }

    // Boş bir bitiş node'u oluşturuluyor
    10 usages
    public static availablesEpisode FinishEpisodeNode = new availablesEpisode( Episode: " ", new ArrayList());
    1 usage
    public availablesEpisode() {
        FinishEpisodeNode.last = null;
        FinishEpisodeNode.next = null;
    }

    // Yeni bir node ekleyen metod
    1 usage
    public static void addNodeList(Object Episode, ArrayList Days) {
        availablesEpisode newNode = new availablesEpisode(Episode, Days);
        newNode.last = FinishEpisodeNode.last;
        FinishEpisodeNode.last = newNode;
        newNode.next = FinishEpisodeNode;
    }
}
```

```
// Ana metod - Bölümü tamamen müsait olarak oluşturacak
public static void main() {

    // Tüm Bölümlerin ID'lerini al
    String sql = "SELECT id FROM Episode";
    ArrayList<Integer> episodeIds = (ArrayList<Integer>) jdbcTemplate.queryForList(sql, Integer.class);

    // Ders sayısını al 8 ders gibi
    String lessonCountQuery = "SELECT COUNT(*) FROM times WHERE type = ?";
    int lessonCount = jdbcTemplate.queryForObject(lessonCountQuery, Integer.class, ...args: "Ders");

    // Günleri al
    String daySql = "SELECT Day FROM days";
    ArrayList schoolDay = (ArrayList) jdbcTemplate.queryForList(daySql, String.class);

    // Döngü ile Pazartesi 1 gibi kayıt et.
    for (Object id : episodeIds) {
        ArrayList sendActive = new ArrayList();
        for (Object Day : schoolDay) {
            for (int i = 1; i <= lessonCount; i++) {
                sendActive.add(Day + " " + i);
            }
        }
        addNodeList(id, sendActive);
    }
}
```

2. Dersliklerin Müsaitliği

Burada aynı dersliklere birden fazla bölüm ve ders eklememek için dersliklere de müsaitlik durumu belirtip ekleyeceğiz..

```
15 usages
public class availablesRoom {
    // Node sınıfı için değişkenler
    4 usages
    public Object Room;
    4 usages
    public int Custom;
    1 usage
    public int Capacity;
    public ArrayList Days;
    3 usages
    public availablesRoom next;
    7 usages
    public availablesRoom last;

    // Yapıcı metod
    2 usages
    public availablesRoom(Object Room, int Custom, int Capacity, ArrayList Days) {
        this.Room = Room;
        this.Custom = Custom;
        this.Days = Days;
        this.Capacity = Capacity;
        this.next = null;
        this.last = null;
    }
    // Boş bir bitiş node'u oluşturuyor
    9 usages
    public static availablesRoom FinishRoomNode = new availablesRoom( Room: " ", Custom: ' ', Capacity: ' ', new ArrayList());
    // Boş bir bitiş node'u oluşturuyor
    1 usage
    public availablesRoom() {
        FinishRoomNode.last = null;
        FinishRoomNode.next = null;
    }
}
```

```

// Ana metod - Sınıfların kullanılabilirlik durumlarını oluşturur
public static void main() {

    // Tüm sınıfların ID'lerini al
    String sql = "SELECT id FROM classroom";
    ArrayList<Integer> classroomIds = (ArrayList<Integer>) jdbcTemplate.queryForList(sql, Integer.class);

    // Ders programındaki ders sayısını al
    String lessonCountQuery = "SELECT COUNT(*) FROM times WHERE type = ?";
    int lessonCount = jdbcTemplate.queryForObject(lessonCountQuery, Integer.class, ...args: "Ders");

    // Tüm günleri al
    String daySql = "SELECT Day FROM days";
    ArrayList schoolDay = (ArrayList) jdbcTemplate.queryForList(daySql, String.class);

    for (Object id : classroomIds) {
        // Derslik, Labotuar bunların adetini al kaç adet var
        String customSql = "SELECT Custom FROM classroom Where id = ?";
        int custom = jdbcTemplate.queryForObject(customSql, Integer.class, id);
        // Her bir sınıf için günleri ve saatleri oluşturup linked list'e ekle
        String classroomCapacity = "SELECT Capacity FROM classroom WHERE id = ?";
        int Capacity = jdbcTemplate.queryForObject(classroomCapacity, Integer.class, id);

        // Müsaitlikleri ekleme...
        for (int j = 1; j <= custom; j++) {
            ArrayList sendActive = new ArrayList();
            for (Object Day : schoolDay) {
                for (int i = 1; i <= lessonCount; i++) {
                    sendActive.add(Day + " " + i);
                }
            }
            addNodeList(id, j, Capacity, sendActive);
        }
    }
}

```

3. Öğretmen Müsaitliği

Öğretmenlere aynı gün aynı ders birden fazla ders atamamak için onlara müsaitlik durumu ekleyeceğiz ama availabilityteacher tablosuna dikkat ederek ekleyeceğiz.

```

15 usages
public class availablesTeacher {
    // Node sınıfı için değişkenler
    public Object Teacher;
    public ArrayList Days;
    3 usages
    public availablesTeacher next;
    7 usages
    public availablesTeacher last;

    // Yarıcı metod
    2 usages
    public availablesTeacher (Object Teacher, ArrayList Days){
        this.Teacher = Teacher;
        this.Days = Days;
        this.next = null;
        this.last = null;
    }

    // Boş bir bitiş node'u oluşturuluyor
    9 usages
    public static availablesTeacher FinishTeacherNode = new availablesTeacher( Teacher, new ArrayList());

    // Boş bir bitiş node'u oluşturuluyor
    1 usage
    public availablesTeacher() {
        FinishTeacherNode.last = null;
        FinishTeacherNode.next = null;
    }

    // Yeni bir node ekleyen metod
    1 usage
    public static void addNodeList(Object Teacher, ArrayList Days){
        availablesTeacher newNode = new availablesTeacher(Teacher,Days);
        newNode.last = FinishTeacherNode.last;
        FinishTeacherNode.last = newNode;
        newNode.next = FinishTeacherNode;
    }
}

```

```

// Ana metod - Öğretmenlerin müsaitlik durumlarını oluşturur
public static void main(){

    // Tüm öğretmenlerin ID'lerini al
    String sql = "SELECT id FROM Teacher";
    ArrayList<Integer> teacherIds = (ArrayList<Integer>) jdbcTemplate.queryForList(sql, Integer.class);

    // Ders sayısını al
    String lessonCountQuery = "SELECT COUNT(*) FROM times WHERE type = ?";
    int lessonCount = jdbcTemplate.queryForObject(lessonCountQuery, Integer.class, ...args: "Ders");

    for (Object id : teacherIds){
        // Tüm günleri al
        String daySql = "SELECT Day FROM days";
        ArrayList schoolDay = (ArrayList) jdbcTemplate.queryForList(daySql, String.class);

        // Öğretmenin müsait olmadığı günleri al
        String notDaySql = "SELECT Day FROM availabilityteacher Where Teacher = ?";
        ArrayList notTeacherDay = (ArrayList) jdbcTemplate.queryForList(notDaySql, String.class, id);

        // Tüm günlerden çıkar.
        schoolDay.removeAll(notTeacherDay);

        ArrayList sendActive = new ArrayList();
        // Müsaitlikleri al
        for (Object Day : schoolDay){
            for (int i = 1; i <= lessonCount; i++) {
                sendActive.add(Day + " " + i);
            }
        }
        // Kaydet
        addNodeList(id,sendActive);
    }
}

```

4. Ders Bilgileri

Şimdi ders bilgilerini kayıt etmek için bir node oluşturacağız..

```
public class nodeLesson {
    // Node sınıfı için değişkenler
    public Object Lesson;  public Object lessonId;
    public Object Type;   public Object Classroom;
    1 usage
    public Object Capacity; public Object Custom;
    4 usages
    public Object Code;   public Object Time;
    1 usage
    public Object Student; public Object Branch;
    public ArrayList Episode;  public ArrayList Teacher;
    3 usages
    public nodeLesson next; public nodeLesson last;

    // Yarıcı metod
    2 usages
    public nodeLesson(Object Lesson, Object lessonId, Object Classroom, Object Type, Object Capacity,
                      Object Custom, Object Code, Object Time, Object Student, Object Branch, ArrayList Episode, ArrayList Teacher) {
        this.Lesson = Lesson;
        this.lessonId = lessonId;
        this.Classroom = Classroom;
        this.Type = Type;
        this.Capacity = Capacity;
        this.Custom = Custom;
        this.Code = Code;
        this.Time = Time;
        this.Student = Student;
        this.Branch = Branch;
        this.Episode = Episode;
        this.Teacher = Teacher;
        this.next = null;
        this.last = null;
    }
}
```

```
// Boş bir bitiş node'u olşturuluyor
7 usages
public static nodeLesson FinishLessonNode = new nodeLesson( Lesson: " ",  lessonId: " ",  Classroom: " ",
    Type: " ",  Capacity: " ",  Custom: " ",  Code: " ",  Time: " ",  Student: " ",  Branch: " ",
    new ArrayList(), new ArrayList());

// Boş bir bitiş node'u olşturuluyor
2 usages
public nodeLesson() {
    FinishLessonNode.last = null;
    FinishLessonNode.next = null;
}

// Yeni bir node ekleyen metod
2 usages
public static void addNodeList(Object Lesson, Object lessonId, Object Classroom, Object Name,
                               Object Capacity, Object Custom, Object Code, Object Time, Object Student,
                               Object Branch, ArrayList Episode, ArrayList Teacher) {
    nodeLesson newNode = new nodeLesson(Lesson, lessonId, Classroom, Name, Capacity, Custom, Code, Time, Student, Branch, Episode, Teacher);

    newNode.last = FinishLessonNode.last;
    FinishLessonNode.last = newNode;
    newNode.next = FinishLessonNode;
}
```

```

// Ana metod - Ders bilgilerini çekip linked list'e ekler
public static void main() {

    // Tüm derslerin ID'lerini al
    String sql = "SELECT id FROM lesson";
    ArrayList<Integer> lessons = (ArrayList<Integer>) jdbcTemplate.queryForList(sql, Integer.class);

    // Dersle ilgili bilgileri çekmesi için SQL
    String lessonClassroom = "SELECT Classroom FROM lesson WHERE id = ?";
    String lessonCode = "SELECT Code FROM lesson WHERE id = ?";
    String lessonName = "SELECT Name FROM lesson WHERE id = ?";
    String lessonTime = "SELECT Time FROM lesson WHERE id = ?";
    String lessonStudent = "SELECT Student FROM lesson WHERE id = ?";
    String lessonBranch = "SELECT Branch FROM lesson WHERE id = ?";
    String lessonTeacher = "SELECT Episode FROM episodecourses WHERE lesson = ?";
    String lessonEpisode = "SELECT Teacher FROM teachercourses WHERE lesson = ?";
    String classroomCapacity = "SELECT Capacity FROM classroom WHERE id = ?";
    String classroomCustom = "SELECT Custom FROM classroom WHERE id = ?";
    String classroomName = "SELECT Type FROM classroom WHERE id = ?";

    // Default
    String Classroom = " ";
    String Code = " ";
    String Lesson = " ";
    String Time = " ";
    String Student = " ";
    String Branch = " ";
    String Capacity = " ";
    String Custom = " ";
    String Type = " ";

    ArrayList<Integer> episode = new ArrayList<>();
    ArrayList<Integer> teacher = new ArrayList<>();

    Object ide = 1;
}

```

```

// Ders bilgilerini çekip linked list'e ekle
for (Object id : lessons) {

    // Dersle ilgili bilgileri çek
    Classroom = jdbcTemplate.queryForObject(lessonClassroom, String.class, id);
    Code = jdbcTemplate.queryForObject(lessonCode, String.class, id);
    Lesson = jdbcTemplate.queryForObject(lessonName, String.class, id);
    Time = jdbcTemplate.queryForObject(lessonTime, String.class, id);
    Student = jdbcTemplate.queryForObject(lessonStudent, String.class, id);
    Branch = jdbcTemplate.queryForObject(lessonBranch, String.class, id);

    Capacity = jdbcTemplate.queryForObject(classroomCapacity, String.class, Classroom);
    Custom = jdbcTemplate.queryForObject(classroomCustom, String.class, Classroom);
    Type = jdbcTemplate.queryForObject(classroomName, String.class, Classroom);

    episode = (ArrayList<Integer>) jdbcTemplate.queryForList(lessonTeacher, Integer.class, id);
    teacher = (ArrayList<Integer>) jdbcTemplate.queryForList(lessonEpisode, Integer.class, id);

    ide = id;
    addNodeList(Lesson,id, Classroom, Type, Capacity, Custom, Code, Time, Student, Branch, episode, teacher);
}

addNodeList(Lesson,ide, Classroom, Type, Capacity, Custom, Code, Time, Student, Branch, episode, teacher);
}

```

Programı Başlatmak

Şimdi programı başlatmak için ilk önce nodeleri çalıştıracağımız çalıştmak için

```
public class program {
    public static void main(){
        // Academic programı sıfırlıyoruz
        String sql = "delete from academic_program";
        jdbcTemplate.execute(sql);
        // Öğretmen müsaitlikleri güncelle
        availablesTeacher teacher = new availablesTeacher();
        teacher.main();
        // Classroom'ları müsaitliği ekle
        availablesRoom room = new availablesRoom();
        room.main();
        // Bölümü'lerin müsaitliğini ekle
        availablesEpisode episode = new availablesEpisode();
        episode.main();
        // Lesson bilgilerini al
        nodeLesson lesson = new nodeLesson();
        lesson.main();
        // Başlaaa...
        starlend();
    }
}
```

Nodelerimizi tamamen oluşturuktan sonra starlend menüsüne geçiyoruz...

```

2 usages
public static void starlend() {

    // Lesson bilgileri alıyoruz
    nodeLesson lesson = new nodeLesson();
    lesson.main();

    // En son node alıyoruz
    nodeLesson active = FinishLessonNode.last;

    // Node bitene kadar çalıştırıyoruz
    while (active != null) {

        // Öğretmen ve müsaitlik için bir ArrayList oluşturuyoruz
        ArrayList<String> teacherDay = new ArrayList<>();
        ArrayList<String> episodeDay = new ArrayList<>();
        try {
            // Buradan bilgileri çekiyoruz.....
            teacherDay = getNodeTeacher(active.Teacher.get(0));
            episodeDay = getNodeEpisode(active.Episode.get(0));
        } catch (Exception e) {
            throw new RuntimeException(e);
        } finally {
            // Eğer hata varsa... hata yazdırıyoruz...
            if (teacherDay.size() == 0) {
                showMessage.main("Lütfen " + active.Lesson + " Dersine öğretmen atayın.");
            } else if (episodeDay.size() == 0) {
                showMessage.main("Lütfen " + active.Lesson + " Dersine bölüm atayın.");
            }
        }
    }
}

```

```

// Öğretmenin müsait olduğu kontrol edilen bir fonksiyon çağrılır.
// Eğer öğretmen müsait değilse, başka bir müsait öğretmen bulma işlemleri gerçekleştiriliyor.
String Timec = (String) active.Time; // Dersin saat bilgisini bir stringe çevirir.
int Time = Integer.parseInt(Timec); // String olarak alınan saat bilgisini integer'a dönüştürür.
// Bootstrap hatasından dolayı ilk string sonra int

// controlTeacher fonksiyonu ile öğretmenin müsait olup olmadığı kontrol edilir.
// Eğer öğretmen müsait değilse, başka bir müsait öğretmen bulunması için işlemler yapılır.
if (!controlTeacher(teacherDay, episodeDay, Time)) {
    // Tüm öğretmen ID'lerini içeren bir liste alınır.
    String sql = "SELECT id FROM Teacher";
    ArrayList<Integer> teacherIds = (ArrayList<Integer>) jdbcTemplate.queryForList(sql, Integer.class);
    // Mevcut öğretmen (active.Teacher) listeden çıkarılır.
    teacherIds.remove(active.Teacher);
    // Müsait öğretmeni bulmak için teacherAlert fonksiyonu çağrılır.
    // Bu fonksiyon, müsait öğretmeni bulana kadar işlemi tekrarlar.
    active.Teacher.set(0, teacherAlert(teacherIds, active.Lesson));
}

```

Şimdi müsait günleri alacağız...

```

// Eğer Branch 1 ise bölünmeden...
if (active.Branch.equals("1")) {

    // Müsaitlikleri al ArrayList sonunda classroom var.
    ArrayList availablesDay =
        getNodeAvailables(getNodeTeacher(active.Teacher.get(0)), // Aktif öğretmenin düğümünü alır.
        getNodeEpisode(active.Episode.get(0)), // Aktif bölümün düğümünü alır.
        Integer.parseInt(active.Time.toString()), // Aktif dersin saatı
        active.Classroom, // Aktif dersin sınıf bilgisi
        active.Lesson); // Aktif dersin adı

    String liste1 = (String) availablesDay.get(0);
    String[] parts = liste1.split( regex: " " );

    while (parts.length != 2) { // ufak bir kontrol... eğer atama olmadıysa olaşa kadar tekrar yapır.
        availablesDay = getNodeAvailables(getNodeTeacher(active.Teacher.get(0)),
            getNodeEpisode(active.Episode.get(0)), Integer.parseInt(active.Time.toString()),
            active.Classroom, active.Lesson);
    }

    // Teacher ve episode müsait günü alıyoruz ve günleri siliyoruz.
    setNodeTeacher(active.Teacher.get(0), availablesDay);
    setNodeEpisode(active.Episode.get(0), availablesDay);
}

```

getTeacherAvailables ile müsait günleri ve classroomları alıyoruz. Methodun içini acarsak eğer...

```

2 usages
public class branch1Availables {

    /**
     * Öğretmenin uygun gün ve saatlerini kontrol eder.
     *
     * @param teacherDay Öğretmenin uygun gün ve saatlerini içeren liste
     * @param episodeDay Bölüm planlı gün ve saatlerini içeren liste
     * @param lessonTime Dersin süresi (kaç saat olduğu)
     * @param Classroom Dersin gerçekleştirileceği sınıf bilgisi
     * @param lessonName Dersin adı
     * @return Dersin gerçekleştirilebileceği uygun gün ve saatleri içeren liste
     */
    3 usages
    public static ArrayList<String> getTeacherAvailables(ArrayList<String> teacherDay,
                                                          ArrayList<String> episodeDay, int lessonTime,
                                                          Object Classroom, Object lessonName)
    {
        // Öğretmenin uygun olduğu gün ve saatleri kontrol etmek için döngü
        for (String day : teacherDay) {
            String[] slot = day.split( regex: " ", limit: 2);
            String selectDay = slot[0];
            int selectTime = Integer.parseInt(slot[1]);

            // Bölüm planlı olduğu gün ve saat aralığını kontrol etmek için döngü
            ArrayList<String> control = new ArrayList<>();
            for (int i = 0; i < lessonTime; i++) {
                control.add(selectDay + " " + (selectTime + i));
            }

            // HashSet'ler kullanılarak öğretmenin ve dersin gün ve saatleri kontrol edilir
            HashSet<String> teacherDaySet = new HashSet<>(teacherDay);
            HashSet<String> episodeDaySet = new HashSet<>(episodeDay);
        }
    }
}

```

```

// HashSet'ler kullanılarak öğretmenin ve dersin gün ve saatleri kontrol edilir
HashSet<String> teacherDaySet = new HashSet<>(teacherDay);
HashSet<String> episodeDaySet = new HashSet<>(episodeDay);

int cont = 0;
// Bölüm ve öğretmenin uygun olduğu gün ve saatleri saymak için döngü
for (String otherController : control) {
    if (teacherDaySet.contains(otherController) && episodeDaySet.contains(otherController)) {
        cont++;
    }
}
// Eğer bölüm ve öğretmenin uygun olduğu gün ve saatler tam olarak eşlesiyorsa
if (cont == lessonTime) {
    // Sınıfın uygun olduğu gün ve saatleri kontrol eder
    ArrayList returnList = getRoomAvailables(Classroom, control, lessonTime);

    // Eğer sınıf uygun değilse uygun bir sınıf belirler
    String sql = "SELECT id FROM classroom";
    ArrayList<Integer> classroomIds = (ArrayList<Integer>) jdbcTemplate.queryForList(sql, Integer.class);

    // Uygun sınıf belirlenecek kadar döngü yapar
    while (returnList == null) {
        classroomIds.remove(Classroom);
        Classroom = classroomAlert(classroomIds, lessonName);
        returnList = getRoomAvailables(Classroom, control, lessonTime);
    }

    // Eğer uygun sınıf bulunduysa, uygun gün ve saatleri döndürür
    if (returnList != null) {
        return returnList;
    }
}
}

// Uygun gün ve saat bulunamazsa null döndürür
return null;

```

Bu şekilde methodumuz şimdi ise. ...

```

// Teacher ve episode müsait günü alıyoruz ve günleri siliyoruz.
setNodeTeacher(active.Teacher.get(0), availablesDay);
setNodeEpisode(active.Episode.get(0), availablesDay);

```

setNodeEpisode ve setNodeTeacher ile o gün müsaitliğini siliyoruz...

```
/**  
 * Belirtilen öğretmenin müsait günlerinden belirtilen günleri çıkarır.  
 *  
 * @param Teacher      Günleri çıkarılacak öğretmen  
 * @param removeDays   Çıkarılacak günlerin listesi  
 */  
3 usages  
public static void setNodeTeacher(Object Teacher, ArrayList removeDays) {  
    // Listenin başından başlayarak öğretmeni bulana kadar döngü yapar  
    availablesTeacher active = FinishTeacherNode;  
    while (!active.Teacher.equals(Teacher)) {  
        active = active.last;  
    }  
    // Belirtilen günleri öğretmenin müsait günlerinden çıkarır  
    active.Days.removeAll(removeDays);  
}
```

```
/**  
 * Belirtilen bölümün müsait günlerinden belirtilen günleri çıkarır.  
 *  
 * @param Episode      Günleri çıkarılacak bölüm  
 * @param removeDays   Çıkarılacak günlerin listesi  
 */  
1 usage  
public static void setNodeEpisode(Object Episode, ArrayList removeDays) {  
    // Listenin başından başlayarak bölümü bulana kadar döngü yapar  
    availablesEpisode active = FinishEpisodeNode;  
    while (!active.Episode.equals(Episode)) {  
        active = active.last;  
    }  
    // Belirtilen günleri bölümün müsait günlerinden çıkarır  
    active.Days.removeAll(removeDays);  
}
```

Şimdi algoritmamıza devam edelim..

```
// Classroom bilgileri alıyoruz malum en sonda Arrylistin...
liste1 = (String) availablesDay.get(availablesDay.size() - 1);
parts = liste1.split( regex: " " );
String classroom = parts[0];
String stringCustom = parts[1];
int Custom = Integer.parseInt(stringCustom);
int Classroom = Integer.parseInt(classroom);
setNodeClassroom(Classroom, Custom, availablesDay);
```

Arrylistin sonunda classroom olduğu için arrylistin sonunu alıyoruz.... Ve parçalara ayırip Derslik 2 diye alıyoruz...

```
// Kayıt işlemini Nodeden ve kontrollerden aldığımız bilgileri alıyoruz..
for (int i = 0; i < Integer.parseInt(active.Time.toString()); i++) {
    String day = (String) availablesDay.get(i);
    String[] part = day.split( regex: " " );
    String Day = part[0];
    String lessonTime = part[1];
    sendAcademicProgram(active.Episode.get(0), active.lessonId, active.Code,
        active.Teacher.get(0), Classroom, Custom, Day, lessonTime);
}
```

Kayıt işlemi bitmiştir.... Fakat eğer şubelere ayrılacaksa durum daha farklıdır şimdi bu durumda ne yapılacağına bakalım.

Eğer ders bölünecekse ve o derse tek 1 hoca aktarılıysa...

```

if (active.Teacher.size() == 1) {
    // 2 döngü kullanarak müsait günleri işle
    for (int j = 1; j <= 2; j++) {
        // Öğretmenin, bölümün ve gün sayısının dikkate alındığı müsait günleri al
        ArrayList availablesDay = getNodeAvailables2(getNodeTeacher(active.Teacher.get(0)),
            getNodeEpisode(active.Episode.get(0)), Integer.parseInt(active.Time.toString()),
            active.Classroom, active.Lesson, j);

        // Müsait günü öğretmenden siliyoruz.
        setNodeTeacher(active.Teacher.get(0), availablesDay);

        // Müsait günü bölümden siliyoruz.
        setNodeEpisode2(active.Episode.get(0), availablesDay, String.valueOf(j));

        // Müsait günü Derslikten siliyoruz.
        String liste1 = (String) availablesDay.get(availablesDay.size() - 1);
        String[] parts = liste1.split( regex: " " );
        String classroom = parts[0];
        String stringCustom = parts[1];
        int Custom = Integer.parseInt(stringCustom);
        int Classroom = Integer.parseInt(classroom);

        // Gün formatını düzenle
        for (int i = 0; i < availablesDay.size() - 1; i++) {
            String liste2 = (String) availablesDay.get(i);
            String[] part = liste2.split( regex: " " );
            String day = part[0];
            String custom = part[1];
            availablesDay.set(i, day + " " + custom);
        }
    }

    // Kayıt et
    for (int i = 0; i < Integer.parseInt(active.Time.toString()); i++) {
        String day = (String) availablesDay.get(i);
        String[] part = day.split( regex: " " );
        String Day = part[0];
        String lessonTime = part[1];
        sendAcademicProgram(active.Episode.get(0), active.lessonId, Lesson_Code: active.Code + "/" + i,
            active.Teacher.get(0), Classroom, Custom, Day, lessonTime);
    }
}

```

2 Öğretmen yapılmacı zaman yaptığımız tek şey teacher.get ile arrylist sabit 0 olarak değil j değişkenine göre alacağız.. Sadece teacher get kısımlarını dikkatli bakın...

```
    } else {
        for (int j = 1; j <= 2; j++) {
            |
            // Müsait günü ve Classroom alıyoruz. Değişen şey teacher.get 0 değil j-1 alacağımız
            ArrayList availablesDay = getNodeAvailables2(getNodeTeacher(active.Teacher.get(j - 1)),
                getNodeEpisode(active.Episode.get(0)), Integer.parseInt(active.Time.toString()),
                active.Classroom, active.Lesson, j);

            // Müsait günü öğrencimden siliyoruz.
            setNodeTeacher(active.Teacher.get(j-1), availablesDay);

            // Müsait günü bölümünden siliyoruz.
            setNodeEpisode2(active.Episode.get(0), availablesDay, String.valueOf(j));

            // Müsait günü Derslikten siliyoruz.
            String liste1 = (String) availablesDay.get(availablesDay.size() - 1);
            String[] parts = liste1.split( regex: " " );
            String classroom = parts[0];
            String stringCustom = parts[1];
            int Custom = Integer.parseInt(stringCustom);
            int Classroom = Integer.parseInt(classroom);

            // Gün formatını düzenle
            for (int i = 0; i < availablesDay.size() - 1; i++) {
                String liste2 = (String) availablesDay.get(i);
                String[] part = liste2.split( regex: " " );
                String day = part[0];
                String custom = part[1];
                availablesDay.set(i, day + " " + custom);
            }
        }

        // Döndürmek üzere günleri belirtilen günlerden sil
        setNodeClassroom(Classroom, Custom, availablesDay);

        for (int i = 0; i < Integer.parseInt(active.Time.toString()); i++) {
            String day = (String) availablesDay.get(i);
            String[] part = day.split( regex: " " );
            String Day = part[0];
            String lessonTime = part[1];
            sendAcademicProgram(active.Episode.get(0), active.lessonId, Lesson_Code: active.Code + " "
                + i, active.Teacher.get(j - 1), Classroom, Custom, Day, lessonTime);
        }
    }
}
```

Alogirtma Akış Diyagramı...

1. Bölümlerin okul açık olduğu tüm gün müsait olarak kayıt et.
 2. Derslikleri okul açık olduğu tüm gün müsait olarak kayıt et.
 3. Öğretmenlerin okul açık olduğu tüm gün müsait olarak kayıt et.
 4. Öğretmenlerde müsait olmadığı günleri sil.
 5. Derslerin gereklili bilgilerini al.
 6. Aldığın bilgileri topla ve sırayla oku.
 7. Uygun öğretmen ve bölümü uygun saatlerini bul.
 8. Uygun Derslik bul.
 9. Uygun zamanı bölüm, öğretmen, derslik kayıt et.
 10. Bölüm ve öğretmen, derslik müsaitliğini sil o gün için

