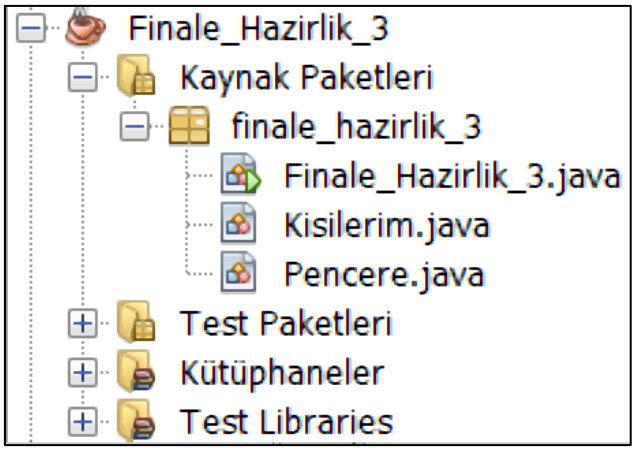
GÖRSEL PROGRAMLAMA

FİNALE HAZIRLIK ÖRNEKLER

1. KULLANICI GRAFİK ARAYÜZ ÖRNEKLERİ

1.1. Kayıt Defteri Örneği

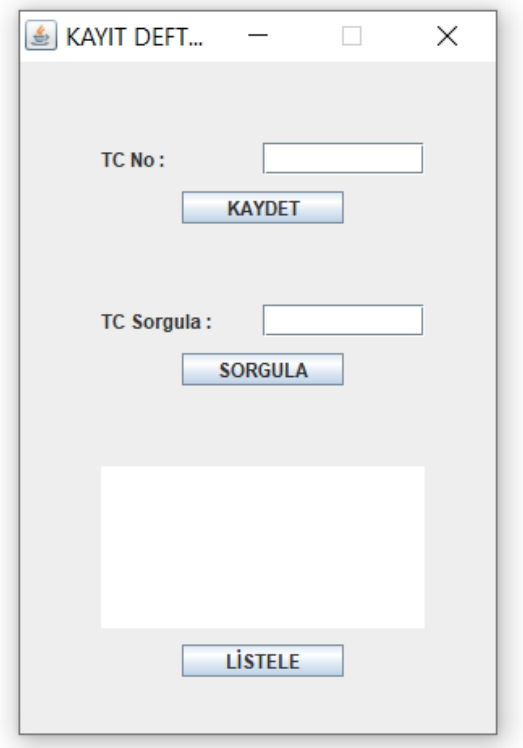
Kayıt defteri örneğinde temel amaç **swing** ve **awt** sınıflarının temel grafik ara yüz bileşenlerinin incelenmesi ve temel bileşenlerin uygulamasının gerçekleştirilmesidir. Temelde uygulamada **T.C. Kimlik** numarası kaydı gerçekleştirilmektedir. Bu kayıtlar **Kişilerim** isminde yine aynı paket içerisinde oluşturulmuş bir sınıftan üretilen nesnelere kaydedilmektedir. Bu kayıtlar içerisinden sorgu bölümü ile sorgulanma yapılabilmektedir. Son bölümde ise kaydı oluşturulan tüm veriler metin alanında listelenmektedir. Uygulamanın paket görünümü **Şekil 1.1'de** görülmektedir.



Şekil 1.1. Uygulama Paket Görünümü

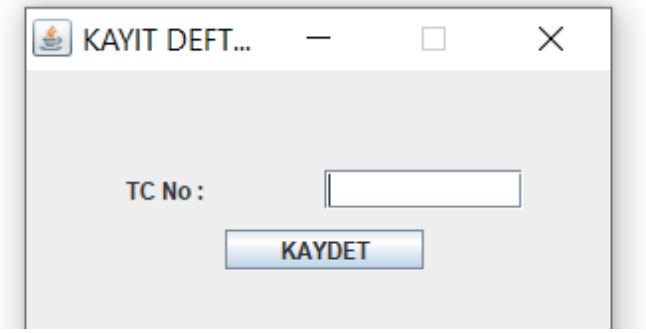
1.1.1. Uygulamanın Temel Özellikleri ve Çalışma Prensibi

Bu uygulama örneğinde **JFrame** sınıfından kalıtım yoluyla (**extends** ifadesi kullanılarak) Pencerem adında bir sınıf üretilmiştir. Kullanıcı ara yüzü temel olarak 3 adet bölümden oluşmaktadır. Uygulamaya ait pencere görüntüsü **Şekil 1.2'de** görülmektedir.



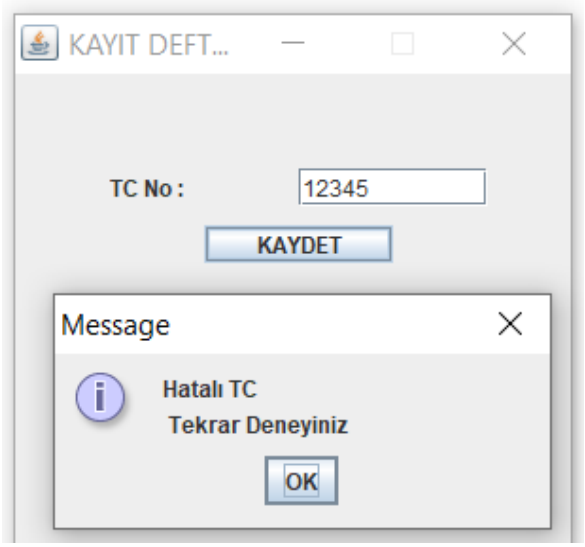
Şekil 1.2 Pencere Görüntüsü

a. Birinci bölümde, T.C. Kimlik numarası ile kayıt işleminin gerçekleştirilmesi ve bu kayıt işleminden önce kimlik numarasının uzunluğunun (11 hane) olup olmadığının yine tanımlanan bir fonksiyon ile kontrol edilmesi işlemleri gerçekleştirilmektedir.

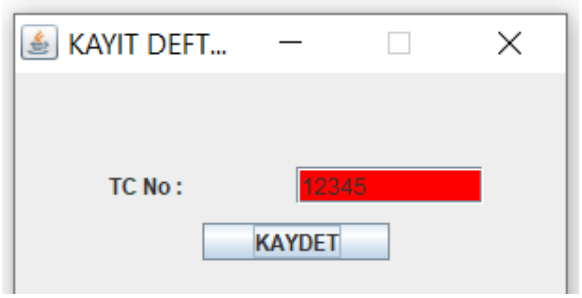


Şekil 1.3 Kayıt Bölümü

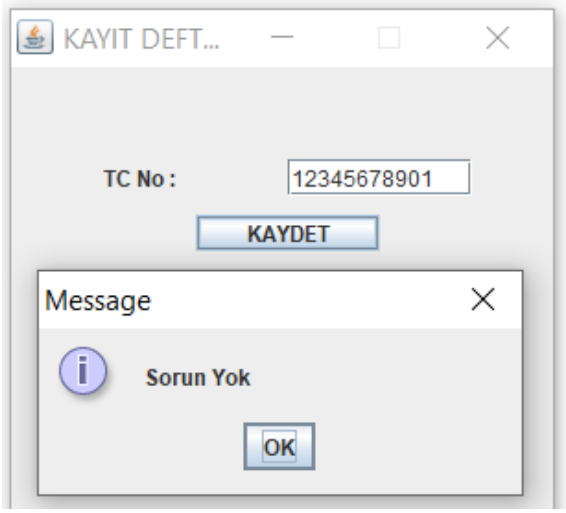
Uygulamada eğer T.C. Kimlik numarası 11 hane olarak girilmemişse metin kutusunun arka plan rengi kırmızıya dönmektedir ve ekrana hata mesajı gelmektedir.



Şekil 1.4 Hatalı Kimlik Numarası Girişi

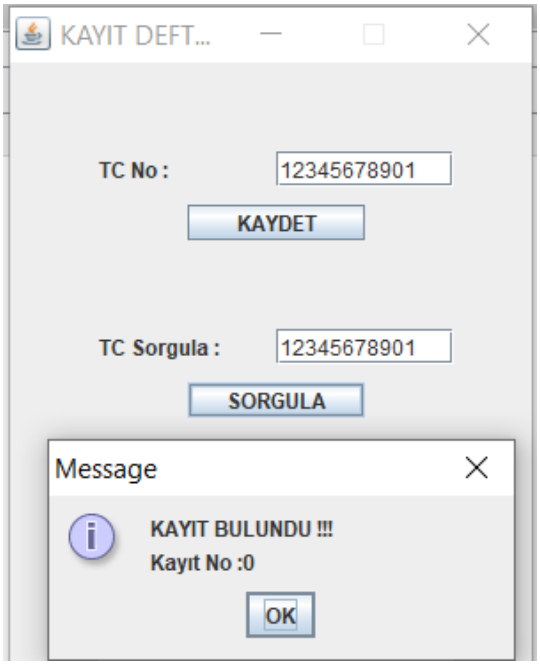


Şekil 1.5 Metin Alanı Renkli Uyarısı



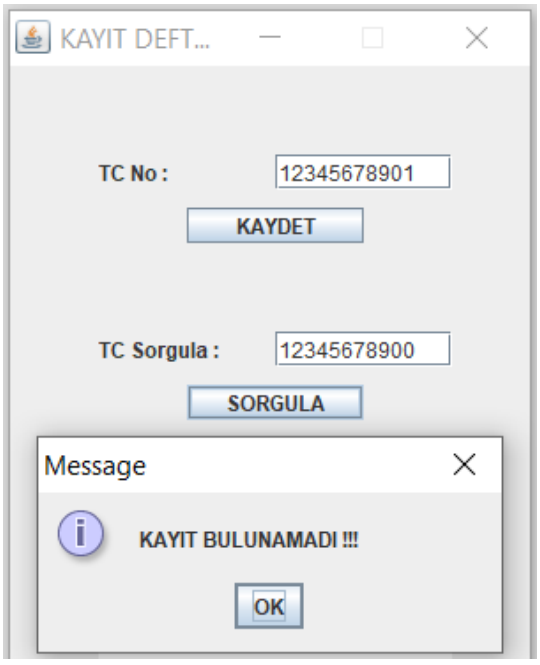
Şekil 1.6 Uygun Şekilde Girilen Kimlik Numarası

b. İkinci bölümde, Sorgu amacıyla tanımlanmış alandaki metin kutusuna kimlik numarası girerek kaydedilen veriler içerinden kontrol işlemi gerçekleştirilmektedir.



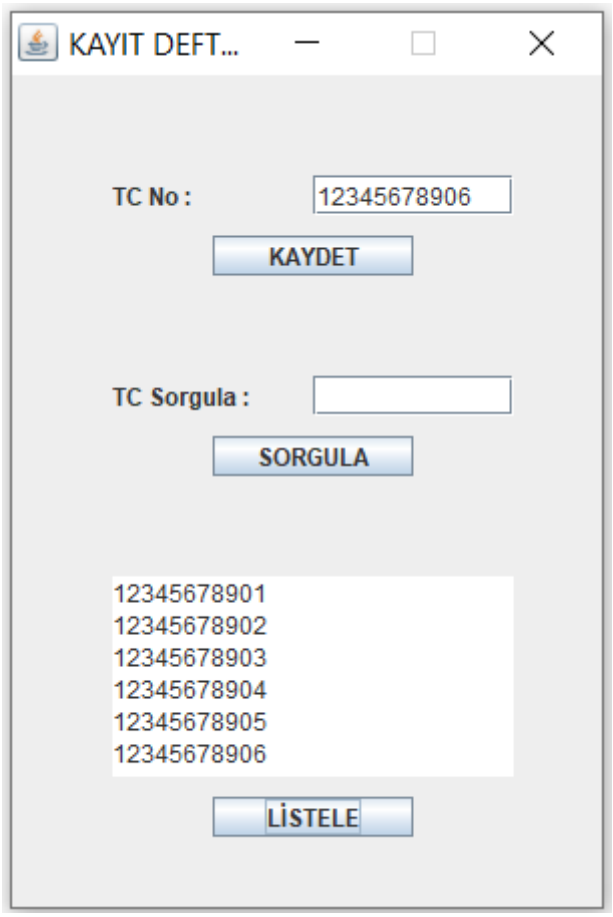
Şekil 1.7 Uygun Kaydın Bulunması ve Kayıt Numarası ile Ekrana Mesaj Olarak Verilmesi

Eğer ikinci bölümdeki sorgu metin alanına girilen kimlik numarası oluşturulan kayıtların arasında bulunmuyor ise hata mesajı oluşturulmaktadır.



Şekil 1.8 Uygun Kaydın Bulunmaması ve Hata Mesajı

c. Uygulamanın son bölümünde tüm kayıtlar metin alanında listelenmektedir. Aşağıda şekil 1.9'da örnek olarak sırayla sisteme kaydı yapılmış 6 adet kaydın metin alanında listelenmesi işlemi görülmektedir.



Şekil 1.9 Kayıtlı Tüm Verilerin Listelenmesi İşlemi

1.1.2. Kullanılan Sınıfların Özellikleri

1.1.2.1. Finale_Hazırlık_3.java

Bu Sınıf sadece main metod içermektedir ve JFrame sınıfından kalıtım yolu ile türetilen **Pencere** sınıfından türetilmiş **p** nesnesinin ekrana getirilmesini sağlamaktadır. Sınıfa ait kod aşağıda görülmektedir.

```
package finale_hazirlik_3;
public class Finale_Hazirlik_3 {
    public static void main(String[] args) {
        Pencere p = new Pencere();
    }
}
```

1.1.2.2. Kisilerim.java

Bu sınıf kayıt işlemi için kullanılacak sanal alanları oluşturmak için tanımlanmıştır. Kayıt butonuna her basıldığında kayıt işlemi hayali bir değişkende tutulmak zorundadır. Bu durumda eğer listeleme işlemi yapılmazsa kayıt yapılan değişkenin üzerinde yazma işlemi gerçekleştirilecek olup veri kaybı yaşanacaktır. Bir başka deyişle, bu sınıftan üretilecek nesneler ile ara bir bellek mekanizması oluşturulmuştur.

```
package finale_hazirlik_3;
public class Kisilerim {
    private String tcNo;
    public String getTC(){
        return tcNo;
    }
    public void setTC(String s){
        tcNo = s;
    }
}
```

Bu Sınıf içerisinde **String** tipte ve erişimi **private** olarak tanımlanmış dolayısıyla erişimine sınırlama getirilmiş **tcNo** isminde bir değişken bulunmaktadır. Bu şekilde tanımlanmış değişkenlerin değerlerine ulaşmak için genel olarak **set() "değer değiştirmek için"** ve **get() "değer temin etmek için"** fonksiyonları tanımlanmaktadır. Dolayısıyla bu şekilde tanımlı bir sınıftan bir nesne üretildiğinde bu nesnenin **private** değişkeninin değerini kullanmak için get(), değerini değiştirmek için set() fonksiyonu kullanılır.

Bu örnekte de **getTC()** fonksiyonu ile **tcNo** değişkeninin değerine ulaşılmaktayken, **setTC(String s)** fonksiyonu ile parametre olarak gönderilen **s** değişkeninin değeri **tcNo** değişkenine aktarılmaktadır.

1.1.2.3. Pencere.java

Bu sınıf kullanıcı ara yüzü oluşturmak amacıyla **swing** sınıfının üyesi **JFrame** alt sınıfından kalıktım yolu ile türetilmiş bir sınıftır. Bu sınıf türetme işlemi,

public class Pencere extends JFrame

ifadesi ile gerçekleştirilmektedir. **Pencere** sınıfı **JFrame** sınıfının tüm özelliklerini taşımaktadır.

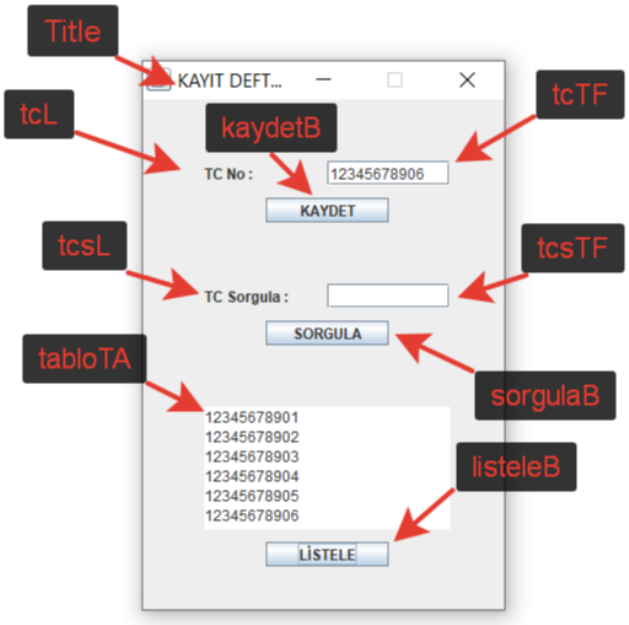
Sınıf türetim ifadesinin üzerinde Pencere sınıfında kullanılacak bileşenlerin uygulamada kullanılmasını sağlayacak **import** cümleleri bulunmaktadır.

```
package finale_hazirlik_3;
import java.awt.Color;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Locale;
import javax.swing.*;
```

Varsayılan kurucunun yani pencere çalıştığında sahip olması gereken temel özelliklerin kodlandığı **Pencere()** ifadesinden önceki bölümde, kullanılması planlanan değişken ve bileşenlerin ön tanımları gerçekleştirilmiştir.

```
public class Pencere extends JFrame {
    public static int WIDTH  = 300;
    public static int HEIGHT = 450;

    public static int kayit = 0;
    Kisilerim[] k =  new Kisilerim[10];

    private JLabel tcL, tcsL;
    private JTextField tcTF, tcsTF;
    private JButton kaydetB, sorgulaB, listeleB;
    private JTextArea tabloTA;

    private KaydetButtonHandler kbhandler;
    private SorgulaButtonHandler sbhandler;
    private ListeleButtonHandler lbhandler;
```

Gerçekleştirilen uygulamada pencere boyutları genişlik 300 ve yükseklik 450 birim olacak şekilde tanımlanmıştır. Diğer ifadeler kullanılan ifadeler tanımlaması yapılacak görsel bileşenleri ifade etmektedir.



Tanımlanan **handler** ifadeleri ile butonların hareketleri sonrası oluşacak eylemlerinin kodlanacağı aksiyon fonksiyonları ile ara bağlantı görevini üstelenen ön tanımlamalardır.

public static int kayit = 0;

Kisilerim[] k = new Kisilerim[10];

Yukarıdaki ifadeler ile 10 adet **Kisilerim** sınıfının özelliklerini taşıyan **k** nesnesi tanımlanmıştır. k nesneleri **k[0]**'dan başlayarak **k[9]**'a kadar 10 adet olarak tanımlanmıştır. Bu nesneler Kayıt işlemi esnasında sanal bir alan olarak kullanılacaktır. **kayit** değişkeni ise sayaç olarak kullanılmaktadır. Kaydet butonuna basıldığında ilk olarak, **kayit** değerinin başlangıç değeri 0 olduğundan ilk kayıt **k[0]** nesnesinin kimlik numarası olarak kaydedilecektir. Bu durum **KaydetButtonHandler** sınıfının aksiyon fonksiyonu altında açıkça kodlanmıştır.

Varsayılan kurucu fonksiyonu **Pencere()** ile Frame üzerindeki bileşenler ve Frame'in özellikleri kodlanmaktadır.

```
Pencere(){
    setTitle("KAYIT DEFTERİ");
    Container pane = getContentPane();
    setSize(WIDTH,HEIGHT);
    setLocation(300,300);
```

Sırasıyla yukarıdaki ifadeler ile, pencerenin ismi tanımlanmıştır, pencere üzerine öğelerin yerleştirileceği panel oluşturulmuş, pencerenin boyutları belirlenmiş ve pencerenin ana ekran üzerinde ilk defa görüneceği pozisyon tanımlanmıştır.

Aşağıdaki ifadeler ile görsel bileşenler sırasıyla tanımlanmış, pencere üzerindeki pozisyonları tanımlanmış ve görsel bileşenler panele eklenmiş (tutturulmuş) ve fonksiyonun son bölümünde yine fonksiyonun başlangıç bölümünde olduğu gibi düzenlemeler yapılmıştır.

```
setVisible(true);
setDefaultCloseOperation(EXIT_ON_CLOSE);
setResizable(false);
```

Fonksiyonun sonundaki bu ifadeler ile sırasıyla, pencerenin görünürlüğü ayarlanmış, kapanırlığı düzenlenmiş ve boyutunun kullanıcı tarafından değiştirile bilirliği engellenmiştir.

1.1.2.4. public boolean Tc_Kontrol(String x) Fonksiyonu

Bu fonksiyon parametre olarak String tipte bir değişkeni parametre olarak almakta ve çağırıldığı yere boolean tipte bir değer döndürmektedir.

```
public boolean Tc_Kontrol(String x){
        boolean durum = true;
        if(x.length()!=11){
            durum = false;
        }
        return durum;
    }
```

Fonksiyon temelde parametre x değişkeninin boyutunu length() fonksiyonunu kullanarak ölçmekte ve bu ölçüm sonucunda, x değişkeninin boyutunun 11 karakter olmaması durumunda çağırıldığı yere false (olumsuz) değer döndürürken, aksi durumda true (pozitif) değer döndürmektedir.

1.1.2.5. private class KaydetButtonHandler Sınıfı ve Aksiyon Fonksiyonu

Bu sınıf ve tanımlı aksiyon fonksiyonu ile kaydet butonuna basıldığında uygulamanın yapması gereken işler kodlanmıştır.

```
private class KaydetButtonHandler implements ActionListener{
      public void actionPerformed(ActionEvent e){
         if(Tc_Kontrol(tcTF.getText())==false){
            JOptionPane.showMessageDialog(null,"Hatalı TC \n Tekrar Deneyiniz");
            tcTF.setBackground(Color.red);
         }
         else{
            JOptionPane.showMessageDialog(null,"Sorun Yok");
            tcTF.setBackground(Color.white);
            k[kayit] = new Kisilerim();
            k[kayit].setTC(tcTF.getText());
            kayit++;
            System.out.println("Toplam Kayit : "+kayit);
         }
      }
   }
```

Fonksiyonun ilk bölümünde kontrol ifadesi içerisinde Tc_Kontrol() fonksiyonuna tcTF metin kutusu içerine girilen ifadenin içeriği parametre olarak gönderilmiş ve bu ifadenin uygunluğu kontrol edilmiştir. Kontrol fonksiyonundan dönen değer olumsuz (false) ise ekrana mesaj penceresi ile **"Hatalı TC \n Tekrar Deneyiniz"** ifadesi yazdırılmakta ardından metin kutusunun arka plan rengi **tcTF.setBackground(Color.red);** komutu kullanılarak kırmızıya dönüştürülmektedir.

Kontrol fonksiyonundan dönen değer olumlu ise, ekrana mesaj penceresi ile **"Sorun Yok"** görsel mesajı getirildikten sonra **tcTF.setBackground(Color.white);** komutu ile metin kutusunun arka plan rengi orijinal rengi olan beyaz renge dönüştürülmektedir. Bu işlemlerin ardından, kayit sayacının mevcut halini kullanarak **k[kayit] = new Kisilerim();** bir ara bellek nesnesi oluşturulmuş ve bu nesnenin ilgili alanına **k[kayit].setTC(tcTF.getText());** komutu ile

değer aktarımı yapılmıştır. Bu işlem ardından **kayit++;** komutu ile kayit değişkeni dolayısıyla kayıt sayısı güncellenmiştir.

1.1.2.6. private class SorgulaButtonHandler Sınıfı ve Aksiyon Fonksiyonu

Bu sınıf ve tanımlı aksiyon fonksiyonu ile sorgula butonuna basıldığında uygulamanın yapması gereken işler kodlanmıştır.

```
private class SorgulaButtonHandler implements ActionListener{
      public void actionPerformed(ActionEvent e){
         boolean sorgu = false;
         int i;
         for(i=0;i<kayit;i++){
            if(k[i].getTC().equals(tcsTF.getText())){
               sorgu =true; break;
            }
         }
         if(sorgu==false){
            JOptionPane.showMessageDialog(null,"KAYIT BULUNAMADI !!!");
         }
         else{
            JOptionPane.showMessageDialog(null,"KAYIT BULUNDU!!!\n
                                                      Kayıt No :"+i);
         }
      }
   }
```

Aksiyon fonksiyonu orta bölümde bulunan metin kutusu içerisine girilen değerin tüm kayıtlar içerisinde bulunup bulunmadığı kontrol edilmektedir. Tanımlanan döngü ifadesi ile döngü elemanı i değişkeninin değeri kayit değişkeninin son değerine ulaşıncaya kadar k nesnesinin tcNo değişkeninin değeri ile karşılaştırılmaktadır. Bu karşılaştırmada herhangi bir eşleşme yakalandığı durumda döngü sonlanmakta ve boolean tipi değişkenin değeri true olarak değiştirilmiştir. Döngü sonrası bu boolean değişkenin değeri kontrol edilerek kaydın bulunup bulunmaması ekrana bir mesaj penceresi ile kullanıcıya bildirilmektedir.

1.1.2.7. private class ListeleButtonHandler Sınıfı ve Aksiyon Fonksiyonu

Bu sınıf ve tanımlı aksiyon fonksiyonu ile listele butonuna basıldığında uygulamanın yapması gereken işler kodlanmıştır.

```
private class ListeleButtonHandler implements ActionListener{
      public void actionPerformed(ActionEvent e){
         int i;
         for(i=0;i<kayit;i++){
            tabloTA.append(k[i].getTC()+"\n");
         }
      }
   }
```

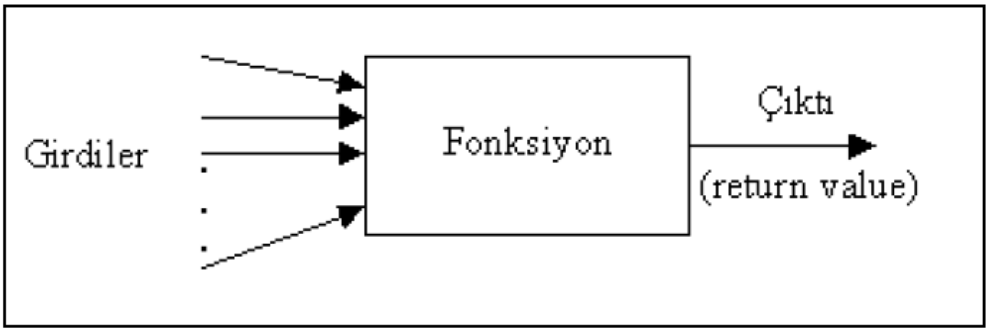
Aksiyon fonksiyonu ile kaydet butonu kullanılarak ara belleğe alınan kayıtların tümü tanımlı tabloTA metin alanı içerisine her kayıt alt alta gelecek şekilde yazdırılmaktadır. Döngü ifadesi ile döngü sayaç elemanı i değişkeni kayit değişkeninin değerine ulaşıncaya kadar her i değeri için tutulan kaydın metin alanına yazdırılması sağlanmaktadır.

Projeye ait kaynak kodları ekler bölümünde verilmektedir.

2. FONKSİYONLAR

(METOTLAR)

Fonksiyon, belirli sayıda verileri kullanarak bunları işleyen ve bir sonuç üreten komut grubudur. Her fonksiyonun bir adı ve fonksiyona gelen değerleri gösteren parametreleri (bağımsız değişkenleri) vardır. Genel olarak bir fonksiyon Şekil 2.1'deki gibi bir kutu ile temsil edilir:



Şekil 0.1 Fonksiyon Blok Tanımı

Fonksiyonların girdilerine parametre denir. Bir fonksiyon bu parametreleri alıp bir işleme tabi tutar ve bir değer hesaplar. Bu değer, çıktı veya geri dönüş değeri (return value) olarak adlandırılır. Unutmayın ki, bir fonksiyonun kaç girişi olursa olsun sadece bir çıkışı vardır.

Bir fonksiyon tanımlanırken önce erişim ifadesi (public, private, protected ya da friendly) sonra varsa geri dönüş tipi sonra fonksiyon adı ve parantez içerisinde ve yine var ise parametreleri tanımlanır. Örnek;

public int kinetikEnerji(Double m, Double v)

2.1. Temel Fonksiyon Tipleri ve Uygulaması

Aşağıda 4 ana başlıkta parametre durumuna göre ve geri dönüş tipi durumuna göre fonksiyonlar örnekler ile açıklanmaktadır.

2.1.1. F1() fonksiyonu

Bu fonksiyon parametresi olmayan ve çağırıldığı yere herhangi bir değer döndürmeyen bir fonksiyondur. Bu tip fonksiyonlar genelde ekrana ya da konsola mesaj yazan fonksiyonlardır.

```
public static void f1(){
      // Ekrana Gel Merhaba Yaz...
      System.out.println("MERHABA");
  }
```

Bu örnekte f1() fonksiyonu çağırıldığı zaman konsola "MERHABA" mesajını yazmaktadır.

2.1.2. F2(int x) fonksiyonu

Bu fonksiyon parametre olarak tamsayı tipte x adında bir değişkeni almaktadır. F1() fonksiyonundaki gibi bu fonksiyonda void yani geri dönüş tipi belirsiz bir fonksiyon olmakla beraber çağırıldığı yere değer döndürmemektedir.

```
public static void f2(int x){
      while(x>0){
        System.out.println("MERHABA");
        x--;
      }
  }
```

Bu örnekte **f2(int x)** fonksiyonu parametre olarak aldığı x tamsayısı kadar ekrana alt alta gelecek şekilde "MERHABA" yazmaktadır.

2.1.3. int f3() fonksiyonu

Bu fonksiyon tipinde parametre olarak herhangi bir değer alınmazken fonksiyon çağırıldığı yere istenen tipte değer döndürülebilmektedir.

```
public static int f3(){
    // Çağırıldığı yere 5 döndürsün...
    return 5;
  }
```

Bu örnekteki f3() fonksiyonu parametre almamakta ama çağırıldığı yere tamsayı olarak tanmlı bir değeri döndürmektedir.

2.1.4. int f4(int a, int b)

Bu fonksiyon tipi en yaygın şekilde kullanılan fonksiyon tiplerinden biridir. Bu fonksiyon tipinde hem parametre kullanımı hem de geri dönüş tipi tanımlanmaktadır.

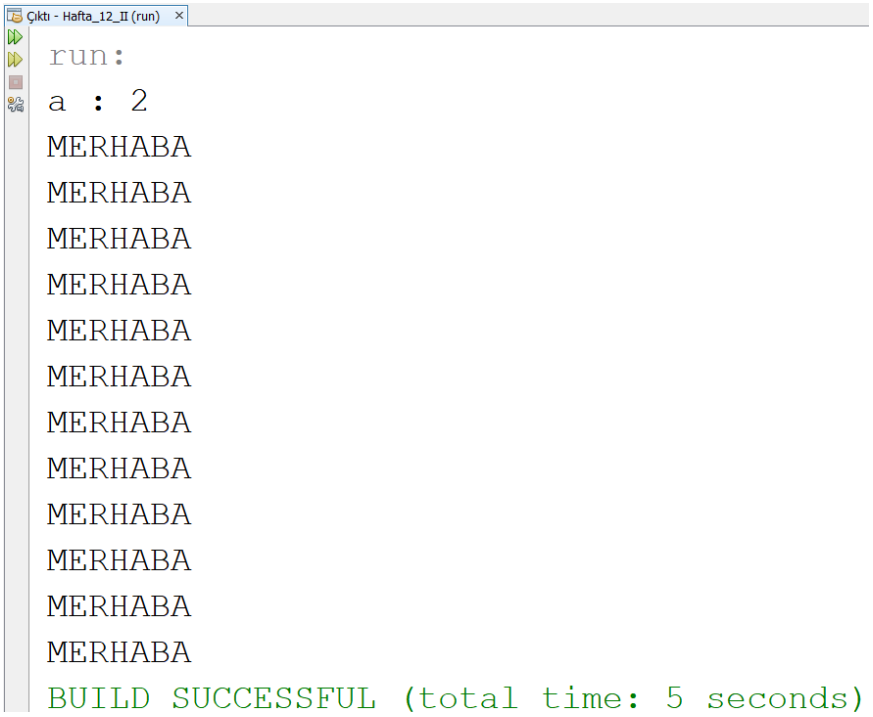
```
public static int f4(int a, int b){
    return a*b;
  }
```

Bu örnekte f4() fonksiyonu parametre olarak tamsayı tipinde a ve b sayıları alınmaktadır. Fonksiyon çağırıldığı yere tamsayı değer döndürecek şekilde tanımlanmıştır. Geri döndürülecek bu tamsayı değer fonksiyon içinde parametre olarak gönderilen a ve b sayılarının çarpımını döndürerek gerçekleştirmektedir.

2.1.5. Ana Fonksiyon

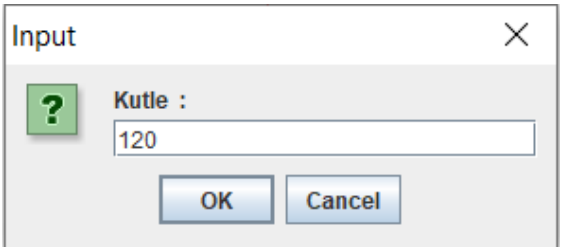
Yukarıda bahsedilen fonksiyonlar tanımlanmış bir sınıf ve yine tanımlı bir ana fonksiyon içerisinde kullanılmıştır. Uygulamanın kodu ve ekran çıktısı aşağıda görülmektedir.

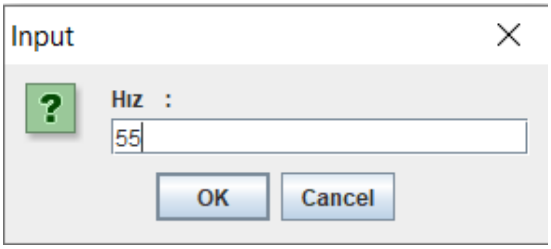
```
package hafta_12_ıı;

import java.util.Scanner;

public class Fonksiyonlar {
   public static void main(String[] args){
      int a,b;
      Scanner x = new Scanner(System.in);
      System.out.print("a : ");
      a = x.nextInt();
      f1();
      f2(a);
      f2(f3());
      f2(f4(f3(),a));
   }
   public static void f1(){
      // Ekrana Gel Merhaba Yaz...
      System.out.println("MERHABA");
   }
   public static void f2(int x){
      // Ekrana parametre x kadar MERHABA yaz...
      while(x>0){
        System.out.println("MERHABA");
        x--;
      }
   }
   public static int f3(){
      // Çağırıldığı yere 5 döndürsün...
      return 3;
   }
   public static int f4(int a, int b){
      return a*b;
   }
}
```

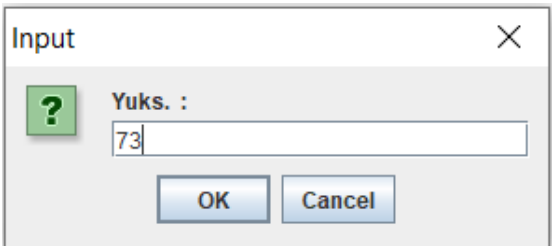

Çıktı - Hafta_12_II (run)
run:
a : 2
MERHABA
MERHABA
MERHABA
MERHABA
MERHABA
MERHABA
MERHABA
MERHABA
MERHABA
MERHABA
MERHABA
MERHABA
BUILD SUCCESSFUL (total time: 5 seconds)

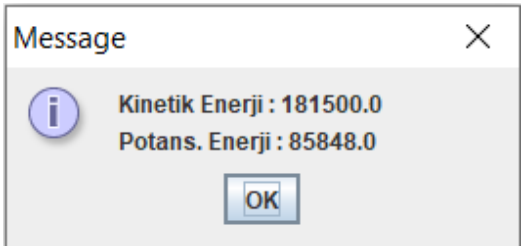
2.2. Fizikte Kullanılan Temel Fonksiyonlar (Kinetik ve Potansiyel Enerji)

```
package hafta_12_ıı;

import javax.swing.JOptionPane;

public class FizikFonksiyonları {
    public static void main(String[] args){
        double m,v,h;
        String ms,vs,hs,sonuc;
        ms = JOptionPane.showInputDialog("Kutle  :");
        vs = JOptionPane.showInputDialog("Hız    :");
        hs = JOptionPane.showInputDialog("Yuks.  :");
        m = Double.parseDouble(ms);
        v = Double.parseDouble(vs);
        h = Double.parseDouble(hs);
        sonuc  = "Kinetik Enerji : "+kinetik(m,v)+"\n";
        sonuc += "Potans. Enerji : "+potans(m,h);
        JOptionPane.showMessageDialog(null,sonuc);
    }
    public static double kinetik(double m, double v){
        return (0.5*m*Math.pow(v,2));
    }
    public static double potans(double m, double h){
        return (m*9.8*h);
    }
}
```



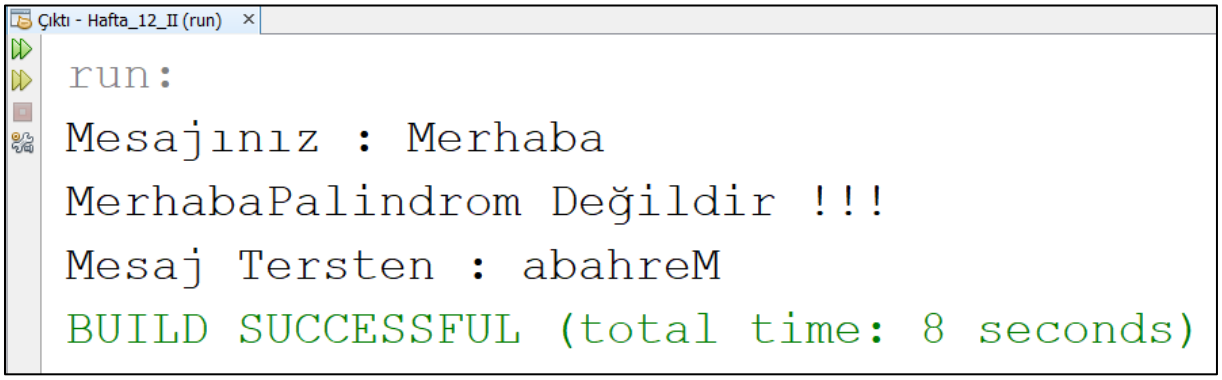






2.3. Palindrom ve Tersleye Fonksiyonlar

```
package hafta_12_ıı;
import java.util.Scanner;
public class Palindrome {
   public static void main(String[] args){
      Scanner x = new Scanner(System.in);
      System.out.print("Mesajınız : ");
      String m = x.next();
      if(pali(m)==true)
         System.out.println(m+"Palindromdur !!!");
      else
         System.out.println(m+"Palindrom Değildir !!!");
      System.out.println("Mesaj Tersten : "+ters(m));
   }
   public static boolean pali(String s){
      boolean message = true;
      for(int i=0;i<(s.length()-1)/2;i++){
         if(s.charAt(i) != s.charAt(s.length()-1-i)){
            message = false;
         }
      }
      return message;
   }
   public static String ters(String s){
      int boy = s.length()-1;
      String yeni = "";
      while(boy>=0){
         yeni = yeni + s.charAt(boy);
         boy--;
      }
      return yeni;
   }
}
```



EKLER

Kayıt Defteri Uygulaması

Finale_Hazirlik_3.java

```
package finale_hazirlik_3;
public class Finale_Hazirlik_3 {
   public static void main(String[] args) {
      Pencere p = new Pencere();
   }
}
```

Kisilerim.java

```
package finale_hazirlik_3;
public class Kisilerim {
   private String tcNo;
   public String getTC(){
      return tcNo;
   }
   public void setTC(String s){
      tcNo = s;
   }
}
```

Pencere.java

```
package finale_hazirlik_3;

import java.awt.Color;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Locale;
import javax.swing.*;

public class Pencere extends JFrame {

   public static int WIDTH  = 300;
   public static int HEIGHT = 450;

   public static int kayit = 0;
   Kisilerim[] k =  new Kisilerim[10];

   private JLabel tcL, tcsL;
   private JTextField tcTF, tcsTF;
   private JButton kaydetB, sorgulaB, listeleB;
```

```
private JTextArea tabloTA;

private KaydetButtonHandler kbhandler;
private SorgulaButtonHandler sbhandler;
private ListeleButtonHandler lbhandler;

Pencere(){
   setTitle("KAYIT DEFTERİ");
   Container pane = getContentPane();
   setSize(WIDTH,HEIGHT);
   setLocation(300,300);

   /*  Kayıt İşlemi İçin Gerekli Bileşenlerin Tanımları  */
   tcL      = new JLabel("TC No : ");
   tcTF     = new JTextField();
   kaydetB  = new JButton("KAYDET");
   kbhandler = new KaydetButtonHandler();
   kaydetB.addActionListener(kbhandler);
   /*////////////////////////////////////////////////////*/
   /*  Sorgu İşlemi İçin Gerekli Bileşenlerin Tanımları  */
   tcsL     = new JLabel("TC Sorgula : ");
   tcsTF    = new JTextField();
   sorgulaB  = new JButton("SORGULA");
   sbhandler = new SorgulaButtonHandler();
   sorgulaB.addActionListener(sbhandler);
   /*////////////////////////////////////////////////////*/
   /*  Listeleme İşlemi İçin Gerekli Bileşenlerin Tanımları  */
   tabloTA   = new JTextArea();
   listeleB  = new JButton("LİSTELE");
   lbhandler = new ListeleButtonHandler();
   listeleB.addActionListener(lbhandler);
   /*////////////////////////////////////////*/

   pane.setLayout(null);

   /*  Kayıt İşlemi İçin Gerekli Bileşenler  */
   tcL.setLocation(50,50);
   tcL.setSize(100,20);
   pane.add(tcL);
   tcTF.setLocation(150,50);
   tcTF.setSize(100,20);
   pane.add(tcTF);
   kaydetB.setLocation(100,80);
   kaydetB.setSize(100,20);
   pane.add(kaydetB);
   /*////////////////////////////////////////*/
   /*  Sorgulama İşlemi İçin Gerekli Bileşenler  */
   tcsL.setLocation(50,150);
   tcsL.setSize(100,20);
   pane.add(tcsL);
```

```
        tcsTF.setLocation(150,150);
        tcsTF.setSize(100,20);
        pane.add(tcsTF);
        sorgulaB.setLocation(100,180);
        sorgulaB.setSize(100,20);
        pane.add(sorgulaB);
        /*////////////////////////////////////////////////*/
        /*  Listeleme İşlemi İçin Gerekli Bileşenler  */
        tabloTA.setLocation(50,250);
        tabloTA.setSize(200,100);
        pane.add(tabloTA);
        listeleB.setLocation(100,360);
        listeleB.setSize(100,20);
        pane.add(listeleB);
        /*////////////////////////////////////*/

        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setResizable(false);
    }
    public boolean Tc_Kontrol(String x){
        boolean durum = true;
        if(x.length()!=11){
            durum = false;
        }
        return durum;
    }
    private class KaydetButtonHandler implements ActionListener{
        public void actionPerformed(ActionEvent e){
            if(Tc_Kontrol(tcTF.getText())==false){
                JOptionPane.showMessageDialog(null,"Hatalı TC \n
Tekrar Deneyiniz");
                tcTF.setBackground(Color.red);
            }
            else{
                JOptionPane.showMessageDialog(null,"Sorun Yok");
                tcTF.setBackground(Color.white);
                k[kayit] = new Kisilerim();
                k[kayit].setTC(tcTF.getText());
                kayit++;
                System.out.println("Toplam Kayit : "+kayit);
            }
        }

    }
    private class SorgulaButtonHandler implements
ActionListener{
        public void actionPerformed(ActionEvent e){
            boolean sorgu = false;
            int i;
```

```
        for(i=0;i<kayit;i++){
           if(k[i].getTC().equals(tcsTF.getText())){
              sorgu =true; break;
           }
        }
        if(sorgu==false){
           JOptionPane.showMessageDialog(null,"KAYIT
BULUNAMADI !!!");
        }
        else{
           JOptionPane.showMessageDialog(null,"KAYIT
BULUNDU !!!\nKayıt No :"+i);
        }
     }
  }
  private class ListeleButtonHandler implements ActionListener{
     public void actionPerformed(ActionEvent e){
        int i;
        for(i=0;i<kayit;i++){
           tabloTA.append(k[i].getTC()+"\n");
        }
     }
  }
}
```