Sarrera

Laborategi honetan login eta passwork baieztatzen duen aplikazio bat garatuko dugu, negozio logika eta interfaze grafikoa bi maila desberdinetan definituz.

Helburuak

Laborategi honen helburuak hurrengoak dira:

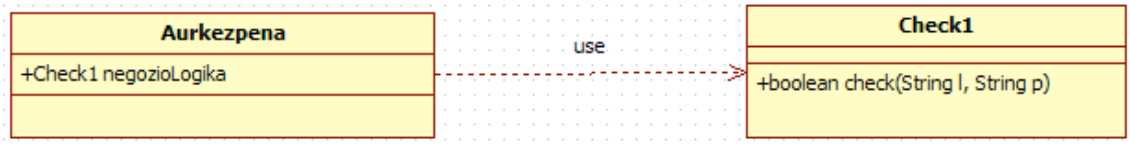
- Aurkezpen eta negozioaren logika maila nola banatu daitezkeen ulertzea.
- Java interfazeen erabilgarritasuna azaltzea: aukera ematen dute klase bateko objektu batetik **ezagutzen ez den** beste klase bateko objektu bati eskatzeko exekuzio denboran metodo zehatz bat egikari dezan (metodoaren izena bai izango da ezaguna Java interfazean definituta egongo delako)
- Ulertzea ez dela beharrezkoa aurkezpen maila birkonpilatzea negozioaren logika aldatzean, eta gainera **exekuzio denboran** egitea posible dela.

Jarraitzeko pausoak

Aplikazio garatzeko 3 prototipo sortuko ditugu.

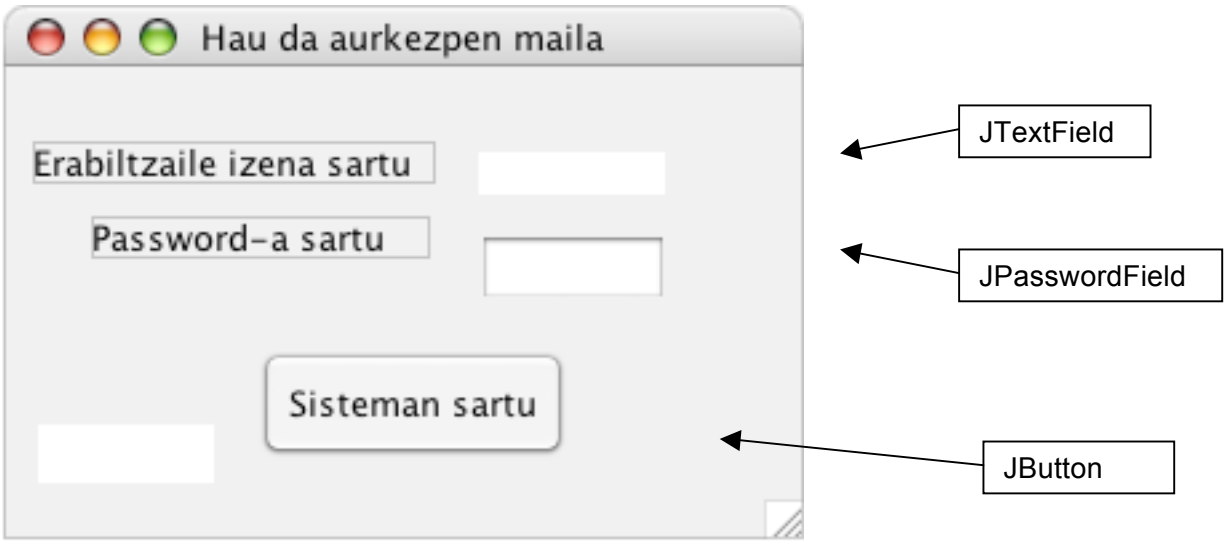
1. Lehendabiziko prototipoa

Lehendabiziko prototipoan 2 klase definituko ditugu. Lehendabiziko klasean interfaze grafikoa definituko dugu eta bigarrengoan negozio logika hurrengo irudian agertzen den bezala:



Horretarako hurrengo pausoak jarraitu.

a) Eclipse proiektu berri bat sortu.
b) Frame berri bat sortu (Aurkezpena klasea) ondoko itxurarekin. Klase hau sortzeko File>>New>>Visual class aukeratu, eta jarraian Swing>>Frame klasea sakatu. Jarraian Frame-a osatu bi label, bi JtextField eta botoi bat hurrengo irudian agertzen den bezala:



Aurkezpena klasea

Bukatzeko beste JtextField bat definituko dugu emaitza emateko (OK edo Errorea).

c) Negozioaren Check1 klase bat definitu login eta password-a baieztatzen duena. Gogoratu klase hau "logic" paketean definitzea. Hurrengo kodean, login eta password-a baieztatzen da biak berdinak direnean.

```
package logic;

public class Check1 {
 public boolean check(String login, String password){
      return login.compareTo(password)==0;
 }
}
```

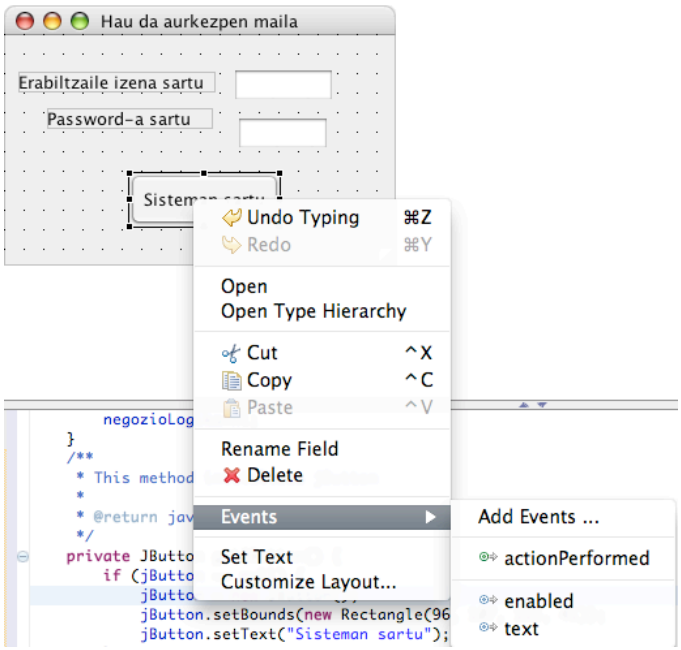
d) Aurkezpena klasean Check1 motako atributu bat sortu negozio Logikaren objektu bat gordetzeko.

```
    private Check1 negozioLogika;
```

e) Aurreko atributuari balioak esleitzeko, setNegozioLogika metodoa definitu Aurkezpena klasean:

```
public void setNegozioLogica(Check1 nl){
          negozioLogika=nl;
     }
```

f) Botoiaren gertaera kudeatu. Botoian ipini, eskuineko botoia sakatu eta "*actionPerformed*" gertaeraren metodoa definitu hurrengo irudian agertzen den bezala:



Metodo hau exekutatuko da botoia sakatzen dugun bakoitzean.

g) Botoia sakatzen den metodoa inplementatu. Gogoratu sarrerako datuak "nameInput" eta "passInput" jTextField objektuetan gordeta daudela.

```
private JButton getJButton() {
     if (jButton == null) {
      jButton = new JButton();
      jButton.setBounds(new Rectangle(96, 107, 115, 41));
      jButton.addActionListener(new
java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent e) {
      String izena=nameInput.getText();
      String pass=passInput.getText();
      boolean b=negozioLogika.check(izena, pass);
      if (b) jTextArea.setText("Aurrera");
        else jTextArea.setText("Errorea");
    }
  });
}
return jButton;}
```

h) Klase jaurtitzailea sortu.Klase honetan Aurkezpena objektu bat sortzen da, Negozio Logikaren objektu bat sortu eta Aurkezpenari esleitzen zaio eta aplikazioa martxan ipintzen da. Hau da kodea:

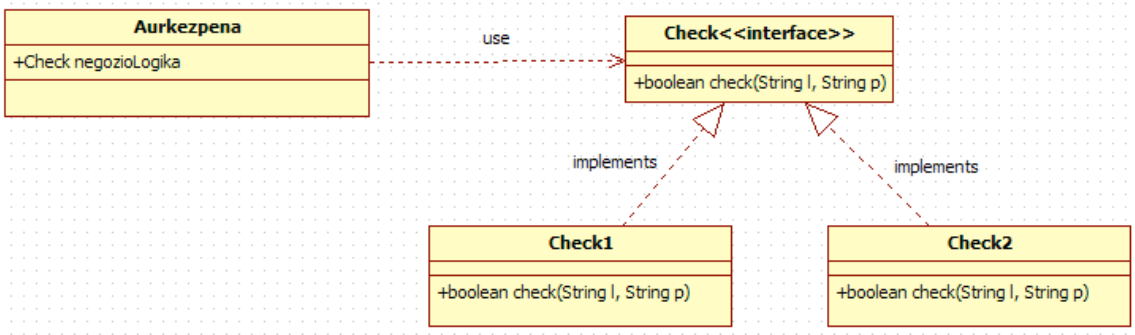
```
public class Jaurtitzailea {

      public static void main(String[] args) {
       Aurkezpena a=new Aurkezpena();
       Check1 nl=new Check1();
       a.setNegozioLogica(nl);
       a.setVisible(true);
      }
}
```

Prototipoa bukatuta dago. Hobetu daitekeen gauzak:

Zer gertatzen da NegozioLogika Check1 klasean egon ordez Check2 klasean badago? Nola aldatu dezakegu Aurkezpen klasearen Negozio Logikaren objektua, **Aurkezpen klasearen inongo kode lerrorik aldatu gabe**?

2. Bigarren prototipoa

Bigarrengo prototipoan Aurkezpena eta negozio logikaren erlazioa interfaze klase berezi baten bidez definituko da hurrengo irudian agertzen den bezala.



Interfaze bat metodo batzuen "espezifikazio" bat definitzen du. Klase berezi honetan ez da implementatzen inongo metodorik, metodoen signaturak agertzen dira soilik. Jarraian Check interfazearen kodea agertzen da:

```
package logic;

public interface Check {
 public boolean check(String l, String p);
}
```

Jarraian klase konkretu batzuk interfaze hauek inplementatuko dituzte. Klase bat interfaze bat inplementatzen badu, interfazean agertzen diren metodo guztiak

inplementatu beharko ditu. Jarraian Check1 klasea birdefinitu dugu Check interfazea inplementatuz:

```
package logic;

public class Check1 implements Check{
 public boolean check(String login, String password){
      return login.compareTo(password)==0;
 }
}
```

Klase eta interfazearen arteko erlazio ulertzea oso garrantzitsua da. Klase bat interfaze bat inplementatzen badu, klase horretako objektuak interfaze klasearen motako objektuak dira, hau da, gure klasuan, hurrengoa definitu dezakegu:

```
Check nl=new Check1();
```

Check1 klasea Check interfazea inplementatzen duenez, Check1-eko objektuak Check interfazeren motakoak dira.

Jarraian gure prototipoa egitura berri honetara egikitzeko aldaketa batzuk egingo ditugu Aurkezpena klasean.

a) Aurkezpena klasean Check interfaze motako atributu bat definituko dugu. Honek esan nahi du, Check interfazea inplementatzen duen edozein klasearen objektuak erreferentzia-tu daitekeela atributu honetatik:

```
     private Check negozioLogika;
```

b) Aurreko atributuari balioak esleitzeko, setNegozioLogika metodoa birdefinitu:

```
public void setNegozioLogica(Check nl){
          negozioLogika=nl;
     }
```

c) Klase Jaurtitzailea aldatu:

```
public class Jaurtitzailea {
  public static void main(String[] args) {
      Aurkezpena a=new Aurkezpena();
      Check nl=new Check1();
      a.setNegozioLogica(nl);
      a.setVisible(true);

     }
}
```

Orain arte,lehendabiziko prototipoaren funtzionalitate berdina lortu dugu. Orain beste negozio logika desberdin bat definituko dugu:

d) Negozio logika berria sortu. Kasu honetan sistemaren sarrera baieztatuko du login-a eta password-a karaktere kopuru berdina dutenean. Hau da kodea:

```
package logic;

public class Check2 implements Check{
      public boolean check(String l, String p){
           return a.lenght==p.lenght;     }
}
```

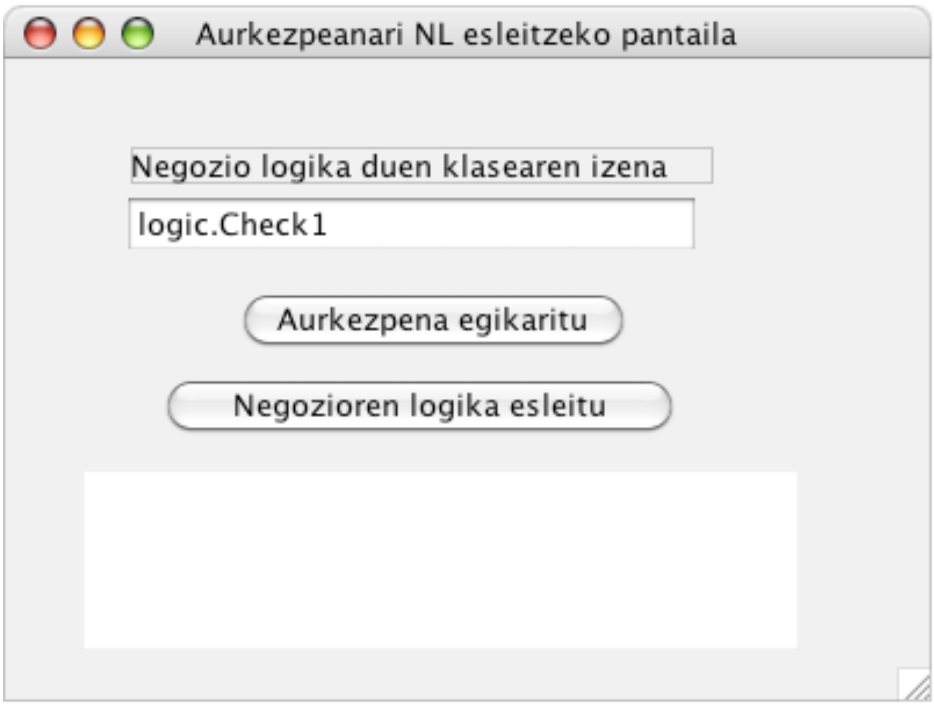
e) Aurkezpenaren Negozio Logika aldatu bere kodea aldatu gabe. Programa jaurtitzailean aurkezpenari esleitzen zaion negozio logika objektua aldatu. Biak Check interfazea inplementatzen dutenez ez dago arazorik.

```
public class Jaurtitzailea {

  public static void main(String[] args) {
      Aurkezpena a=new Aurkezpena();
      Check nl=new Check2();
      a.setNegozioLogica(nl);
      a.setVisible(true);

     }
}
```

Aplikazioa exekutatu eta frogatu.

3. Hirugarren prototipoa

Azkeneko prototipoan Aurkezpenaren negozio logika exekuzioan aldatzeko aukera garatzen duen aplikazioa definituko dugu. Horretarako hurrengo pantaia definituko dugu (`GUIJaurtitzailea` klasea):



Lehendabiziko JTextField-an Aurkezpenari esleitu nahi diogun negozio logikaren klasea idatziko da. "Negozioaren logika esleitu" botoiak Aurkezpena klaseari idatzitatako klasearen objektu bat esleituko dio. Bukatzeko "Aurkezpena egikaritu" Aurkezpen klasea irudikatuko du.

Hauek dira jarratitu behar diren pausoak aplikazio hau garatzeko:

a) `GUIJaurtitzailea` klasean Aurkezpena klasearen aldagai bat sortu:

```
public class GUIJaurtitzailea extends JFrame {
     private Aurkezpena aurkezpena=new Aurkezpena();
```

b) Negozio logika esleitu botoian, "actionPerformed" gertaera kudeazen duen metodo bat definitu hurrengo kodearekin:

```
private JButton getJButton1() {
 if (jButton1 == null) {
  jButton1 = new JButton();
  jButton1.setBounds(new Rectangle(65, 137, 229, 29));
  jButton1.setText("Negozioren            logika            esleitu");
jButton1.addActionListener(new java.awt.event.ActionListener() {
 public void actionPerformed(java.awt.event.ActionEvent e) {
  try {
  Check                              nl                                =
(Check)Class.forName(jTextField.getText()).newInstance();
  aurkezpena.setNegozioLogica(nl);
 } catch (Exception e1) {
      e1.printStackTrace();
   }
}
});
}
```

Class.forName metodoak String batean pasatzen zaion klase bat kargatzen du exekuzio gararaian. Jarraian newInstance kargatutako klasearen objektu bat sortzen du. Objektua sortu ondoren aurkezepanari pasatzen zaio setNegozioLogika metodoaren bidez.

c) Aurkezpena egikaritu botoian, "actionPerformed" gertaera kudeazen duen metodo bat definitu hurrengo kodearekin:

```
jButton.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(java.awt.event.ActionEvent e) {
    aurkezpena.setVisible(true);
}
```

d) main metodo bat sortu GUIJaurtitzaile klasean:

```
public static void main(String[] args) {
     GUIJaurtitzailea gui=new GUIJaurtitzailea();
     gui.setVisible(true);
}
```