MANUAL BOOK GAME

PROJECT LAB REKAYASA PERANGKAT LUNAK 2

Nama : Sion Eleazar Santoso

NPM : 51420197

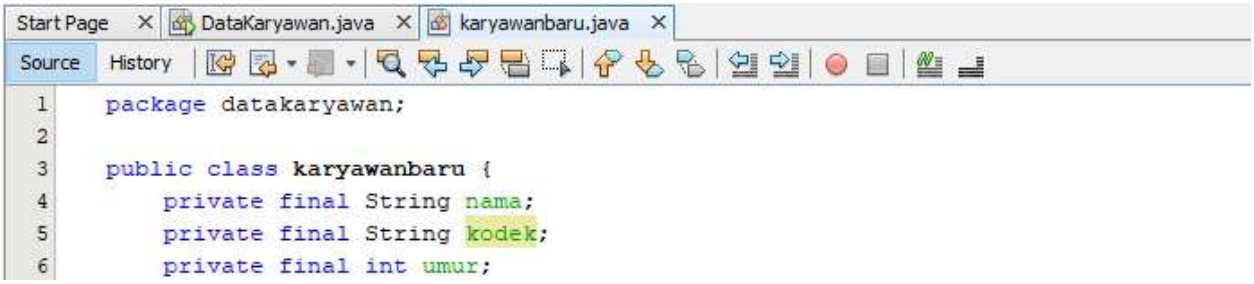
Kelas : 4IA16

LABORATORIUM TEKNIK INFORMATIKA

UNIVERSITAS GUNADARMA

2023

MANUAL BOOK REKAYASA
PERANGKAT LUNAK 2




```
1   package datakaryawan;
2
3   public class karyawanbaru {
4       private final String nama;
5       private final String kodek;
6       private final int umur;
```

Langkah awal yaitu dengan import paket dan kelas yang diperlukan. Pada pertemuan kali ini dimulai dengan mendefinisikan paket yang ada pada netbeans dan mengimport “ArrayList”, “List”, dan “Scanner”.

```
7   public class DataKaryawan {
8       public static void main(String[] args) {
9           List<karyawanbaru> daftarKaryawan = new ArrayList<>();
10          Scanner input = new Scanner(System.in);
```

Selanjutnya, mendeklarasikan class dengan nama “DaftarKaryawan” menjadi Public, setelah itu variable “daftarkaryawan” merupakan *List* yang akan digunakan untuk menyimpan data karyawan dan “input” merupakan *Scanner* yang akan digunakan untuk menerima masukan dari pengguna.

```
12          while (true) {
13              System.out.println("Tambah Data Karyawan Baru (y/n): ");
14              String jawaban = input.nextLine();
```

Lalu, membuat loop utama agar program berjalan dalam loop tak terbatas untuk terus memproses data karyawan, setelah itu, pada line ke-13 dan 14 berfungi untuk input konfirmasi penambahan data karyawan dimana program akan mencetak pesan dan mengunggu pengguna memasukkan “y” untuk ya atau “n” untuk tidak.

```
16              if (jawaban.equalsIgnoreCase("n")) {
17                  break;
18              }
```

Jika user memasukkan “n” maka perintah if akan aktif, dimana program akan keluar dari loop.

```
19               System.out.print("Nama: ");
20              String nama = input.nextLine();
21
22              System.out.print("Kode Karyawan: ");
23              String kodek = input.nextLine();
```

ne ke-27 dimana program akan meminta user unruk memasukkan data karyawan, seperti nama, kode karyawan, dan umur. Dari masing – masing perintah yang diinput akan muncul tulisan “Nama: “, “Kode Karyawan: “, dan “Umur: “ dari perintah System.out.print() untuk setiap data yang dimasukkan, diharapkan sesuai seperti String untuk character dan Integer untuk angka.

```
25              System.out.print("Umur: ");
26              int umur = input.nextInt();
27              input.nextLine();
```

Setelah itu, membuat objek karyawan dan menambahkannya ke daftar. Objek “karyawanbaru” dibuat dengan data yang dimasukkan pengguna, dan kemudian objek tersebut ditambahkan ke “daftarkaryawan”.

```
29              karyawanbaru karyawan = new karyawanbaru(nama, kodek, umur);
30              daftarKaryawan.add(karyawan);
31          }
```

Langkah selanjutnya yaitu untuk menampilkan daftar karyawan dengan codingan di atas, dimana saat loop berakhir, program mencetak daftar karyawan yang telah dimasukkan oleh user.

```
32            System.out.println("Daftar Karyawan:");
              for (karyawanbaru karyawan : daftarKaryawan) {
34                System.out.println(karyawan);
35            }
36        }
37    }
```

Pada project selanjutnya yaitu membuat user bisa menginput jika ada karyawan baru. Dimulai dengan mendeklarasikan paket "datakaryawan" dari netbeans dan setelah itu mendeklarasikan "karyawanbaru" sebagai class public. Lalu mendeklarasikan atribut untuk nama, kode karyawan, dan umur. Semua atribut bersifat private dimana hanya dapat diakses di dalam class "karyawanbaru".

```
8       public karyawanbaru(String nama, String kodek, int umur) {
9           this.nama = nama;
10          this.kodek = kodek;
11          this.umur = umur;
```

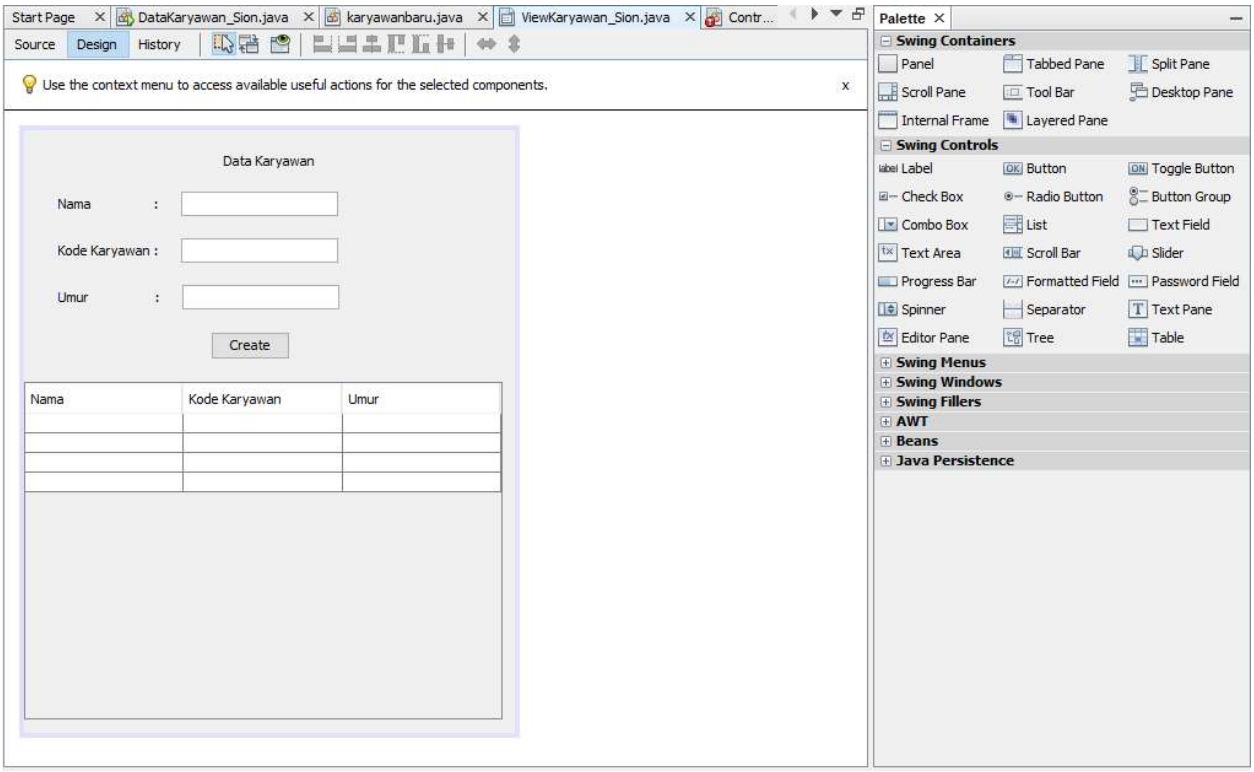
Selanjutnya yaitu membuat konstruktor, yang digunakan untuk membuat objek "karyawanbaru". Saat objek telah dibuat, konstruktor akan dijalankan, dan nilai – nilai untuk atribut "nama", "kodek", dan "umur" diatur.

```
13      public String getNama() {
14          return nama;
15      }
16      public String getKodek() {
17          return kodek;
18      }
19      public int getUmur() {
20          return umur;
21      }
```

Lalu, menggunakan metode Getter pada Java, dimana "getNama()", "getKodek()", dan "getUmur()" merupakan metode getter yang digunakan untuk mengakses nilai atribut "nama", "kodek", dan "umur" karena atribut tersebut bersifat private, sehingga perlu menggunakan getter untuk mengakses nilai atribut dari luar kelas.

```
23      @Override
        public String toString() {
25          return "Nama: "+ nama + ", Kode Karyawan: " + kodek + ", Umur: " + umur;
26      }
27  }
```

Untuk langkah terakhirnya yaitu menggunakan metode toString() dimana metode yang digunakan untuk menggambarkan objek "karyawanbaru" dalam bentuk string. Ketika user mencetak objek "karyawanbaru", metode toString() akan dipanggil untuk mengembalikan string yang berisi informasi tentang Nama, Kode karyawan, dan umur karyawan agar objeknya dapat diubah dengan mudah menjadi string dengan tujuan pencetakan atau representasi.



Langkah selanjutnya yaitu dengan mengimport libraries terlebih dahulu yaitu JDBC setelah sudah, maka membuat tampilan seperti gambar di atas dengan menggunakan 3 Label, 3 Text Field, dan table.

Sebelum melanjutkan untuk mengedit source pada viewkaryawan, membuat controller nya terlebih dahulu. Untuk langkah awal nya yaitu mengimport Arraylist dan table model yang ingin digunakan, tidak lupa untuk menggunakan package data.karyawan_sion.

```
6   public class ControllerKaryawan {
7       ArrayList<karyawanbaru> ArrayData;
8       DefaultTableModel tablelist;
```

Lalu, menginisialisasi ArrayList dan DefaultTableModel dengan mendeklarasikannya di class public dari controller karyawan, dimana “ArrayData” merupakan ArrayList yang akan digunakan untuk menyimpan objek “karyawanbaru” yang merupakan kelas dari data karyawan dan untuk “tablelist” merupakan variable yang akan digunakan untuk menyimpan model data yang akan ditampilkan di komponen JTable.

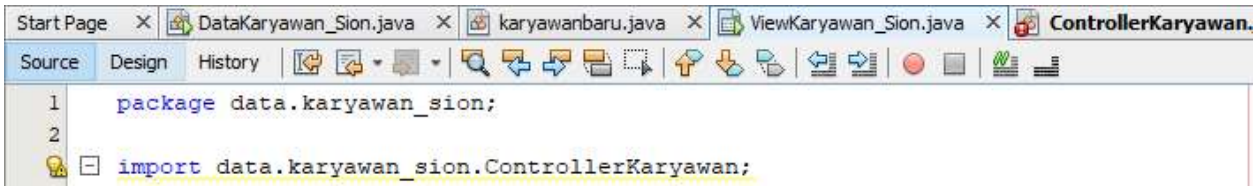
```
10      public ControllerKaryawan(){
11          ArrayData = new ArrayList<karyawanbaru>();
12      }
13      public void InsertData(String nama, String kodek, int umur){
14          karyawanbaru kryn = new karyawanbaru(nama, kodek, umur);
15          ArrayData.add(kryn);
16      }
```

Selanjutnya yaitu menkonstruksi Controller karyawan dengan konstruktor yang digunakan untuk membuat instance dari “ControllerKaryawan” dan menginisialisasi “ArrayData” sebagai ArrayList kosong untuk menyimpan data karyawan. Dilanjutkan dengan metode “InsertData” yang digunakan untuk menambahkan data karyawan baru ke dalam ArrayList “ArrayData”. Data karyawan baru

dibuat sebagai instance dari “Karyawanbaru” dengan parameter nama, kodeK, umur yang diterima dari pemanggilan metode ini, lalu ditambahkan ke ArrayList.

```
17      public DefaultTableModel showData(){
18          String[] kolom = { "Nama", "Kode Karyawan", "Ümur"};
19          Object[][] objData = new Object[ArrayData.size()][3];
20          int i = 0;
21
22          for(karyawanbaru n : ArrayData){
            String[] arrData = (n.getNama(), n.getKodek(), String.valueOf(n.umur));
24              objData[i] = arrData;
25              i++;
26          }
27
28          tablelist = new DefaultTableModel(objData, kolom){
29              public boolean inCellEditTable(int rowIndex, int colIndex){
30                  return false;
32          };
33
34          return tablelist;
35      }
36  }
```

Yang terakhir dari bagian controller karyawan yaitu menggunakan metode “showData” yang digunakan untuk membuat model data yang sesuai dengan “DefaultTableMode” yang akan digunakan pada JTable. Metode ini mendefinisikan array “Kolom” yang berisi nama – nama kolom yang akan ditampilkan di JTable, membuat array “objData” dengan ukuran yang sesuai dan mengisi data dari objek “ArrayData” ke dalam array, membuat model “tablelist” dengan “DefaultTableMode” menggunakan data dalam “objData” dan kolom yang telah didefinisikan, dan yang terakhir yaitu mengatur JTable agar tidak dapat diedit dengan meng-override metode “inCellEditTable” dan selalu mengembalikan “false”



Selanjutnya yaitu mengkonfigurasi source untuk ViewKaryawan yang dimulai dengan mengimport controller karyawan yang telah dibuat dengan perintah import data.karyawan_sion.ControllerKaryawan.

```
5   public class ViewKaryawan_Sion extends javax.swing.JFrame {
6       ControllerKaryawan kryn = new ControllerKaryawan();
7           public ViewKaryawan_Sion() {
8           initComponents();
9       }
```

Lalu membuat konstruktor “ViewKaryawan_Sion” yang akan digunakan untuk menginisialisasi frame beserta komponen Swing. Terdapat pemanggilan “initComponents()” yang dihasilkan oleh editor antarmuka pengguna (IDE) untuk merancang tampilan swing.

```
149     private void namaActionPerformed(java.awt.event.ActionEvent evt) {
150
151     }
152
153     private void umurActionPerformed(java.awt.event.ActionEvent evt) {
154         // TODO add your handling code here:
155     }
156
157     private void kodeActionPerformed(java.awt.event.ActionEvent evt) {
158         // TODO add your handling code here:
```

Lalu, untuk metode – metode pada gambar di atas yang ada dengan cara mengubah nama variable dari label untuk menangani peristiwa yang terjadi saat pengguna melakukan interaksi, sehingga tidak perlu dimasukkan input karena proses yang terjadi hanya akan di tombol button.

```
161     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
            String namaku = nama.getText();
            String kodeku = kode.getText();
            int umurku = Integer.parseInt(umur.getText());
165 
```

Yang terakhir yaitu mengkonfirgurasikan JButton dengan metode “JButton1ActionPerformed” yang dipanggil Ketika tombol/button di klik, Program akan mengambil teks dari tiga elemen JTextField: “nama”, “kode”, dan “umur”, lalu akan mengonversi nilai dari “umur” menjadi tipe data integer, melakukan panggilan metode “InsertData” dari objek “kryn” yang merupakan instance dari ControllerKaryawan untuk menyimpan data karyawan baru dengan parameter yang diambil dari elemen GUI, dan yang terakhir yaitu memperbarui model dari “tablelist” dengan data karyawan yang telah disimpan dengan memanggil “kryn.showData”.

```
17      public boolean updateData(int index, String nama, String kodek, int umur) {
18          if (index >= 0 && index < ArrayData.size()) {
19              karyawanbaru kryn = ArrayData.get(index);
20              kryn.setNama(nama);
21              kryn.setKodek(kodek);
22              kryn.setUmur(umur);
23              return true;
24          }
25          return false;
26      }
```

Pada pertemuan kali ini, untuk membuat tombol update dan delete langkah awalnya yaitu dengan menambahkan 2 jButton ke jframe yang telah dibuat, lalu mulai dengan memasukkan input untuk updateData dimana public Boolean updateData untuk memeriksa apakah indeks yang akan digunakan valid atau tidak. Maka akan masuk ke logika if yang digunakan untuk mengambil objek karyawan baru dari koleksi berdasarkan indeks dan setelah itu memperbarui file karyawanbaru dengan nilai baru dengan codingan di bawah ini.

```
22      public void setNama(String nama) {
23          this.nama = nama;
24      }
25 
26      public void setKodek(String kodek) {
27          this.kodek = kodek;
28      }
29 
30      public void setUmur(int umur) {
31          this.umur = umur;
32      }
```

Setelah diinput dan di set ke public void, maka langkah selanjutnya yaitu dengan mengembalikan true untuk menandakan bahwa pembaruan data telah berhasil dilakukan dan mengembalikan false jika indeks tidak valid.

```
28      public boolean deleteData(int index) {
29          if (index >= 0 && index < ArrayData.size()) {
30              ArrayData.remove(index);
31              return true;
32          }
33          return false;
34      }
```

Lalu, untuk metode delete data yang harus dilakukan yaitu melakukan hal yang sama seperti update dimana harus memeriksa indeks dan menghapus objek karyawanbaru dari koleksi berdasarkan indeks, setelah itu mengembalikan true bila penghapusan data telah berhasil dilakukan dan mengembalikan false jika indeks tidak valid. Kedua metode ini memeriksa apakah indeks yang diberikan berada dalam batas yang valid (antara 0 dan ArrayData.size() - 1). Jika indeks valid, mereka melakukan operasi yang sesuai: updateData memperbarui data pada indeks tersebut, sementara deleteData menghapus data pada indeks tersebut. Jika indeks tidak valid, keduanya mengembalikan false untuk menunjukkan bahwa operasi tidak dapat dilakukan.

```
192    private void updateActionPerformed(java.awt.event.ActionEvent evt) {
193        String namaku = nama.getText();
194        String kodeku = kode.getText();
195        int umurku = Integer.parseInt(s: umur.getText());
196
197       int selectedRow = tabel.getSelectedRow();
```

Lalu untuk ViewKaryawan disini, karena menambahkan button maka langkah awal untuk tombol update yaitu mengambil nilai dari komponen GUI dari komponen teks('nama', 'kode' dan 'umur'. Setelah selesai, maka deklarasikan int selectedRow agar mendapatkan baris yang terpilih dari tabel.

```
198        if (selectedRow >= 0) {
199            // Panggil fungsi updateData dari ControllerMotor
200            boolean updated = kryn.updateData(index: selectedRow, nama: namaku, kodek: kodeku, umur: umurku);
201
202            if (updated) {
203                // Data berhasil diperbarui, perbarui tampilan tabel
204                tabel.setModel(dataModel: kryn.showData());
205            } else {
206                // Data tidak dapat diperbarui (indeks salah), tampilkan pesan kesalahan jika diperlukan
207                JOptionPane.showMessageDialog(parentComponent:this, message:"Pilih baris yang ingin diperbarui.", title: "Kesalahan", messageType:JOptionPane.ERROR_MES
208            }
209        } else {
210            // Tidak ada baris yang dipilih, tampilkan pesan kesalahan jika diperlukan
211            JOptionPane.showMessageDialog(parentComponent:this, message:"Pilih baris yang ingin diperbarui.", title: "Kesalahan", messageType:JOptionPane.ERROR_MESSAGE
212        }
213    }
```

Untuk langkah terakhir update disini yaitu melakukan pembaruan data dari baris yang terpilih dari tabel tetapi menggunakan if-case untuk Memeriksa apakah ada baris yang dipilih (selectedRow >= 0). Jika ada baris yang dipilih, memanggil fungsi updateData dari objek kryn (mungkin objek yang bertanggung jawab atas pengelolaan data). Jika pembaruan berhasil, memperbarui model tabel (tabel.setModel(kryn.showData())) untuk merefresh tampilan data di GUI. Jika pembaruan gagal karena indeks yang tidak valid, menampilkan pesan kesalahan. JOptionPane untuk menampilkan pesan kesalahan atau sukses kepada pengguna. Selain itu, kode ini terlihat seperti bagian dari metode yang disebut updateActionPerformed, yang mungkin dihubungkan dengan suatu tombol atau elemen antarmuka pengguna lainnya yang menyebabkan aksi tersebut dipicu.

```
    private void deleteActionPerformed(java.awt.event.ActionEvent evt) {
229     int selectedRow = tabel.getSelectedRow();
230
231         if (selectedRow >= 0) {
232             // Get the index of the selected row in the JTable
233             int dataIndex = jTabel1.convertRowIndexToModel(viewRowIndex: selectedRow);
234
235             // Call the deleteData method to delete the data
236             boolean deleted = kryn.deleteData(index: dataIndex);
```

Untuk button delete, langkah awalnya sama dengan updateData yaitu dengan mengambil indeks baris yang terpilih dari tabel, selanjutnya yaitu mengonversi indeks baris tabel ke indeks model karena menggunakan convertRowIndexToModel untuk mendapatkan indeks model yang sesuai dengan baris yang terpilih di JTable. Hal ini penting jika tabel menggunakan sorter, sehingga Anda mendapatkan indeks model yang benar dan setelah itu memanggil metode deleteData dari objek kryn.

```
238             if (deleted) {
239                 // Refresh the JTable to reflect the updated data
240                 tabel.setModel(dataModel: kryn.showData());
241             } else {
242                 // Handle the case where data deletion failed
243                 JOptionPane.showMessageDialog(parentComponent:this, message:"Data deletion failed.");
244             }
245         } else {
246             // Handle the case where no row is selected
247             JOptionPane.showMessageDialog(parentComponent:this, message:"Please select a row to delete.");
248         }
249     }
```

Terakhir yaitu dengan menggunakan if-case untuk memperbarui tabel jika penghapusan berhasil. Jika penghapusan berhasil, memperbarui model tabel (tabel.setModel(kryn.showData())) untuk merefresh tampilan data di GUI. Tetapi penting untuk memeriksa apakah ada baris yang terpilih. Jika tidak, menampilkan pesan kesalahan.

```
package data.karyawan_sion.springHibernate;

import data.karyawan_sion.springHibernate.
import data.karyawan_sion.view.MahasiswaView;
import data.karyawan_sion.ApplicationContext;
import data.karyawan_sion.ClassPathXmlApplicationContext;

public class app {
    private static ApplicationContext applicationContext;

    public static void main(String[] args) {
        applicationContext = newClassPathXmlApplicationContext("classpath:spring-configuration.xml");
        new ViewKaryawan().setVisible(true);
    }

    public static KaryawanService getKaryawanService(){
        return(KaryawanService)applicationContext.getBean("KaryawanService");
    }
}
```

Objek applicationContext diinisialisasi dengan menggunakan kelas ClassPathXmlApplicationContext dan file konfigurasi Spring yang bernama "spring-configuration.xml" diambil dari classpath.
Kemudian, objek dari kelas ViewKaryawan dibuat dan diatur untuk ditampilkan. Metode ini digunakan untuk mendapatkan bean KaryawanService dari konteks aplikasi Spring. Mungkin KaryawanService adalah sebuah service atau komponen yang akan digunakan dalam aplikasi.

```
package data.karyawan_sion.springHibernate.dao;
import data.karyawan_sion.springHibernate.model.Karyawan;
import java.util.List;

public interface KaryawanDAO {
    public void save(Karyawan karyawan);
    public void update(Karyawan karyawan);
    public void delete(Karyawan karyawan);
    public Karyawan getKaryawan(String kodek);
    public List<Karyawan> getKaryawan();

}
```

Interface ini berada dalam package data.karyawan_sion.springHibernate.dao. Lalu mengimpor kelas Karyawan dan List yang digunakan dalam deklarasi metode interface. KaryawanDAO adalah nama interface yang mendefinisikan operasi-operasi terhadap entitas Karyawan. Metode ini bertujuan untuk menyimpan objek Karyawan ke sumber data. Metode ini digunakan untuk memperbarui data Karyawan yang sudah ada di sumber data. Metode ini bertujuan untuk menghapus data Karyawan dari sumber data. Metode pertama (getKaryawan(String kodek)) digunakan untuk mendapatkan objek Karyawan berdasarkan kodek (kemungkinan kode identifikasi unik).
Metode kedua (getKaryawan()) digunakan untuk mendapatkan daftar semua objek Karyawan dari sumber data.
Interface ini memberikan kerangka kerja umum untuk operasi-operasi dasar yang dapat dilakukan terhadap entitas Karyawan. Kelas yang mengimplementasikan interface ini akan memberikan implementasi konkret dari setiap metode sesuai dengan sumber data yang digunakan (mungkin menggunakan Hibernate, JDBC, atau sumber data lainnya).

```java
package data.karyawan_sion.springHibernate.dao;
import data.karyawan_sion.model.Mahasiswa;
import java.util.List;
import data.karyawan_sion.SessionFactory;
import data.karyawan_sion.beans.factory.annotation.Autowired;
import data.karyawan_sion.stereotype.Repository;

@Repository
public class KaryawanDAOmpl implements KaryawanDAO {
    @Autowired
    private SessionFactory sessionFactory;

    @Override
    public void save(Karyawan karyawan) {
        sessionFactory.getCurrentSession().save(karyawan);
    }

    @Override
    public void update(Karyawan karyawan) {
        sessionFactory.getCurrentSession().update(karyawan);
    }

    @Override
    public void delete(Karyawan karyawan) {
        sessionFactory.getCurrentSession().delete(mahasiswa);
    }

    @Override
    public Karyawan getKaryawan(String kodek) {
        return (Karyawan) sessionFactory.getCurrentSession().get(Karyawan.class, kodek);
    }
    @Override
    public List<Karyawan> getKaryawan() {
        return sessionFactory.getCurrentSession().createCriteria(Karyawan.class).list();
    }

}
```

Package dan import statement menyediakan akses ke kelas-kelas yang digunakan dalam implementasi. Anotasi "Repostory" ini menandai kelas ini sebagai komponen yang menyimpan, mengakses, dan mengelola data dari sumber data. Biasanya digunakan pada kelas DAO. Kelas ini mengimplementasikan interface KaryawanDAO. Anotasi @Autowired digunakan untuk melakukan injeksi dependensi ke dalam kelas. Di sini, SessionFactory diinjeksikan secara otomatis oleh Spring. SessionFactory adalah komponen yang menyediakan session Hibernate untuk interaksi dengan database. Metode ini digunakan untuk menyimpan objek Karyawan ke dalam database menggunakan session Hibernate. Metode ini digunakan untuk memperbarui data Karyawan yang sudah ada di dalam database menggunakan session Hibernate. Metode ini digunakan untuk menghapus data Karyawan dari dalam database menggunakan session Hibernate. Metode ini mengambil objek Karyawan berdasarkan kodek (kemungkinan kode identifikasi unik) dari database menggunakan session Hibernate. Metode ini mengembalikan daftar semua objek Karyawan dari database menggunakan session Hibernate.

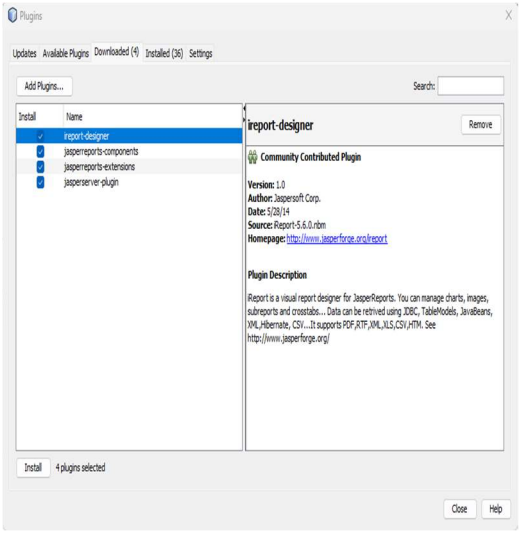
```java
package data.karyawan_sion.Service;
import data.karyawan_sion.springHibernate.model.Karyawan;
import java.util.List;

public interface KaryawanService {
    public void save(Karyawan karyawan);
    public void update(Karyawan karyawan);
    public void delete(Karyawan karyawan);
    public Karyawan getKaryawan(String kodek);
    public List<Karyawan> getKaryawan();
}
```

Package dan import statement menyediakan akses ke kelas-kelas yang digunakan dalam deklarasi interface. KaryawanService adalah nama interface yang mendefinisikan operasi-operasi terhadap entitas Karyawan. Metode ini mendeklarasikan operasi untuk menyimpan objek Karyawan ke dalam sumber data. Metode ini mendeklarasikan operasi untuk memperbarui data Karyawan yang sudah ada di sumber data. Metode ini mendeklarasikan operasi untuk menghapus data Karyawan dari sumber data. Metode pertama (getKaryawan(String kodek)) mendeklarasikan operasi untuk mendapatkan objek Karyawan berdasarkan kodek. Metode kedua (getKaryawan()) mendeklarasikan operasi untuk mendapatkan daftar semua objek Karyawan dari sumber data.
Interface ini memberikan kontrak umum untuk layanan yang berhubungan dengan entitas Karyawan. Implementasi dari interface ini kemungkinan akan menangani panggilan metode ini dengan melakukan komunikasi dengan lapisan akses data (DAO) yang sesuai, seperti kelas KaryawanDAOmpl.

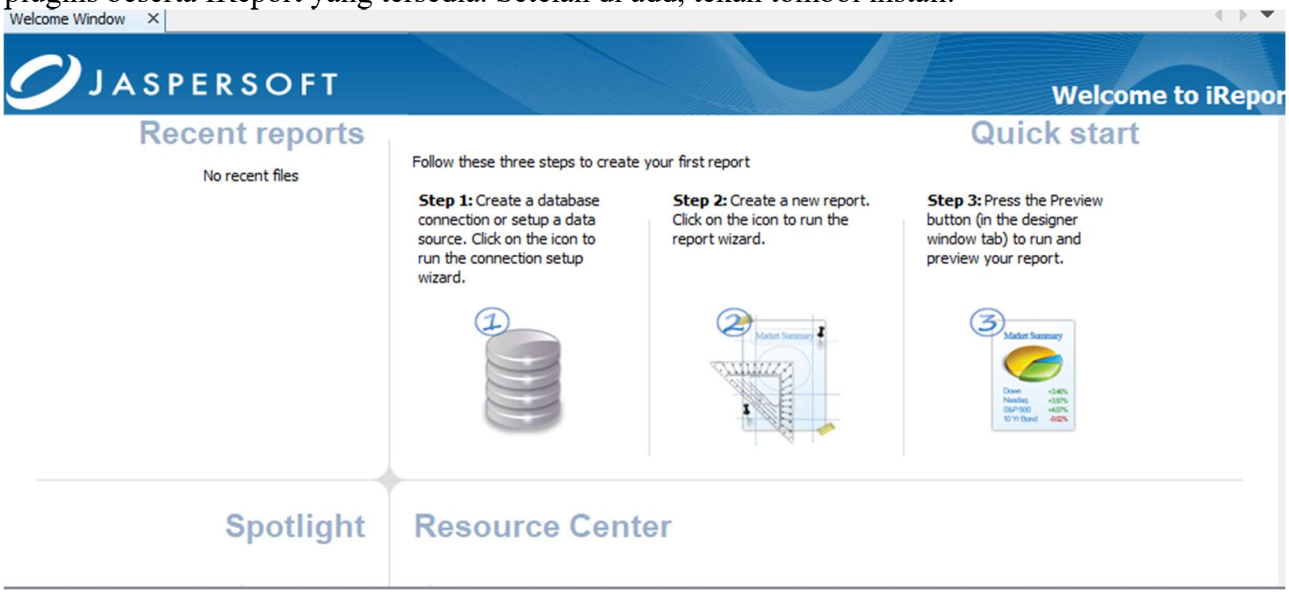
```java
package data.karyawan_sion.Service;
import data.karyawan_sion.dao.KaryawanDAO;
import data.karyawan_sion.springHibernate.model.Mahasiswa;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service("KaryawanService")
@Transactional(readOnly = true)

public class KaryawanServicempl implements KaryawanService {
    @Autowired
    private KaryawanDAO karyawanDao;

    @Transactional
    @Override
    public void save(Karyawan karyawan) {
        karyawanDao.save(karyawan);
    }

    @Transactional
    @Override
    public void update(Karyawan karyawan) {
        karyawanDao.update(mahasiswa);
    }

    @Transactional
    @Override
    public void delete(Karyawan karyawan) {
        karyawanDao.delete(karyawan);
    }

    @Override
    public Karyawan getKaryawan(String kodek) {
        return karyawanDao.getKaryawan(kodek);
    }

    @Override
    public List<Karyawan> getKaryawan() {
        return karyawanDao.getKaryawan();
    }
}
```

Package dan import statement menyediakan akses ke kelas-kelas yang digunakan dalam implementasi. Anotasi "@Service(KaryawanService)" ini menandai kelas ini sebagai layanan (service) Spring dengan nama "KaryawanService". Nama ini dapat digunakan untuk mengacu pada layanan ini saat melakukan injeksi dependensi. Anotasi "@Transactional(readOnly = true)" ini menunjukkan bahwa metode-metode dalam kelas ini akan dijalankan dalam transaksi. Untuk keperluan baca saja (readOnly), transaksi ini bersifat readonly. Anotasi "@Autowired" digunakan untuk melakukan injeksi dependensi ke dalam kelas, di sini, KaryawanDAO diinjeksikan secara otomatis oleh Spring.
KaryawanDAO adalah interface yang menyediakan operasi-operasi terhadap entitas Karyawan.

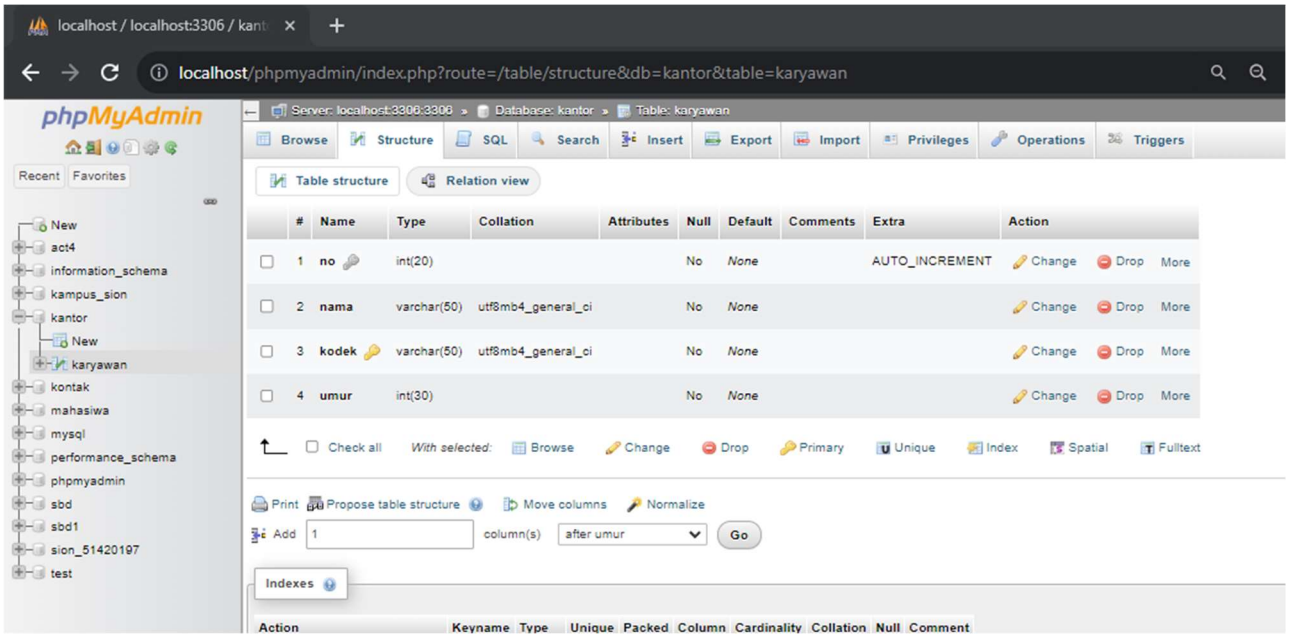
Metode ini awalnya diimplementasikan dengan komentar. Jika ingin mengaktifkan operasi penyimpanan, komentar ini dapat dihilangkan. Metode ini menggunakan objek dari KaryawanDAO untuk melakukan operasi pembaruan pada data Karyawan. Metode ini menggunakan objek dari KaryawanDAO untuk melakukan operasi penghapusan data Karyawan. Metode ini menggunakan objek dari KaryawanDAO untuk mendapatkan objek Karyawan berdasarkan kodek. Metode ini menggunakan objek dari KaryawanDAO untuk mendapatkan daftar semua objek Karyawan. Beberapa metode diannotasi dengan @Transactional untuk menandakan bahwa mereka berjalan dalam konteks transaksi.
Operasi-operasi yang melibatkan manipulasi data (seperti save, update, delete) biasanya memerlukan transaksi. Oleh karena itu, penggunaan anotasi @Transactional sangat penting dalam konteks layanan tersebut.



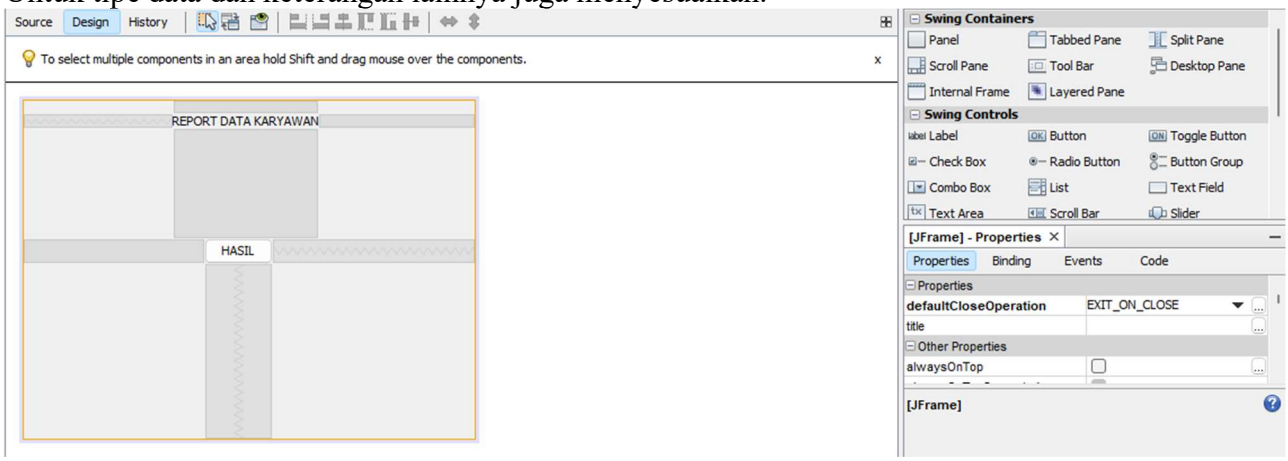
Langkah awal yang perlu kita lakukan untuk membuat Ireport dan juga mengimport jasper nya yaitu dengan menekan tombol plugins pada netbeans yang berada di menu netbeans. Setelah itu, pilih “Downloaded” dan tekan tombol “add plugins” maka akan memunculkan file exploler dan import plugins beserta IReport yang tersedia. Setelah di add, tekan tombol install.



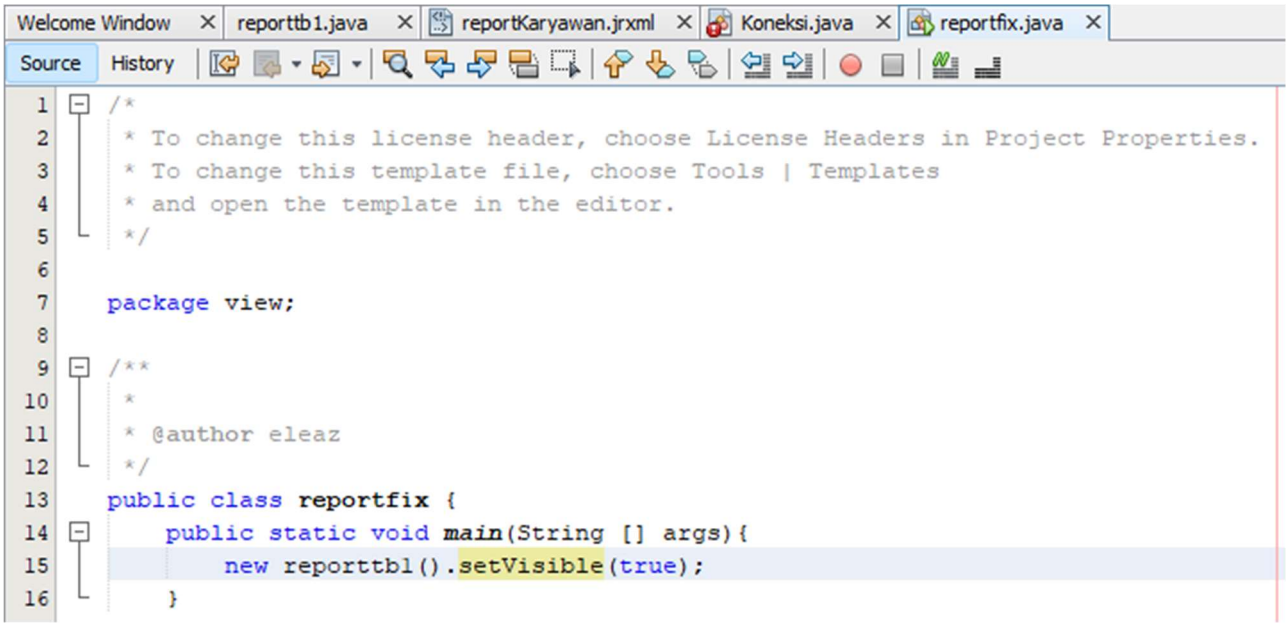
Makan akan muncul tampilan sebagai berikut, jika plugins yang kita import berhasil ditambahkan.



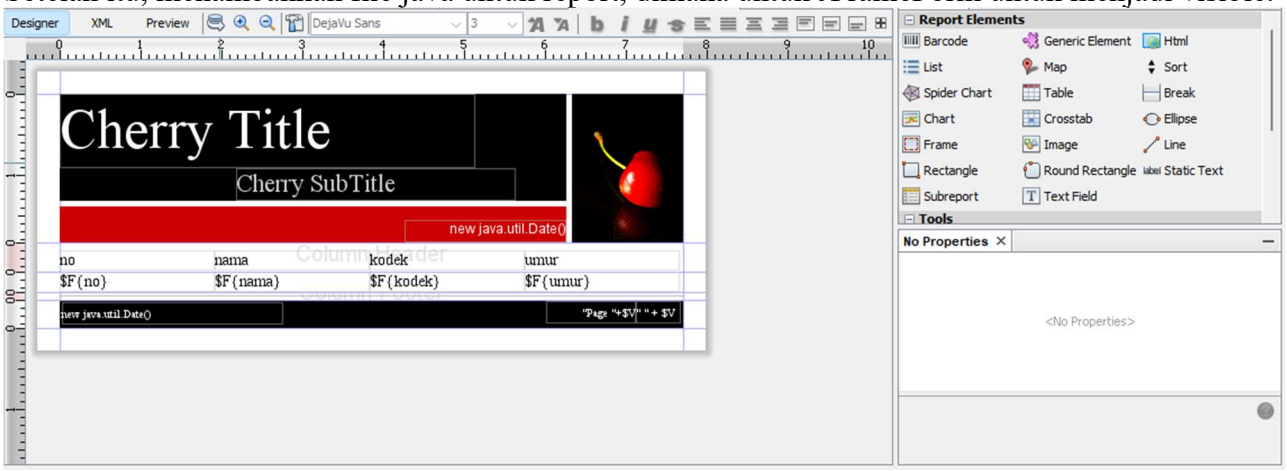
Lalu, masuk ke localhost/phpMyAdmin untuk membuat tabel database untuk Ireport yang ingin dibuat dengan ketentuan jumlah kolom pada tabel menyesuaikan dengan data yang ingin di input. Untuk tipe data dan keterangan lainnya juga menyesuaikan.



Setelah itu, membuat JFrameForm seperti gambar di atas yang hanya perlu menambahkan tombol "Hasil" dan Label1.



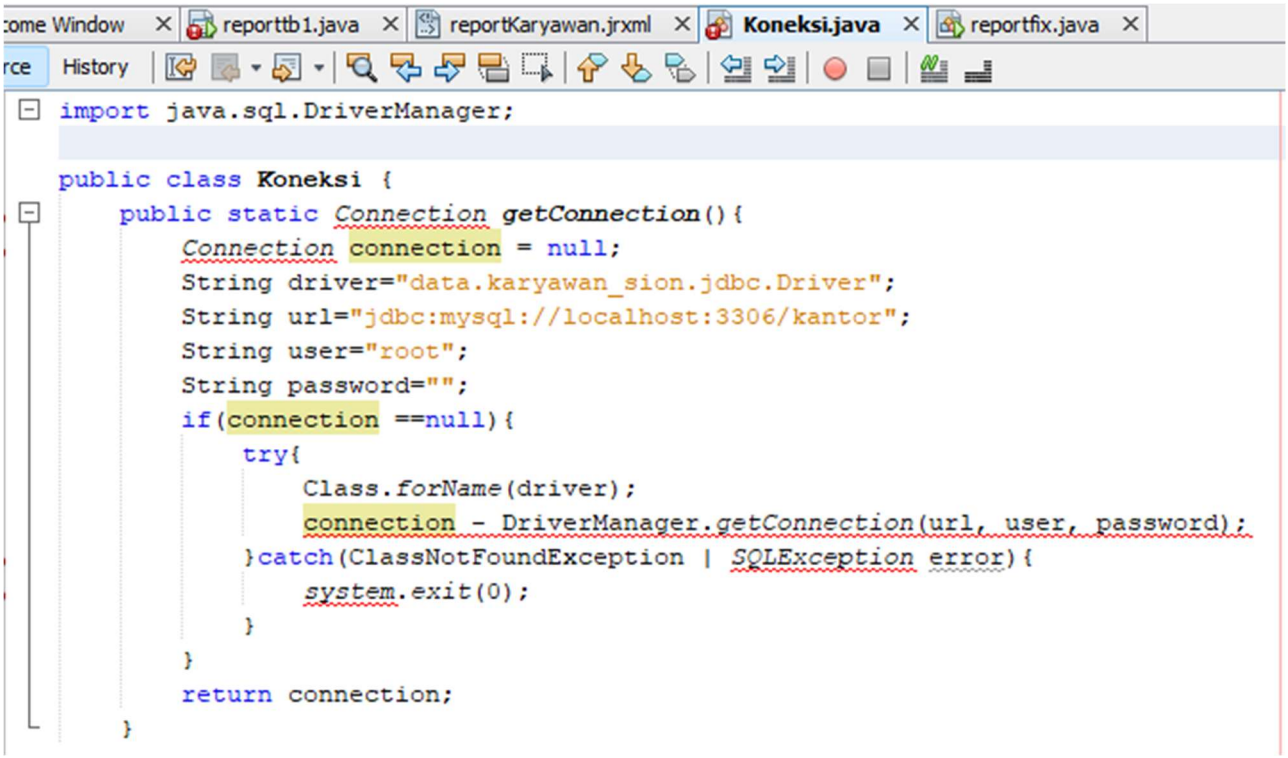
Setelah itu, menambahkan file java untuk report, dimana untuk JFrameForm untuk menjadi visible.



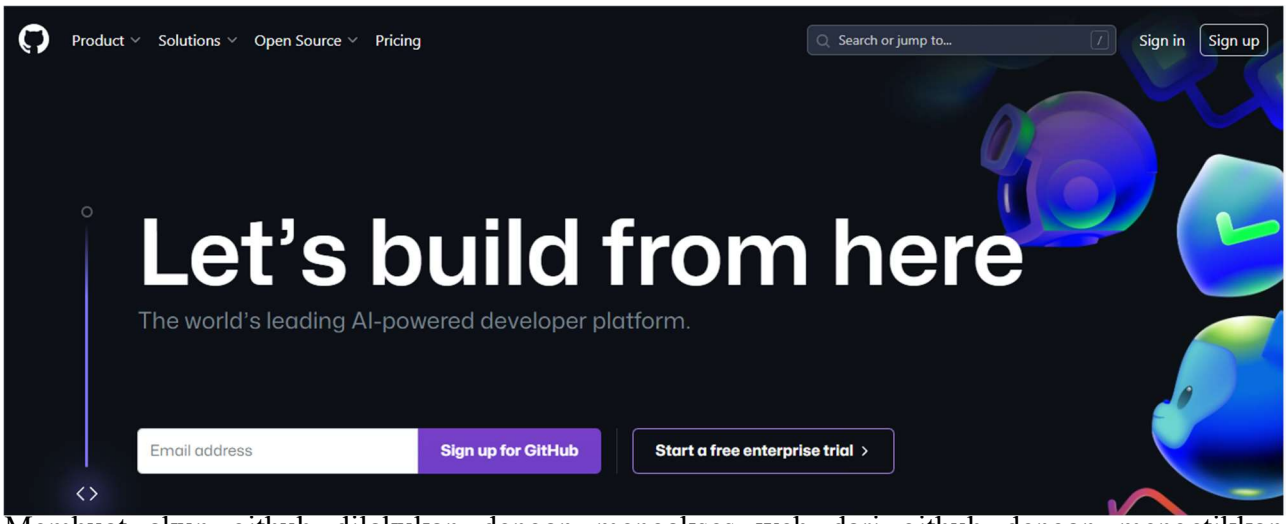
Jika Jasper telah berhasil diimport maka akan menampilkan tampilan berikut sebagai template dari IReport.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        JasperPrint jp = JasperFillManager.fillReport(getClass().getResourceAsStream("reportKaryawan.jasper"),null,Koneksi.getC
        JasperViewer.viewReport(jp,false);
    }catch(exception e){
        JOptionPane.showMessageDialog(rootpane, e);
    }
}
```

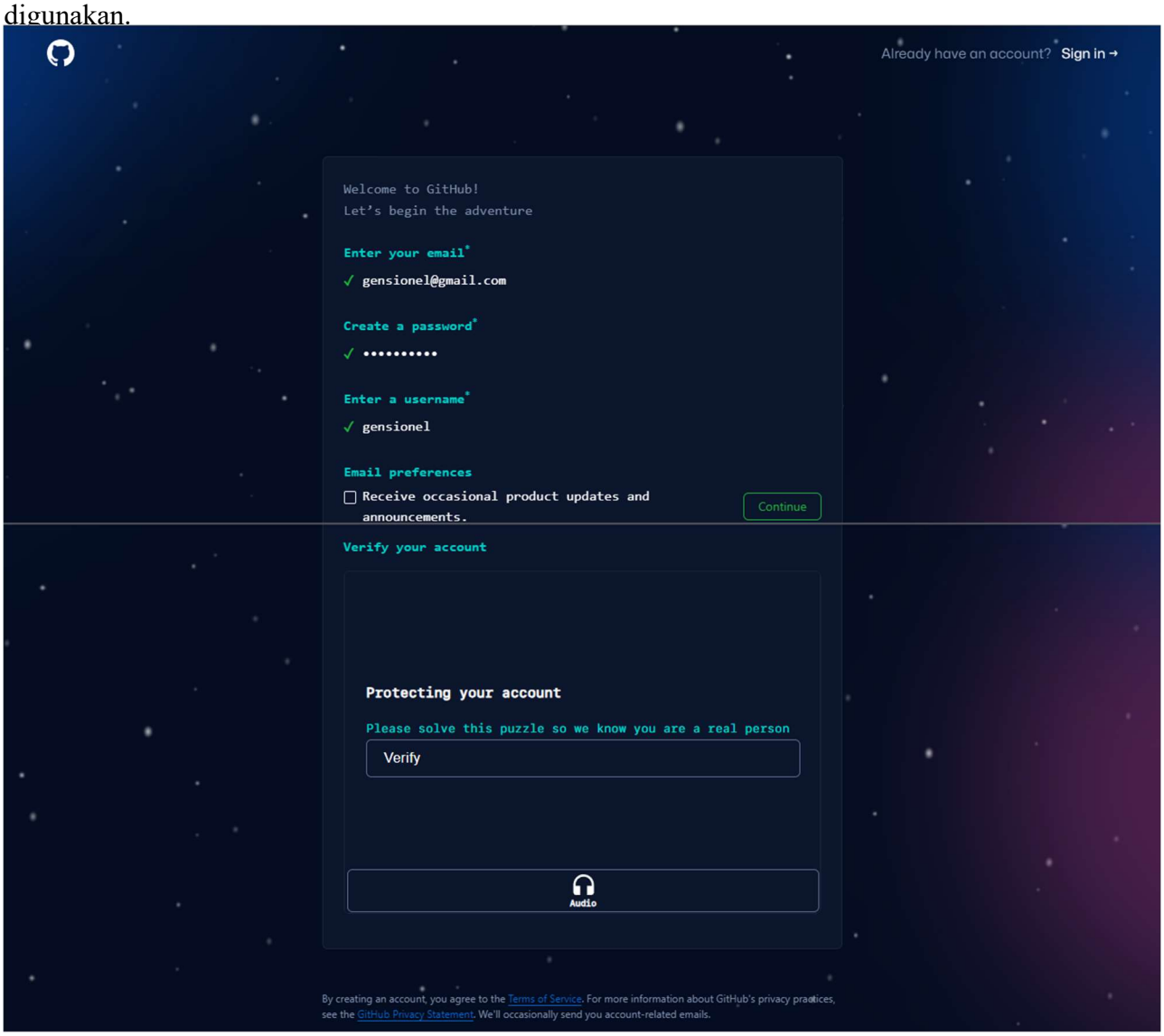
Untuk kodingan dari tombol hasil, kita masukkan kodingan di atas untuk tempat dimana input data Ireport dilakukan dan akan ditampilkan di Ireport.



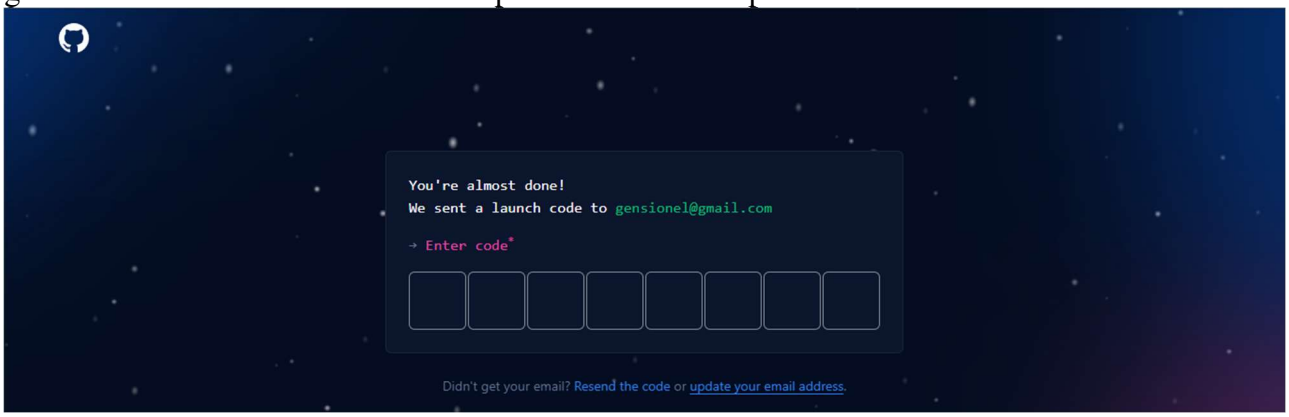
Setelah itu, membuat koneksi untuk tombol hasil dengan IReport, dimana untuk driver, url, user, dan password nya menyesuaikan dengan database yang telah dibuat. Dimana akan connect ke url, username, beserta password. Jika error maka program akan terhenti dan keluar.



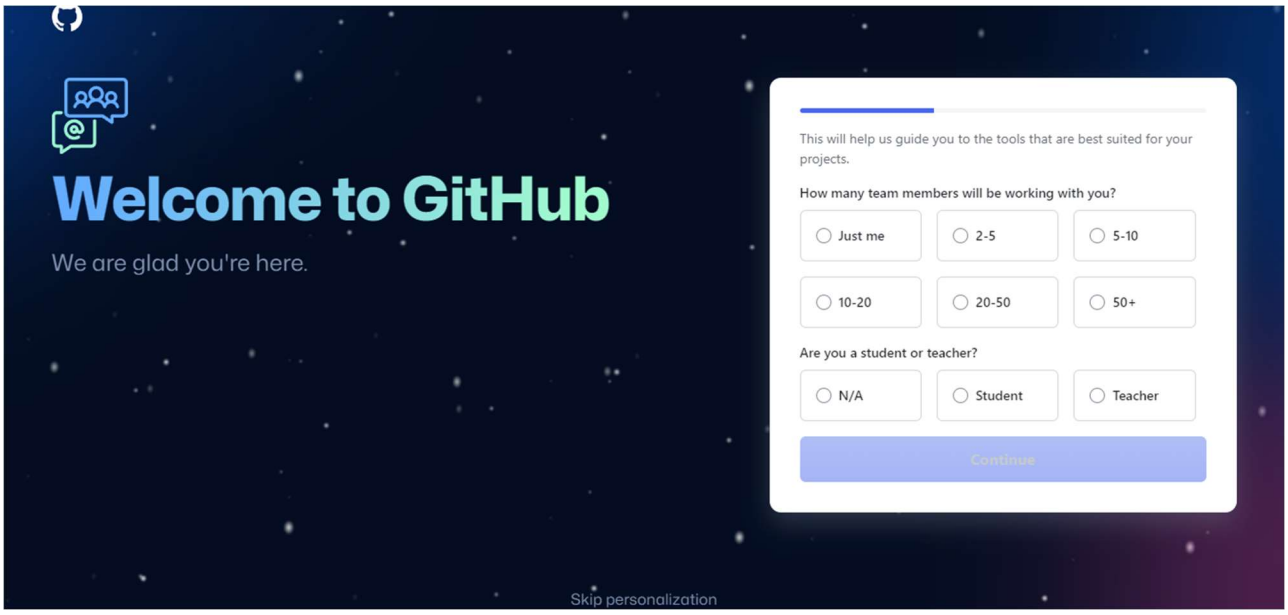
Membuat akun github dilakukan dengan mengakses web dari github dengan mengetikkan www.github.com dan setelah itu melakukan sign up dengan memasukkan Alamat email yang ingin digunakan.



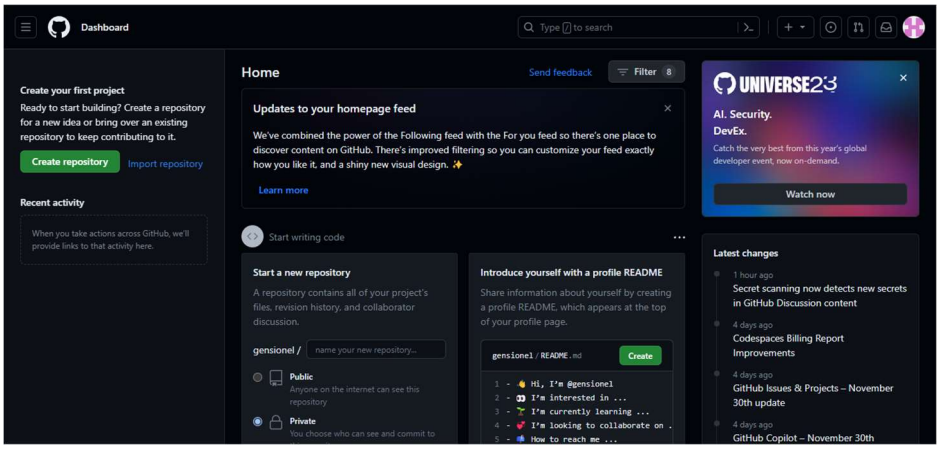
Setelah itu, mengisi data – data seperti gambar diatas secara bertahap, dimulai dengan memasukkan email untuk konfirmasi, membuat password yang sesuai dengan keinginan, memasukkan username yang ingin digunakan dan ada juga email preference untuk mengirimkan notifikasi ke email yang terdaftar informasi terbaru terkait update produk. Lalu akan ada verifikasi akun dengan menyocokkan gambar untuk membuktikan user atau pembuat akun merupakan manusia bukan robot.



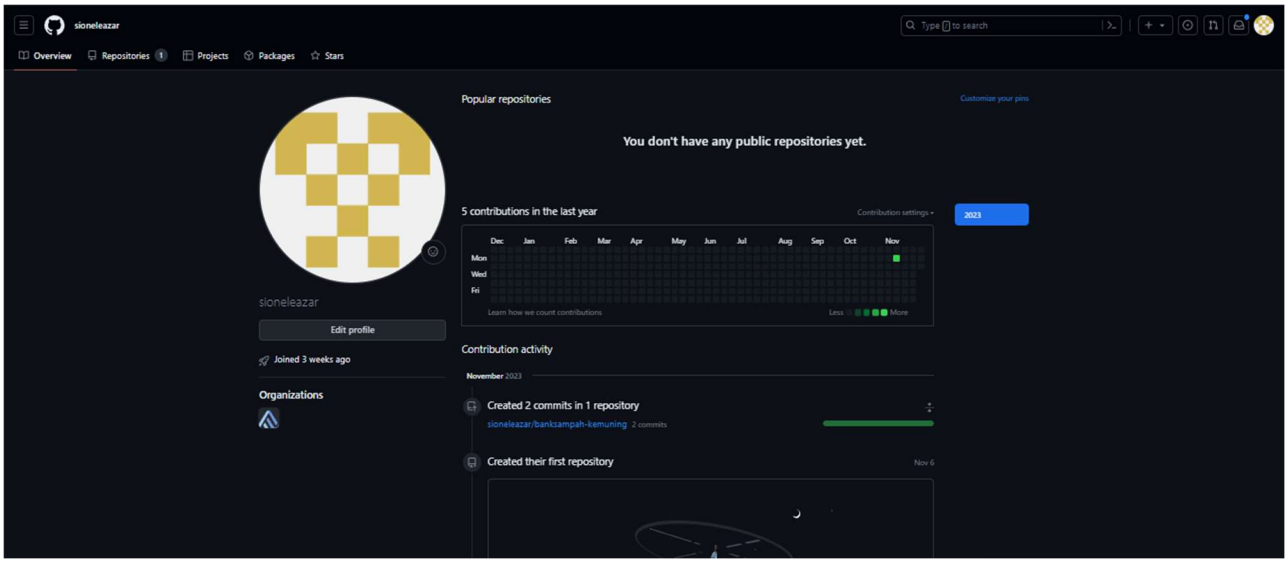
Lalu, memasukkan kode verifikasi ke Alamat email yang telah dimasukkan dan pastikkan agar kode tersebut tidak diketahui orang lain.



Jika sudah, maka bisa memasukki halaman awal untuk melakukan sedikit survey yang bersifat tidak wajib untuk data yang ingin di record oleh pihak github.



Maka user akan diarahkan ke dashboard setelah berhasil masuk.



Klik baigan overview untuk melihat profil dari user, agar bisa melihat apa saja yang telah dibuat oleh user.

```
MINGW64:/c/Users/eleaz

eleaz@Sion MINGW64 ~/OneDrive/Desktop
$ cd

eleaz@Sion MINGW64 ~
$ git config --global user.name "sioneleazar"

eleaz@Sion MINGW64 ~
$ git config --global user.email "sioneleazar@gmail.com"

eleaz@Sion MINGW64 ~
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=sioneleazar
user.email=sioneleazar@gmail.com

eleaz@Sion MINGW64 ~
$ git init
Initialized empty Git repository in C:/Users/eleaz/.git/

eleaz@Sion MINGW64 ~ (master)
$ git status
warning: could not open directory 'Application Data/': Permission denied
warning: could not open directory 'Cookies/': Permission denied
warning: could not open directory 'Documents/My Music/': Permission denied
warning: could not open directory 'Documents/My Pictures/': Permission denied
```

Melakukan pengecekkan untuk perintah apa saja yang ada di github dan melakukan configurasi antara folder yang ada pada PC/Laptop agar bisa terhubung dengan akun github.