

MANUAL BOOK WORKSHEET ABSENSI MAHASISWA
PROJECT LAB REKAYASA PERANGKAT LUNAK 2

Nama : Javier Nathaniel Damanik

NPM: 51420383

kelas: 4IA19



LABORATORIUM TEKNIK INFORMATIKA

UNIVERSITAS GUNADARMA

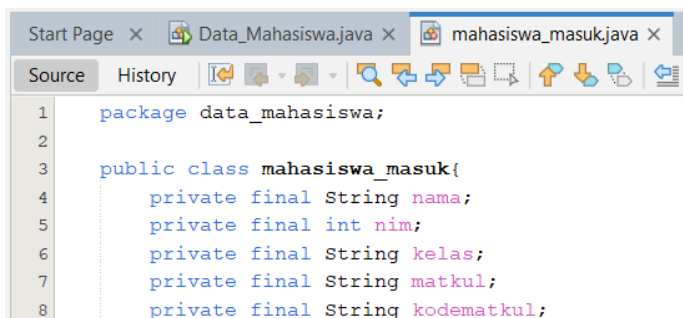
2023

MANUAL BOOK REKAYASA PERANGKAT LUNAK 2

Project pertama yaitu membuat worksheet absensi mahasiswa pada Data_Mahasiswa dan mahasiswa_masuk sebagai objek atau entitas dari mahasiswa yang berisikan Nama, nim, kelas dan kode matkul sebagai atributnya.

```
1 package data_mahasiswa;
2 import java.util.ArrayList;
3 import java.util.List;
4 import java.util.Scanner;
```

Package data_mahasiswa mengindikasikan bahwa kelas-kelas atau objek-objek seperti Data_Mahasiswa dan mahasiswa_masuk dibungkus dalam satu kelompok yakni data_mahasiswa. Import merupakan perintah java agar method-method yang ada didalam program dapat dijalankan seperti Util.ArrayList dan util.List yang merupakan kelas untuk mengoleksi data dalam bentuk daftar (list), juga Scanner yang merupakan kelas untuk input pada Java.



```
1 package data_mahasiswa;
2
3 public class mahasiswa_masuk{
4     private final String nama;
5     private final int nim;
6     private final String kelas;
7     private final String matkul;
8     private final String kodematkul;
```

Pada package yang sama, pada program/ kelas mahasiswa_masuk yang dapat diakses secara publik (bisa di dalam kelas mahasiswa_masuk bisa diluarnya seperti data_mahasiswa). Pada kelas ini terdapat atribut-atribut string nama, kelas, matkul, kodematkul dan integer nim dideklarasikan private atau hanya dapat diakses di dalam kelas itu untuk menjaga keamanan dan final yaitu nilai dari atribut tidak dapat diubah setelah diinisiasi di awal.

```
6 public class Data_Mahasiswa {
7     public static void main(String[] args) {
8         List<mahasiswa_masuk> daftarmahasiswa = new ArrayList<>();
9         Scanner input = new Scanner(System.in);
```

Pada kelas Data_Mahasiswa, terdapat method utama pada program java sehingga

program java dapat benar-benar berjalan dengan baik, dimana setelah program di run, output yang ditampilkan adalah argument yang berupa string atau teks. Kemudian, saya membuat objek daftarmahasiswa pada kelas List yang kan dikonstruksi di kelas ArrayList. Disini terdapat kelas Scanner dan objek input untuk membaca input pada method seperti nextInt() atau nextLine() atau lainnya dari keyboard.

```
10 public mahasiswa_masuk(String nama, int nim, String kelas, String matkul, String kodematkul) {  
11     this.nama = nama;  
12     this.nim = nim;  
13     this.kelas = kelas;  
14     this.matkul = matkul;  
15     this.kodematkul = kodematkul;  
16 }
```

Pada kelas mahasiswa_masuk dibuat konstruktor untuk menginisialisasi objek saat objek tersebut dibuat. Objek tersebut adalah nama, nim, kelas, matkul, kodematkul sebagai parameter yang perlu diberikan nilainya dari fungsi mahasiswa_masuk yang tidak mengembalikan nilai.

```
18 public String getNama() {  
19     return nama;  
20 }  
21  
22 public int getNim() {  
23     return nim;  
24 }  
25  
26 public String getKelas() {  
27     return kelas;  
28 }  
29  
30 public String getMatkul() {  
31     return matkul;  
32 }  
33  
34 public String getKodematkul() {  
35     return kodematkul;  
36 }
```

Pada kelas mahasiswa_masuk atribut-atribut private atau yang hanya dapat diakses di dalam kelas tersebut yakni nama, nim, kelas, matkul, dan kodematkul diambil nilainya dengan method getter agar dapat diakses di dalam kelas tersebut dan dikembalikan nilainya sesuai dengan objek tersebut.

```

11 while (true) {
12     System.out.println(x: "Tambah Data Mahasiswa Baru (y/n): ");
13     String jawaban = input.nextLine();
14
15     if (jawaban.equalsIgnoreCase(anotherString: "n")) {
16         break;
17     }

```

Pada kelas `Data_mahasiswa` dibuat perulangan `while (true)` yang menyatakan jalannya program untuk menambah data mahasiswa baru. Untuk mencetaknya, dideklarasikan variable `jawaban` yang akan disimpan sebagai tipe data `String` berdasarkan method `nextLine()` dengan menggunakan kelas `Scanner` dan objek `input`.

```

19 System.out.print (s: "Nama: ");
20 String nama = input.nextLine();
21
22 System.out.print (s: "NIM: ");
23 int nim = input.nextInt();
24
25 System.out.print (s: "Kelas: ");
26 String kelas = input.nextLine();
27 input.nextLine();
28
29 System.out.print (s: "Nama Mata Kuliah : ");
30 String matkul = input.nextLine();
31 input.nextLine();
32
33 System.out.print (s: "Kode Mata Kuliah : ");
34 String kodematkul = input.nextLine();
35 input.nextLine();
36
37 mahasiswa_masuk mahasiswa = new mahasiswa_masuk (nama, nim, kelas, matkul, kodematkul);
38 daftarmahasiswa.add (e: mahasiswa);
39
40 }

```

Selanjutnya yaitu mencetak pesan tanpa garis baru (“Nama”, “NIM”, “Kelas”, “Nama Mata Kuliah”, “Kode mata Kuliah”) yang berarti user harus menginputnya. Objek-objek tersebut (`nama`, `nim`, `kelas`, `matkul`, `kodematkul`) akan dibaca oleh `Scanner` dan akan disimpan berdasarkan tipe datanya.

```

38 @Override
39 public String toString() {
40     return "Nama: " + nama + ", Nim: " + nim + ", Kelas: " + kelas + ", Mata Kuliah: " + matkul + ", Kode Mata Kuliah: " + kodematkul;
41 }
42

```

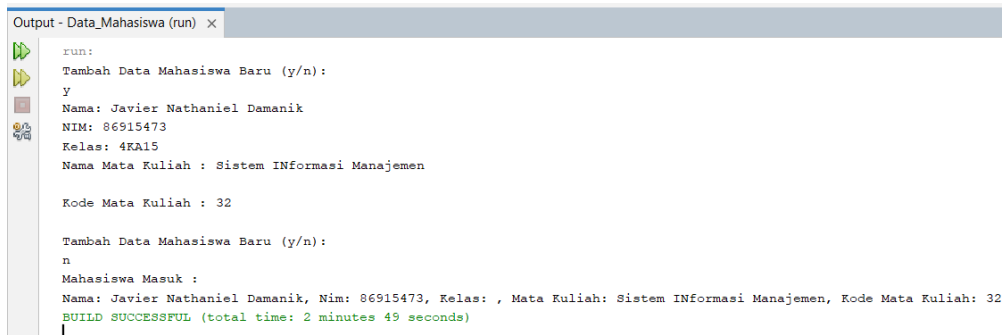
`@Override`: menerangkan bahwa metode `toString()` menggantikan metode dengan maksud yang sama seperti kelas yang diwarisi oleh `mahasiswa_masuk`. Kemudian nilai `String` method tersebut di kembalikan dengan nilai dari tulisan `String` atribut-atribut tersebut seperti “Nama” bergabung dengan nilai yang sudah diinput berdasarkan objek-objek tersebut yaitu contohnya objek `nama`.

```

41      System.out.println(x: "Mahasiswa Masuk :");
42      for (mahasiswa_masuk mahasiswa : daftarmahasiswa) {
43          System.out.println(x: mahasiswa);
44      }
45  }
46  }

```

Setelah mengisi semua nilai dari objek tersebut, maka akan ditampilkan output untuk mahasiswa masuk. Kemudian program mencetak nilai dari method toString() pada kelas mahasiswa_masuk.



```

Output - Data_Mahasiswa (run) x
run:
Tambah Data Mahasiswa Baru (y/n):
y
Nama: Javier Nathaniel Damanik
NIM: 86915473
Kelas: 4KA15
Nama Mata Kuliah : Sistem INformasi Manajemen

Kode Mata Kuliah : 32

Tambah Data Mahasiswa Baru (y/n):
n
Mahasiswa Masuk :
Nama: Javier Nathaniel Damanik, Nim: 86915473, Kelas: , Mata Kuliah: Sistem INformasi Manajemen, Kode Mata Kuliah: 32
BUILD SUCCESSFUL (total time: 2 minutes 49 seconds)

```

Maka seperti ini outputnya.

Program yang sebelumnya sudah dibuat, akan ditambah fitur MVC atau disebut Model-View-Controller untuk membuat Integrated Development Environment (IDE) untuk Java.

Untuk membuat tampilan front end Data masuk pada file View Mahasiswa.java, dibuat 6 buah label untuk Data Mahasiswa Masuk, Nama, Kelas, Nama Mata Kuliah, Kode Mata Kuliah; 5 buah text field untuk mengisi biodata tersebut, 1 button Create, dan 1 buah table dengan 5 buah atribut biodata tersebut seperti gambar di bawah ini.

Data Mahasiswa Masuk

Nama

NIM

Kelas

Nama Mata Kuliah

Kode mata Kuliah

Create

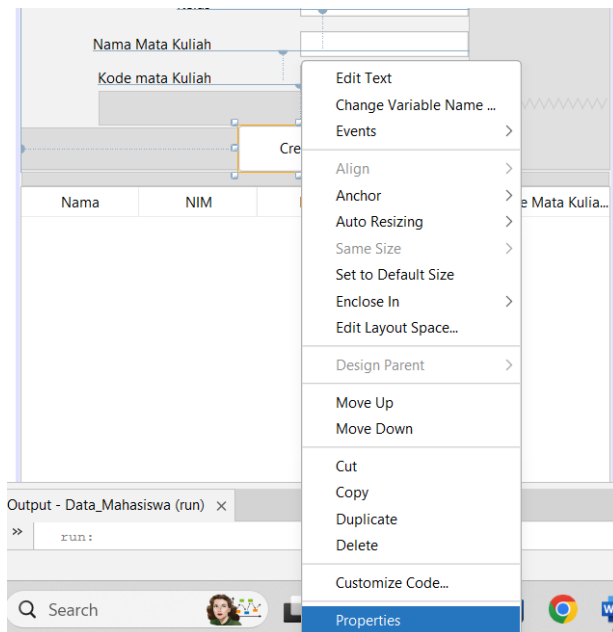
| Nama | NIM | Kelas | Nama Mata Kuli... | Kode Mata Kulia... |
|------|-----|-------|-------------------|--------------------|
| | | | | |

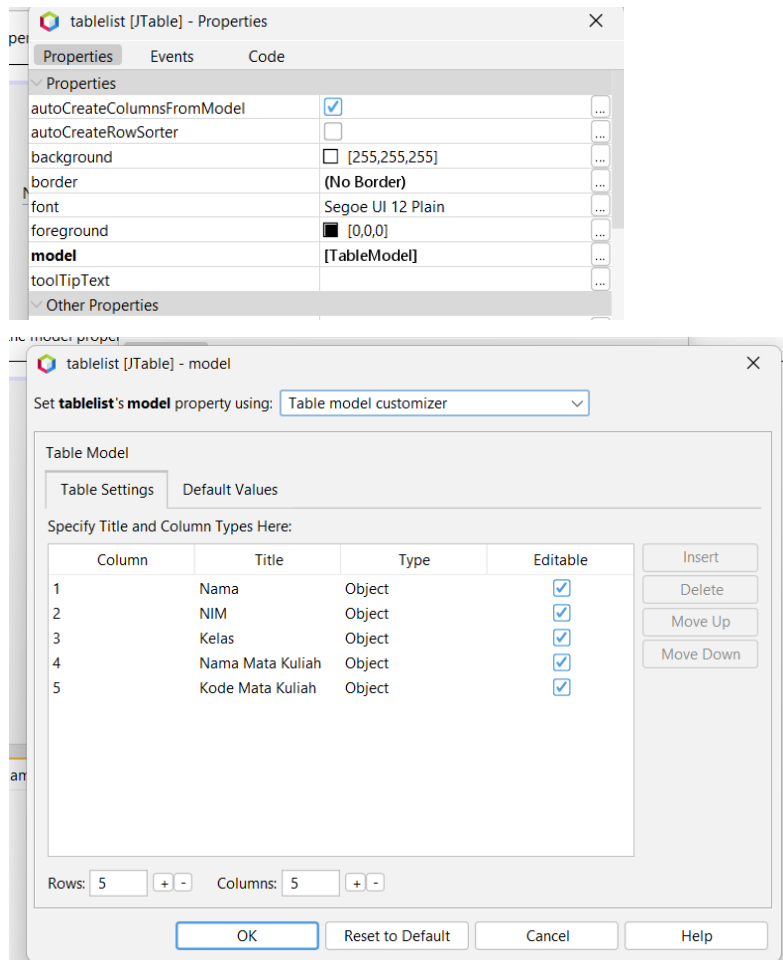
Selanjutnya yaitu membuat variable textfield, button, dan table tersebut dengan cara change variable name

Dengan ketentuan seperti berikut :

Untuk menambah field, klik kanan pada button create, lalu klik properties, lalu pada model klik tanda titik 3.

| COMPONENT | TEXT | NAME* |
|--------------------------|----------------------|------------|
| jLabel1 (Label) | Data Mahasiswa Masuk | - |
| jLabel2 (Label) | Nama | - |
| jLabel3 (Label) | NIM | - |
| jLabel4 (Label) | Kelas | - |
| jLabel5 (Label) | Nama Mata Kuliah | - |
| jLabel6 (Label) | Kode Mata Kuliah | - |
| jTextField1 (Text Field) | - | nama |
| jTextField2 (Text Field) | - | nim |
| jTextField3 (Text Field) | - | kelas |
| jTextField4 (Text Field) | - | matkul |
| jTextField5 (Text Field) | - | kodematkul |
| jButton1 (Button) | Create | buat |
| jTable1 (Table) | - | tablelist |





Kemudian untuk menambahkan field, tekan insert lalu posisinya bisa diubah pada move up atau move down. Jika ingin menghapus field, maka tekan delete.

```

1 package data_mahasiswa;
2
3 public class ViewMahasiswa extends javax.swing.JFrame {
4     ControllerMahasiswa mhs = new ControllerMahasiswa();
5     public ViewMahasiswa() {
6         initComponents();
7     }
8     @SuppressWarnings("unchecked")

```

Selanjutnya masuk pada program pertama yaitu View_Mahasiswa untuk membuat event pada button buat. Seperti pertemuan sebelumnya, masih pada satu direktori yaitu data_mahasiswa, pada kelas "ViewMahasiswa" yang merupakan turunan dari JFrame yang merupakan antarmuka pengguna (GUI) dan menggunakan komponen GUI Swing dari Java. Konstruktor kelas ViewMahasiswa yang memanggil metode initComponents() yang

mungkin menginisialisasi komponen-komponen GUI.

```
166 private void buatActionPerformed(java.awt.event.ActionEvent evt) {  
167     String namaku = nama.getText();  
168     int nimku = Integer.parseInt(s: nim.getText());  
169     String kelasku = kelas.getText();  
170     String matkulku = matkul.getText();  
171     String kodematkulku = kodematkul.getText();  
172     mhs.insertData(namaku, nimku, kelasku, matkulku, kodematkulku);  
173     tablelist.setModel(dataModel:mhs.showData());  
174 }
```

Pada program method ini, kelas ViewMahasiswa berperan sebagai antarmuka pengguna (GUI) yang berinteraksi dengan logika kontrol yang terdapat dalam ControllerMahasiswa untuk mengelola data mahasiswa. Program mengambil nilai dari beberapa komponen atribut GUI seperti nama, nim, kelas, matkul, dan kodematkul.

```
1 package data_mahasiswa;  
2 import java.util.ArrayList;  
3 import javax.swing.table.DefaultTableModel;
```

Selanjutnya masuk pada program kedua yaitu ControllerMahasiswa untuk sistem manajemen data mahasiswa pada GUI. Disediakan paket kelas ArrayList untuk struktur data dinamis berupa array dan DefaultTableModel untuk menyimpan dan mengelola data yang akan ditampilkan pada tabel; seperti menambahkan, menghapus, dan mengakses data secara dinamis.

```
5 public class ControllerMahasiswa {  
6     ArrayList<mahasiswa_masuk> ArrayData;  
7     DefaultTableModel tablelist;  
8  
9     public ControllerMahasiswa() {  
10         ArrayData = new ArrayList<>();  
11     }  
12 }
```

ArrayList<mahasiswa_masuk>: yaitu instance yang berasal dari kelas ArrayList yang akan menyimpan objek-objek dalam bentuk struktur data dinamis dari kelas mahasiswa_masuk. Untuk mengatur data mahasiswa yang akan ditampilkan pada tabel diperlukan instance DefaultTableModel, yang setelahnya akan dikelola format tampilannya pada kelas Controller Mahasiswa. Kemudian dideklarasikan variable ArrayData

```
13 public void InsertData(String nama, int nim, String kelas, String matkul, String kodematkul){  
14     mahasiswa_masuk mhs = new mahasiswa_masuk(nama, nim, kelas, matkul, kodematkul);  
15     ArrayData.add(e: mhs);  
16 }
```

Untuk menambah objek atau data pada mahasiswa baru ke dalam ArrayData. Method ini mengambil beberapa parameter, yaitu nama, nim, kelas, matkul, dan kodematkul berasal dari variable pada textfield yang akan diisi pada output nantinya. Kemudian objek baru dari kelas mahasiswa_masuk diasumsikan sebagai konstruktor berupa nilai-nilai yang diterima sebagai argument berisi informasi mahasiswa pada output. Objek tersebut (mhs) kemudian ditambahkan ke dalam ArrayData menggunakan metode add dari kelas ArrayList sebagai kumpulan objek mahasiswa_masuk.

```
18 public DefaultTableModel showData() {
19     String[] kolom = { "Nama", "Nim", "Kelas", "Mata Kuliah", "Kode Mata Kuliah"};
20     Object[][] objData = new Object[ArrayData.size()][5];
21     int i = 0;
```

Pada kelas DefaultTableModel akan menampilkan data mahasiswa pada antarmuka pengguna. Sebelum dilakukan perulangan for, dideklarasikan array berisi string atau karakter Nama, Nim, Kelas, Mata Kuliah, Kode Mata Kuliah.

```
23 for (mahasiswa_masuk n : ArrayData) {
24     String[] arrData = {n.getNama(), String.valueOf(n.nim), n.getKelas(), n.getMatkul(), n.getKodematkul()};
25     objData[i] = arrData;
26     i++;
27 }
```

Disini dilakukan perulangan for-each yang digunakan untuk memproses setiap nilai input objek mahasiswa_masuk dalam ArrayData.

```
29 tablelist = new DefaultTableModel(data: objData, columnNames: kolom) {
30     public boolean inCellEditable(int rowIndex, int colIndex) {
31         return false;
32     }
33 };
34
35 return tablelist;
36 }
```

Selanjutnya pada variable tablelist untuk tabel, data akan diatur dan ditampilkan pada tabel di antarmuka pengguna dengan parameter array dua dimensi objData dan array 1 dimensi kolom sebelumnya.

| COMPONENT | TEXT | NAME* |
|-----------|--------|--------|
| jButton2 | Clear | clear |
| jButton3 | Delete | delete |

Seperti inilah output dari program MVC tersebut.

Selanjutnya, menambahkan tombol Clear dan Update pada form ViewMahasiswa.java.
nama variable button tersebut adalah clear dan delete.

```

202
203 private void clearActionPerformed(java.awt.event.ActionEvent evt) {
204     nama.setText("");
205     nim.setText("");
206     kelas.setText("");
207     matkul.setText("");
208     kodematkul.setText("");
209 }

```

Untuk mengosongkan isian dari beberapa elemen antarmuka pengguna, maka akan dipanggil method `clearActionPerformed` sesuai variable `clear` pada button `Clear`. `nama.setText("");`, `nim.setText("");`, dll.: merupakan baris-baris kode yang mengatur teks dari komponen antarmuka pengguna (disebut `nama`, `nim`, `kelas`, `matkul`, `kodematkul`) atau isian teks di dalam komponen-komponen tersebut akan dihapus, sehingga memberikan efek "membersihkan" atau "mengosongkan" formulir atau input. menutup blok kode dari metode `clearActionPerformed`.

```

212 private void deleteActionPerformed(java.awt.event.ActionEvent evt) {
213     int selectedRow = tablelist.getSelectedRow();
214     if (selectedRow != -1){
215         DefaultTableModel model = (DefaultTableModel) tablelist.getModel();
216         model.removeRow(row: selectedRow);
217     } else {
218         JOptionPane.showMessageDialog(parentComponent: this, message: "Pilih baris yang ingin dihapus.",
219                                     title: "Error", messageType: JOptionPane.ERROR_MESSAGE);
220     }
221 }

```

Untuk menghapus baris yang dipilih pada tabel, berdasarkan button `Delete` dengan variable `delete`, maka akan dipanggil method `deleteActionPerformed`. Jika tidak ada baris yang dipilih, pengguna akan diberi tahu melalui pesan kesalahan. Baris pertama mengambil indeks baris yang dipilih di tabel (`tablelist`) dan mengembalikan -1 jika tidak ada baris yang dipilih. model tabel (`DefaultTableModel`) dari tabel (`tablelist`) digunakan untuk mengakses dan mengelola data dalam tabel.

Data Mahasiswa Masuk

Nama:

NIM:

Kelas:

Nama Mata Kuliah:

Kode mata Kuliah:

| Nama | Nim | Kelas | Mata Kuliah | Kode Mata Kuliah |
|--------------------------|----------|-------|----------------------------|------------------|
| Javier Nathaniel Damanik | 86915473 | 4KA15 | Sistem Informasi Manajemen | 32 |

Data Mahasiswa Masuk

Nama:

NIM:

Kelas:

Nama Mata Kuliah:

Kode mata Kuliah:

| Nama | Nim | Kelas | Mata Kuliah | Kode Mata Kuliah |
|--------------------------|----------|-------|----------------------------|------------------|
| Javier Nathaniel Damanik | 86915473 | 4KA15 | Sistem Informasi Manajemen | 32 |

Seperti inilah tampilan output dari program MVC yang sudah ditambah fitur Create, Delete, Clear tersebut.

Program selanjutnya adalah Spring dan Hibernate yaitu framework untuk menyederhanakan pengembangan aplikasi Java Enterprise dan pemetaan objek-relasional (ORM) pada interaksi antara aplikasi Java dan database relasional.

MahasiswaTableModel.java

```

1 package com.labti.springHibernate.configuration;
2
3 import com.labti.springHibernate.model.Mahasiswa;
4 import java.util.ArrayList;
5 import java.util.List;
6 import javax.swing.table.AbstractTableModel;
7
8 public class MahasiswaTableModel extends AbstractTableModel {
9     private List<Mahasiswa> mahasiswas = new ArrayList<>();
10    private final String HEADER[] = {"Npm", "Nama", "Kelas", "Alamat"};
11
12    public MahasiswaTableModel(List<Mahasiswa> mahasiswas) {
13        this.mahasiswas = mahasiswas;
14    }
15
16    @Override
17    public int getRowCount() {
18        return mahasiswas.size();
19    }
20
21    @Override
22    public int getColumnCount() {
23        return HEADER.length;
24    }
25
26    @Override
27    public String getColumnName(int columnIndex) {
28        return HEADER[columnIndex];
29    }

```

```

31    @Override
32    public Object getValueAt(int rowIndex, int columnIndex) {
33        Mahasiswa mahasiswa = mahasiswas.get(rowIndex);
34        switch (columnIndex) {
35            case 0:
36                return mahasiswa.getNpm();
37            case 1:
38                return mahasiswa.getNama();
39            case 2:
40                return mahasiswa.getKelas();
41            case 3:
42                return mahasiswa.getAlamat();
43            default:
44                return null;
45        }
46    }
47 }
48

```

Fungsi utama dari program tersebut adalah menyediakan cara untuk merepresentasikan data Mahasiswa dalam bentuk tabel, yang nantinya dapat ditampilkan di antarmuka pengguna grafis (GUI).

MahasiswaController.java

```

1 package com.labti.springHibernate.controller;
2
3 import com.labti.springHibernate.app;
4 import com.labti.springHibernate.config.MahasiswaTableModel;
5 import com.labti.springHibernate.model.Mahasiswa;
6 import com.labti.springHibernate.view.MahasiswaView;
7 import com.labti.springHibernate.service.MahasiswaService;
8 import java.util.List;
9 import javax.swing.JOptionPane;
10
11 public class MahasiswaController {
12     private final MahasiswaView mahasiswaView;
13     private MahasiswaTableModel mahasiswaTableModel;
14     private List<Mahasiswa> mahasiswas;
15
16     public MahasiswaController(MahasiswaView mahasiswaView) {
17         this.mahasiswaView = mahasiswaView;
18     }
19
20     public void tampilData() {
21         mahasiswas = app.getMahasiswaService().getMahasiswas();
22         mahasiswaTableModel = new MahasiswaTableModel(mahasiswas);
23         this.mahasiswaView.getTable().setModel(mahasiswaTableModel);
24     }
25
26     public void show() {
27         int index = this.mahasiswaView.getTable().getSelectedRow();
28         this.mahasiswaView.getNpm().setText(String.valueOf(
29             this.mahasiswaView.getTable().getValueAt(index, 0)));
30         this.mahasiswaView.getNama().setText(String.valueOf(
31             this.mahasiswaView.getTable().getValueAt(index, 1)));
32         this.mahasiswaView.getKelas().setText(String.valueOf(
33             this.mahasiswaView.getTable().getValueAt(index, 2)));
34         this.mahasiswaView.getAlamat().setText(String.valueOf(
35             this.mahasiswaView.getTable().getValueAt(index, 3)));
36     }
37
38     public void clear() {
39         this.mahasiswaView.getNpm().setText("");
40         this.mahasiswaView.getNama().setText("");
41         this.mahasiswaView.getKelas().setText("");
42         this.mahasiswaView.getAlamat().setText("");
43     }
44
45     public void saveMahasiswa() {
46         Mahasiswa mahasiswa = new Mahasiswa();
47         mahasiswa.setNpm(this.mahasiswaView.getNpm().getText());
48         mahasiswa.setNama(this.mahasiswaView.getNama().getText());
49         mahasiswa.setKelas(this.mahasiswaView.getKelas().getText());
50         mahasiswa.setAlamat(this.mahasiswaView.getAlamat().getText());
51         app.getMahasiswaService().save(mahasiswa);
52         JOptionPane.showMessageDialog(null, "Data Berhasil di Simpan", "info",
53             JOptionPane.INFORMATION_MESSAGE);
54         clear();
55         tampilData();
56     }
57
58     public void updateMahasiswa() {
59         Mahasiswa mahasiswa = new Mahasiswa();
60         mahasiswa.setNpm(this.mahasiswaView.getNpm().getText());
61         mahasiswa.setNama(this.mahasiswaView.getNama().getText());
62         mahasiswa.setKelas(this.mahasiswaView.getKelas().getText());
63         mahasiswa.setAlamat(this.mahasiswaView.getAlamat().getText());
64         app.getMahasiswaService().update(mahasiswa);
65         JOptionPane.showMessageDialog(null, "Data Berhasil di Edit", "info",
66             JOptionPane.INFORMATION_MESSAGE);
67         clear();
68         tampilData();
69     }

```

```

70
71 public void deleteMahasiswa() {
72     if(this.mahasiswaView.getNpm().getText() == null){
73         JOptionPane.showMessageDialog(null, "Mahasiswa belum dipilih", "error",JOptionPane.ERROR_MESSAGE);
74     }else{
75         Mahasiswa mahasiswa = new Mahasiswa();
76         mahasiswa.setNpm(this.mahasiswaView.getNpm().getText());
77         int pilih = JOptionPane.showConfirmDialog(null,"Apakah data ingin dihapus ?", "Warning",JOptionPane.YES_NO_OPTION, JOptionPane.WARNING_MESSAGE);
78         if(pilih == JOptionPane.YES_OPTION){
79             app.getMahasiswaService().delete(mahasiswa);
80             JOptionPane.showMessageDialog(null, "Data Berhasil di Hapus", "info",JOptionPane.INFORMATION_MESSAGE);
81             clear();
82             tampilData();
83         }
84     }
85 }

```

Program ini merupakan kelas pengontrol (controller) dalam aplikasi Java Swing yang mengelola interaksi antara antarmuka pengguna (GUI) dan layanan yang berhubungan dengan entitas Mahasiswa.

MahasiswaDAO.java

```

1 package com.labti.springHibernate.dao;
2
3 import com.labti.springHibernate.model.Mahasiswa;
4 import java.util.List;
5
6 public interface MahasiswaDAO {
7     public void save(Mahasiswa mahasiswa);
8     public void update(Mahasiswa mahasiswa);
9     public void delete(Mahasiswa mahasiswa);
10    public Mahasiswa getMahasiswa(String npm);
11    public List<Mahasiswa> getMahasiswas();
12
13 }

```

Ini adalah antarmuka (interface) yang mendefinisikan kontrak atau spesifikasi untuk operasi dasar terkait entitas Mahasiswa dalam konteks penyimpanan data.

MahasiswaDAOImpl.java

```

1 package com.labti.springHibernate.dao;
2
3 import com.labti.springHibernate.model.Mahasiswa;
4 import java.util.List;
5 import org.hibernate.SessionFactory;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Repository;
8
9
10 @Repository
11 public class MahasiswaDAOImpl implements MahasiswaDAO {
12     @Autowired
13     private SessionFactory sessionFactory;
14
15     @Override
16     public void save(Mahasiswa mahasiswa) {
17         sessionFactory.getCurrentSession().save(mahasiswa);
18     }
19
20     @Override
21     public void update(Mahasiswa mahasiswa) {
22         sessionFactory.getCurrentSession().update(mahasiswa);
23     }
24
25     @Override
26     public void delete(Mahasiswa mahasiswa) {
27         sessionFactory.getCurrentSession().delete(mahasiswa);
28     }
29 }

```



```

29
30     @Override
31     public Mahasiswa getMahasiswa(String npm) {
32         return (Mahasiswa) sessionFactory.getCurrentSession().get(Mahasiswa.class, npm);
33     }

34
35     @Override
36     public List<Mahasiswa> getMahasiswas() {
37         return sessionFactory.getCurrentSession().createCriteria(Mahasiswa.class).list();
38     }
39
40 }

```

Kelas ini bertanggung jawab untuk mengimplementasikan operasi-operasi dasar terkait data Mahasiswa menggunakan Hibernate, sebuah kerangka kerja pemetaan objek-relasional (ORM) untuk Java

Mahasiswa.java

```

1 package com.labti.springHibernate.model;
2
3 import java.io.Serializable;
4 import javax.persistence.Column;
5 import javax.persistence.Entity;
6 import javax.persistence.Id;
7 import javax.persistence.Table;
8
9 @Entity
10 @Table(name = "tb_mahasiswa")
11
12 public class Mahasiswa {
13     @Id
14     @Column(name = "npm", length = 8)
15     private String npm;
16
17     @Column(name = "nama", length = 50)
18     private String nama;
19
20     @Column(name = "kelas", length = 10)
21     private String kelas;
22
23     @Column(name = "alamat", length = 150)
24     private String alamat;
25
26     public String getNpm() {
27         return npm;
28     }
29     public void setNpm(String npm) {
30         this.npm = npm;
31     }
32
33     public String getNama() {
34         return nama;
35     }
36
37     public void setNama(String nama) {
38         this.nama = nama;
39     }
40     public String getKelas() {
41         return kelas;
42     }
43     public void setKelas(String kelas) {
44         this.kelas = kelas;
45     }
46
47     public String getAlamat() {
48         return alamat;
49     }
50     public void setAlamat(String alamat) {
51         this.alamat = alamat;
52     }
53
54 }

```

Ini adalah kelas model yang merepresentasikan entitas Mahasiswa dalam sistem. Kelas ini menggunakan anotasi dari Java Persistence API (JPA) untuk memberikan pemetaan objek-relasional ke database.

MahasiswaService.java

```

1 package com.labti.springHibernate.service;
2
3 import com.labti.springHibernate.model.Mahasiswa;
4 import java.util.List;
5
6 public interface MahasiswaService {
7     public void save(Mahasiswa mahasiswa);
8     public void update(Mahasiswa mahasiswa);
9     public void delete(Mahasiswa mahasiswa);
10    public Mahasiswa getMahasiswa(String npm);
11    public List<Mahasiswa> getMahasiswas();
12 }

```

Ini adalah interface dengan abstraksi untuk operasi-operasi dasar pada entitas Mahasiswa untuk pengelolaan data seperti menyimpan, mengupdate, menghapus, dan mengambil data Mahasiswa dari penyimpanan data.

MahasiswaServiceImpl.java

```

1 package com.labti.springHibernate.service;
2
3 import com.labti.springHibernate.dao.MahasiswaDAO;
4 import com.labti.springHibernate.model.Mahasiswa;
5 import java.util.List;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Service;
8 import org.springframework.transaction.annotation.Transactional;
9
10 @Service("MahasiswaService")
11 @Transactional(readOnly = true)
12
13 public class MahasiswaServiceImpl implements MahasiswaService {
14     @Autowired
15     private MahasiswaDAO mahasiswaDao;
16
17     @Transactional
18     @Override
19     public void save(Mahasiswa mahasiswa) {
20         mahasiswaDao.save(mahasiswa);
21     }
22
23     @Transactional
24     @Override
25     public void update(Mahasiswa mahasiswa) {
26         mahasiswaDao.update(mahasiswa);
27     }
28
29     @Transactional
30     @Override
31     public void delete(Mahasiswa mahasiswa) {
32         mahasiswaDao.delete(mahasiswa);
33     }

```

```

35      @Override
36      public Mahasiswa getMahasiswa(String npm) {
37          return mahasiswaDao.getMahasiswa(npm);
38      }
39
40      @Override
41      public List<Mahasiswa> getMahasiswas() {
42          return mahasiswaDao.getMahasiswas();
43      }
44  }
45

```

Kelas ini berfungsi sebagai penyedia layanan yang menghubungkan antara lapisan bisnis aplikasi dengan lapisan data melalui

MahasiswaDAO.

MahasiswaView.java

Form

Data Mahasiswa

NPM

NAMA

KELAS

ALAMAT

| NPM | NAMA | KELAS | ALAMAT |
|-----|------|-------|--------|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

| COMPONENT | TEXT | NAME* |
|---------------------------|----------------|--------|
| jLabel1 (Label) | Data Mahasiswa | - |
| jLabel2 (Label) | NPM | - |
| jLabel3 (Label) | NAMA | - |
| jLabel4 (Label) | KELAS | - |
| jLabel5 (Label) | ALAMAT | - |
| (jTextField1 (Text Field) | - | Npm |
| jTextField2 (Text Field) | - | Nama |
| jTextField3 (Text Field) | - | Kelas |
| jTextArea | - | alamat |
| jButton1 (Button) | Simpan | simpan |
| jButton1 (Button) | Update | update |
| jButton1 (Button) | Hapus | hapus |
| jTabel1 (Table) | - | Tabel |

Program

```

1 package com.labti.springHibernate.view;
2
3 import com.labti.springHibernate.controller.MahasiswaController;
4
5
6 public class MahasiswaView extends javax.swing.JFrame {
7     private final MahasiswaController mahasiswaController = new MahasiswaController(this);
8
9
10    public MahasiswaView() {
11        initComponents();
12        mahasiswaController.tampilData();
13    }
14
15    @SuppressWarnings("unchecked")
16    Generated Code

```

```

168 private void simpanActionPerformed(java.awt.event.ActionEvent evt) {
169     mahasiswaController.saveMahasiswa();
170 }
171
172 private void updateActionPerformed(java.awt.event.ActionEvent evt) {
173     mahasiswaController.updateMahasiswa();
174 }
175
176 private void hapusActionPerformed(java.awt.event.ActionEvent evt) {
177     mahasiswaController.deleteMahasiswa();
178 }
179
180 private void TabelMouseClicked(java.awt.event.MouseEvent evt) {
181     mahasiswaController.show();
182 }

```

Fungsi utama dari kelas ini adalah sebagai wadah untuk elemen-elemen GUI dan berinteraksi dengan MahasiswaController untuk menangani logika bisnis terkait entitas Mahasiswa.

| Nama | NIM | Kelas | Nama Mata Kuliah | Kode Mata Kuliah |
|--------------------------|----------|-------|----------------------------|------------------|
| Javier Nathaniel Damanik | 86915473 | 4KA15 | Sistem Informasi Manajemen | 32 |

Seperti inilah tampilan output dari Spring dan Hibernate Data Mahasiswa.

Selanjutnya pada program yang berbeda, yaitu membuat IReport menggunakan Jasperreport untuk membuat laporan aplikasi dengan bahasa pemrograman Java XML

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    try{  
        JasperPrint jp = Jasper.FillManager.fillReport((getClass().getResourceAsStream("reportMahasiswa.jasper"), null, koneksi.getConnection()),  
        JasperViewer = viewReport (jp, false);  
    } catch (Exception e) {  
        JOptionPane.showMessageDialog (rootPane, e);  
    }  
}
```

Program ini adalah bagian dari aplikasi Java Swing yang menggunakan JasperReports untuk membuat dan menampilkan laporan (report).

```

package view;
import java.sql.DriverManager;

public class Koneksi {
    public static Connection getConnection () {
        Connection connection = null;
        String driver "com.mysql.jdbc.Driver";
        String url="jdbc:mysql://localhost:3306/kampus":
        String user="root";
        String password="";
        if (connection == null) {
            try{
                Class.forName(driver);
                connection = DriverManager.getConnection (url, user, password);
            }catch (ClassNotFoundException | SQLException error )
                System.exit(0);
            }
        }
        return connection;
    }
}

```

Program ini adalah bagian dari kelas Koneksi yang bertanggung jawab untuk mengelola koneksi ke database MySQL.

```

package view;

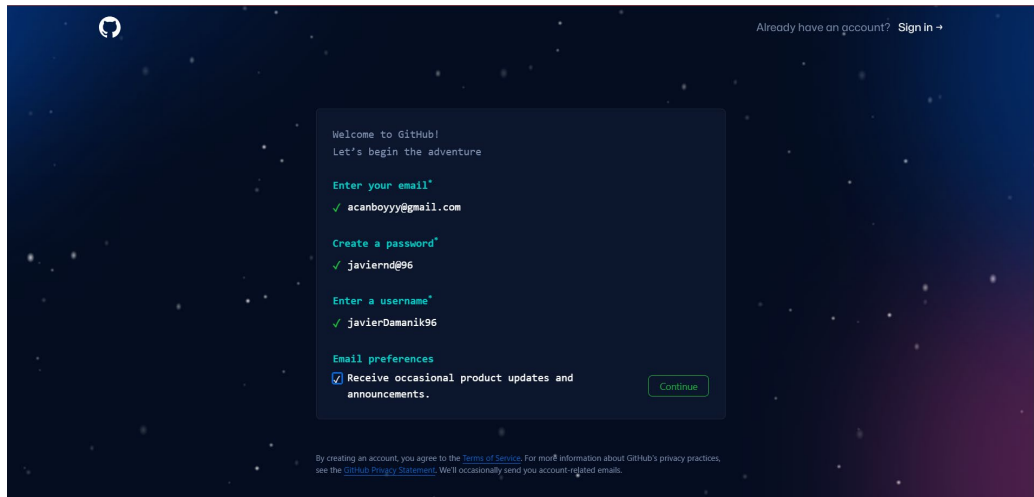
public class reportfix
    public static void main(String[] args) {
        new reporttbl ().setVisible (true);
    }
}

```

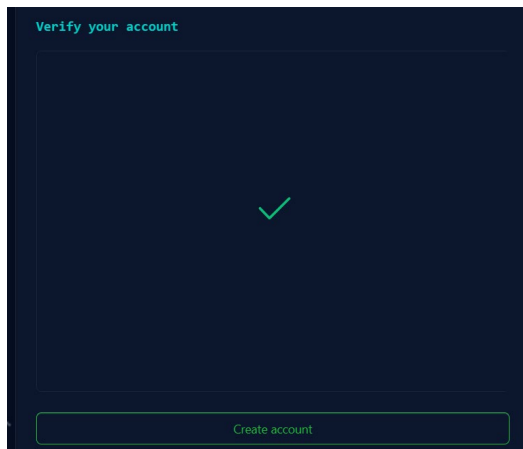
Ini adalah kelas Java dengan nama reportfix. Terdapat metode main yang merupakan metode utama yang akan dijalankan saat program dijalankan.

| | | | | | |
|---|--------|----------|---------------------------|--------------|---|
| <h1>Laporan Data</h1> <p>Data Mahasiswa</p> | | | <h1>ta</h1> <p>asiswa</p> | |  |
| | | | Wednesday, 1 December | | |
| Nrp | Nama | Angkatan | Angkatan | Sekolah asal | |
| 1 | Javier | 2020 | 2020 | Bekasi | |

Seperti ini ouput dari program iReport tersebut
 Program yang terakhir yaitu membuat repository dan mengirim file melalui GIT dan GITHUB.










Untuk menggunakan github, maka pertama -tama yaitu membuat akun github dengan sign up pada github dengan membuat email, password , dan username. Kemudian memncentang Email preferences untuk update dan informasi yang muncul pada email.



Setelah itu maka verifikasi akun github tersebut menggunakan captcha yang dimana user harus mengarahkan posisi benda tertentu searah telunjuk tangan. Jika sudah berhasil maka klik Create account.

What specific features are you interested in using?

Select all that apply so we can point you to the right GitHub plan.


- ☐  **Collaborative coding**
Codespaces, Pull requests, Notifications, Code review, Code review assignments, Code owners, Draft pull requests, Protected branches, and more.
- ☐  **Automation and CI/CD**
Actions, Packages, APIs, GitHub Pages, GitHub Marketplace, Webhooks, Hosted runners, Self-hosted runners, Secrets management, and more.
- ☐  **Security**
Private repos, 2FA, Required reviews, Required status checks, Code scanning, Secret scanning, Dependency graph, Dependabot alerts, and more.
- ☐  **Client Apps**
GitHub Mobile, GitHub CLI, and GitHub Desktop.
- ☐  **Project Management**
Projects, Labels, Milestones, Issues, Unified Contribution Graph, Org activity graph, Org dependency insights, Repo insights, Wikis, and GitHub Insights.
- ☐  **Team Administration**
Organizations, Invitations, Team sync, Custom roles, Domain verification, Audit Log API, Repo creation restriction, and Notification restriction.
- ☐  **Community**
GitHub Marketplace, GitHub Sponsors, GitHub Skills, and Electron.


Kemudian untuk fitur khusus dapat user pilih sesuai kebutuhannya.

Start a new repository

A repository contains all of your project's files, revision history, and collaborator discussion.

javierDamanik96 /

☐  **Public**
Anyone on the internet can see this repository

☒  **Private**
You choose who can see and commit to this repository

[Create a new repository](#)

Jika akun sudah dibuat maka akan tampil homepage dari akun github user tersebut. Selanjutnya untuk membuat repository maka klik Create a new repository. Repository yang

dibuat yaitu TB2 sesuai dengan nama folder yang dituju di perangkat.

```
MINGW64:/c/Users/Javier uceks

Javier uceks@DESKTOP-LFBEGA2 MINGW64 ~/OneDrive/Desktop
$ cd

Javier uceks@DESKTOP-LFBEGA2 MINGW64 ~
$ git config --global user.name "javierDamanik96"

Javier uceks@DESKTOP-LFBEGA2 MINGW64 ~
$ git config --global user.email acanboyyy@gmail.com

Javier uceks@DESKTOP-LFBEGA2 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/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=javierDamanik96
user.email=acanboyyy@gmail.com

Javier uceks@DESKTOP-LFBEGA2 MINGW64 ~
$ |
```

Setelah membuat repository pada github, selanjutnya buka git bash pada dekstop. Git Bash adalah sebuah shell yang memungkinkan user untuk berinteraksi dengan sistem kontrol versi Git melalui antarmuka baris perintah pada sistem operasi Windows. Program Git ini menjalankan beberapa perintah konfigurasi dasar untuk menyesuaikan perilaku Git sesuai dengan preferensi user pada GIT.

```
Javier uceks@DESKTOP-LFBEGA2 MINGW64 ~/OneDrive/Desktop/TUGAS SMT 7 JAVIER/PRAKT
IKUM RPL 2/pert 7 rpl 2 B/TB2
$ git init
Initialized empty Git repository in C:/Users/Javier uceks/OneDrive/Desktop/TUGAS
SMT 7 JAVIER/PRAKTIKUM RPL 2/pert 7 rpl 2 B/TB2/.git/

Javier uceks@DESKTOP-LFBEGA2 MINGW64 ~/OneDrive/Desktop/TUGAS SMT 7 JAVIER/PRAKT
IKUM RPL 2/pert 7 rpl 2 B/TB2 (master)
$ git add *
```

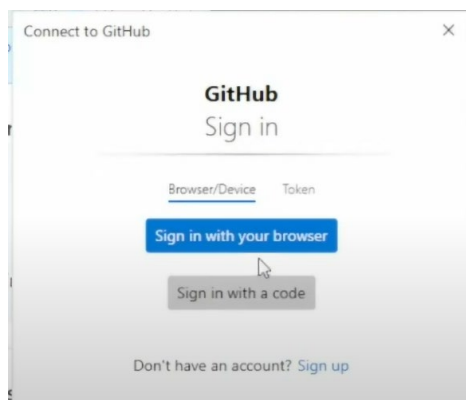
```
Javier ucks@DESKTOP-LFBEGA2 MINGW64 ~/OneDrive/Desktop/TUGAS SMT 7 JAVIER/PRAKT
IKUM RPL 2/pert 7 rp1 2 B/TB2 (master)
$ git commit -m "first init github"
[master (root-commit) f48a2a6] first init github
28 files changed, 2863 insertions(+)
create mode 100644 build.xml
create mode 100644 build/built-jar.properties
create mode 100644 build/classes/.netbeans_automatic_build
create mode 100644 build/classes/.netbeans_update_resources
create mode 100644 build/classes/data_mahasiswa/ControllerMahasiswa$1.class
create mode 100644 build/classes/data_mahasiswa/ControllerMahasiswa.class
create mode 100644 build/classes/data_mahasiswa/Data_Mahasiswa.class
create mode 100644 build/classes/data_mahasiswa/ViewMahasiswa$1.class
create mode 100644 build/classes/data_mahasiswa/ViewMahasiswa$2.class
create mode 100644 build/classes/data_mahasiswa/ViewMahasiswa$3.class
create mode 100644 build/classes/data_mahasiswa/ViewMahasiswa$4.class
create mode 100644 build/classes/data_mahasiswa/ViewMahasiswa$5.class
create mode 100644 build/classes/data_mahasiswa/ViewMahasiswa.class
create mode 100644 build/classes/data_mahasiswa/ViewMahasiswa.form
create mode 100644 build/classes/data_mahasiswa/mahasiswa_masuk.class
create mode 100644 build/classes/data_mahasiswa/mahasiswa_masuk.rs
create mode 100644 manifest.mf
create mode 100644 nbproject/build-impl.xml
create mode 100644 nbproject/genfiles.properties
create mode 100644 nbproject/private/private.properties
create mode 100644 nbproject/private/private.xml
create mode 100644 nbproject/project.properties
create mode 100644 nbproject/project.xml
create mode 100644 src/data_mahasiswa/ControllerMahasiswa.java
create mode 100644 src/data_mahasiswa/Data_Mahasiswa.java
create mode 100644 src/data_mahasiswa/ViewMahasiswa.form
create mode 100644 src/data_mahasiswa/ViewMahasiswa.java
create mode 100644 src/data_mahasiswa/mahasiswa_masuk.java

Javier ucks@DESKTOP-LFBEGA2 MINGW64 ~/OneDrive/Desktop/TUGAS SMT 7 JAVIER/PRAKT
IKUM RPL 2/pert 7 rp1 2 B/TB2 (master)
$ git remote add origin https://github.com/javierDamanik96/TB2.git

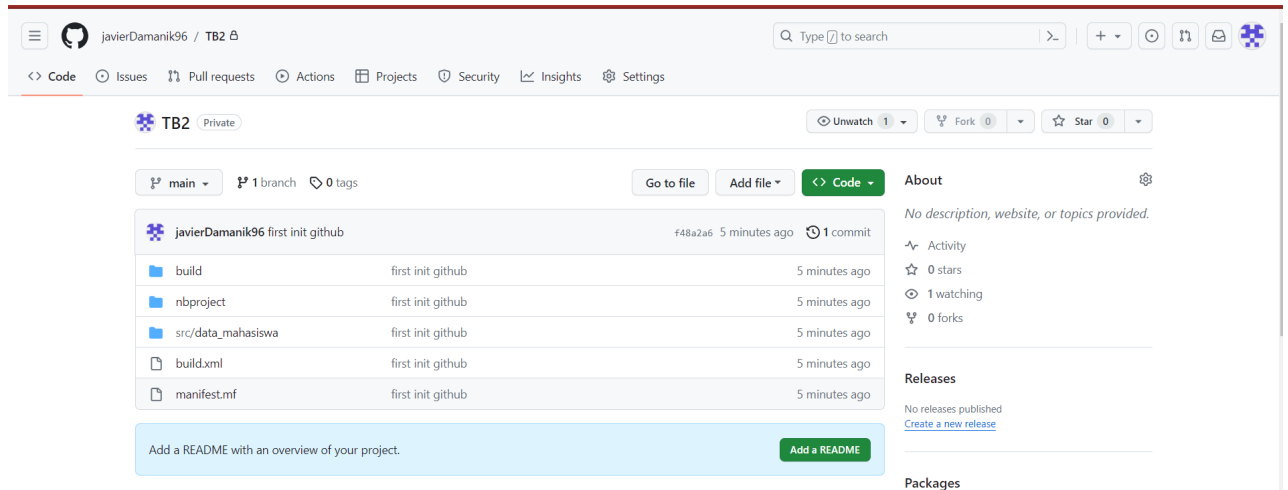
Javier ucks@DESKTOP-LFBEGA2 MINGW64 ~/OneDrive/Desktop/TUGAS SMT 7 JAVIER/PRAKT
IKUM RPL 2/pert 7 rp1 2 B/TB2 (master)
$ git branch -M main
```

```
Javier ucks@DESKTOP-LFBEGA2 MINGW64 ~/OneDrive/Desktop/TUGAS SMT 7 JAVIER/PRAKT
IKUM RPL 2/pert 7 rp1 2 B/TB2 (main)
$ git push -u origin main
Enumerating objects: 35, done.
Counting objects: 100% (35/35), done.
Delta compression using up to 16 threads
Compressing objects: 100% (33/33), done.
Writing objects: 100% (35/35), 30.70 KiB | 5.12 MiB/s, done.
Total 35 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/javierDamanik96/TB2.git
 * [new branch] main -> main
branch 'main' set up to track 'origin/main'.
```

Ini adalah serangkaian program untuk membuat repositori Git baru, menambahkan file, melakukan commit, menghubungkannya dengan repositori GitHub, dan mengirimkan perubahan ke repositori jarak jauh.



Setelah itu, maka akan muncul tampilan seperti ini untuk sign in pada github, untuk menghubungkan git dengan github. Sesudah akun user terhubung, maka folder tersebut (TB2) akan muncul di repository.



Seperti ini jika folder file TB2 tersebut sudah masuk pada repository Github tersebut.