Tugas Praktikum 13

GUI Database

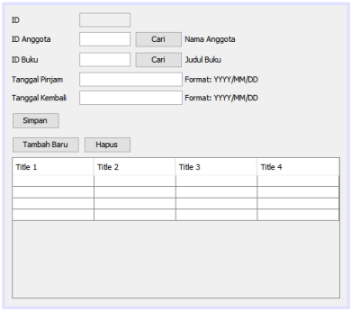


Rafi Ody Prasetyo  
(2341720180)

D-IV Teknik Informatika  
Politeknik Negeri Malang  
Semester 3  
2024

**Tugas**

1. Buatlah class Peminjaman.
2. Buatlah form FrmPeminjaman dan susun sebagai berikut:



1. Atur kode program agar dapat menangani transaksi peminjaman dan pengembalian. Note:

Pada textbox ID Anggota, pengguna tinggal memasukkan ID anggota, kemudian menekan tombol Cari. Jika ketemu, maka label “Nama Anggota” yang ada di samping tombol Cari tersebut akan menampilkan nama anggota dari ID yang dimasukkan tadi. Begitu juga dengan ID Buku.

Peminjaman.java

Class ini digunakan untuk mengatur proses crud yang berjalan pada GUI peminjaman.

public class Peminjaman {

private int idpeminjaman;

private Buku buku = new Buku();

private Anggota anggota = new Anggota();

private Date tanggalpinjam;

private Date tanggalkembali;

private static final SimpleDateFormat DATE\_FORMAT = new SimpleDateFormat("yyyy-MM-dd");

public int getIdpeminjaman() {

return idpeminjaman;

}

public void setIdpeminjaman(int idpeminjaman) {

this.idpeminjaman = idpeminjaman;

}

public Buku getBuku() {

return buku;

}

public void setBuku(Buku buku) {

this.buku = buku;

}

public Anggota getAnggota() {

return anggota;

}

public void setAnggota(Anggota anggota) {

this.anggota = anggota;

}

public String getTanggalpinjam() {

return (tanggalpinjam != null) ? DATE\_FORMAT.format(tanggalpinjam) : null;

}

public void setTanggalpinjam(String tanggalpinjam) {

try {

this.tanggalpinjam = DATE\_FORMAT.parse(tanggalpinjam);

} catch (Exception e) {

e.printStackTrace();

}

}

public String getTanggalkembali() {

return (tanggalkembali != null) ? DATE\_FORMAT.format(tanggalkembali) : null;

}

public void setTanggalkembali(String tanggalkembali) {

try {

this.tanggalkembali = DATE\_FORMAT.parse(tanggalkembali);

} catch (Exception e) {

e.printStackTrace();

}

}

public Peminjaman getById(int id) {

Peminjaman pj = new Peminjaman();

ResultSet rs = DBHelper.selectQuery("SELECT p.idpeminjaman, a.idanggota, b.idbuku, p.tanggalpinjam, "+

"p.tanggalkembali FROM peminjaman AS p INNER JOIN anggota AS a "+

"ON p.idanggota = a.idanggota INNER JOIN buku AS b ON p.idbuku = b.idbuku "+

"WHERE p.idpeminjaman = '"+id+"'");

try{

while(rs.next()){

pj = new Peminjaman();

pj.setIdpeminjaman(rs.getInt("idpeminjaman"));

pj.getBuku().setIdBuku(rs.getInt("idbuku"));

pj.getAnggota().setIdanggota(rs.getInt("idanggota"));

pj.setTanggalpinjam(rs.getString("tanggalpinjam"));

pj.setTanggalkembali(rs.getString("tanggalkembali"));

}

} catch(Exception e) {

e.printStackTrace();

}

return pj;

}

public ArrayList<Peminjaman> getAll() {

ArrayList<Peminjaman> ListPeminjaman = new ArrayList();

ResultSet rs = DBHelper.selectQuery("SELECT p.idpeminjaman, a.idanggota, "+

"a.nama, a.alamat, a.telepon, b.idbuku, "+

"b.idkategori, b.judul, b.penerbit, b.penulis, "+

"p.tanggalpinjam, p.tanggalkembali FROM peminjaman AS p "+

"INNER JOIN anggota AS a ON p.idanggota = a.idanggota "+

"INNER JOIN buku as b ON p.idbuku = b.idbuku");

try{

while(rs.next()){

Peminjaman pj = new Peminjaman();

pj.setIdpeminjaman(rs.getInt("idpeminjaman"));

pj.getBuku().setIdBuku(rs.getInt("idbuku"));

pj.getAnggota().setIdanggota(rs.getInt("idanggota"));

pj.setTanggalpinjam(rs.getString("tanggalpinjam"));

pj.setTanggalkembali(rs.getString("tanggalkembali"));

ListPeminjaman.add(pj);

}

} catch(Exception e) {

e.printStackTrace();

}

return ListPeminjaman;

}

public ArrayList<Peminjaman> search(String keyword) {

ArrayList<Peminjaman> ListPeminjaman = new ArrayList();

ResultSet rs = DBHelper.selectQuery("SELECT p.idpeminjaman, a.idanggota, b.idbuku, p.tanggalpinjam, "+

"p.tanggalkembali FROM peminjaman AS p INNER JOIN anggota AS a "+

"ON p.idanggota = a.idanggota INNER JOIN buku AS b ON p.idbuku = b.idbuku "+

"WHERE p.idpeminjaman LIKE '%"+keyword+"%' "+

"OR a.idanggota LIKE '%"+keyword+"%' "+

"OR b.idbuku LIKE '%"+keyword+"%' "+

"OR p.tanggalpinjam LIKE '%"+keyword+"%' "+

"OR p.tanggalkembali LIKE '%"+keyword+"%'");

try{

while(rs.next()){

Peminjaman pj = new Peminjaman();

pj.setIdpeminjaman(rs.getInt("idpeminjaman"));

pj.getBuku().setIdBuku(rs.getInt("idbuku"));

pj.getAnggota().setIdanggota(rs.getInt("idanggota"));

pj.setTanggalpinjam(rs.getString("tanggalpinjam"));

pj.setTanggalkembali(rs.getString("tanggalkembali"));

ListPeminjaman.add(pj);

}

} catch(Exception e) {

e.printStackTrace();

}

return ListPeminjaman;

}

public void save() {

String formattedTanggalPinjam = (tanggalpinjam != null) ? DATE\_FORMAT.format(tanggalpinjam) : null;

String formattedTanggalKembali = (tanggalkembali != null) ? DATE\_FORMAT.format(tanggalkembali) : null;

if(getById(idpeminjaman).getIdpeminjaman() == 0){

String SQL = "INSERT INTO peminjaman (idanggota, idbuku, tanggalpinjam, tanggalkembali) VALUES"

+"('"+this.getAnggota().getIdanggota()+"', '"+this.getBuku().getIdBuku()+

"', '"+formattedTanggalPinjam+"', '"+formattedTanggalKembali+"')";

this.idpeminjaman = DBHelper.insertQueryGetId(SQL);

} else {

String SQL = "UPDATE peminjaman SET "

+"idanggota = '"+this.getAnggota().getIdanggota()+"', idbuku = '"+this.getBuku().getIdBuku()+"', tanggalpinjam = '"

+formattedTanggalPinjam+"', tanggalkembali = '"+formattedTanggalKembali+"' WHERE idpeminjaman = '"+this.idpeminjaman+"'";

this.idpeminjaman = DBHelper.insertQueryGetId(SQL);

}

}

public void delete() {

String SQL = "DELETE FROM peminjaman WHERE idpeminjaman = '" + this.idpeminjaman + "'";

DBHelper.insertQueryGetId(SQL);

}

}

Inputan untuk tanggalPinjam dan tanggalKembali di konversi dalam bentuk String untuk menyesuaikan form input pada GUI.

public void setTanggalpinjam(String tanggalpinjam) {

try {

this.tanggalpinjam = DATE\_FORMAT.parse(tanggalpinjam);

} catch (Exception e) {

e.printStackTrace();

}

}

public String getTanggalkembali() {

return (tanggalkembali != null) ? DATE\_FORMAT.format(tanggalkembali) : null;

}

public void setTanggalkembali(String tanggalkembali) {

try {

this.tanggalkembali = DATE\_FORMAT.parse(tanggalkembali);

} catch (Exception e) {

e.printStackTrace();

}

}

public Peminjaman getById(int id) {

Peminjaman pj = new Peminjaman();

ResultSet rs = DBHelper.selectQuery("SELECT p.idpeminjaman, a.idanggota, b.idbuku, p.tanggalpinjam, "+

"p.tanggalkembali FROM peminjaman AS p INNER JOIN anggota AS a "+

"ON p.idanggota = a.idanggota INNER JOIN buku AS b ON p.idbuku = b.idbuku "+

"WHERE p.idpeminjaman = '"+id+"'");

try{

while(rs.next()){

pj = new Peminjaman();

pj.setIdpeminjaman(rs.getInt("idpeminjaman"));

pj.getBuku().setIdBuku(rs.getInt("idbuku"));

pj.getAnggota().setIdanggota(rs.getInt("idanggota"));

pj.setTanggalpinjam(rs.getString("tanggalpinjam"));

pj.setTanggalkembali(rs.getString("tanggalkembali"));

}

} catch(Exception e) {

e.printStackTrace();

}

return pj;

}

public ArrayList<Peminjaman> getAll() {

ArrayList<Peminjaman> ListPeminjaman = new ArrayList();

ResultSet rs = DBHelper.selectQuery("SELECT p.idpeminjaman, a.idanggota, "+

"a.nama, a.alamat, a.telepon, b.idbuku, "+

"b.idkategori, b.judul, b.penerbit, b.penulis, "+

"p.tanggalpinjam, p.tanggalkembali FROM peminjaman AS p "+

"INNER JOIN anggota AS a ON p.idanggota = a.idanggota "+

"INNER JOIN buku as b ON p.idbuku = b.idbuku");

try{

while(rs.next()){

Peminjaman pj = new Peminjaman();

pj.setIdpeminjaman(rs.getInt("idpeminjaman"));

pj.getBuku().setIdBuku(rs.getInt("idbuku"));

pj.getAnggota().setIdanggota(rs.getInt("idanggota"));

pj.setTanggalpinjam(rs.getString("tanggalpinjam"));

pj.setTanggalkembali(rs.getString("tanggalkembali"));

ListPeminjaman.add(pj);

}

} catch(Exception e) {

e.printStackTrace();

}

return ListPeminjaman;

}

public ArrayList<Peminjaman> search(String keyword) {

ArrayList<Peminjaman> ListPeminjaman = new ArrayList();

ResultSet rs = DBHelper.selectQuery("SELECT p.idpeminjaman, a.idanggota, b.idbuku, p.tanggalpinjam, "+

"p.tanggalkembali FROM peminjaman AS p INNER JOIN anggota AS a "+

"ON p.idanggota = a.idanggota INNER JOIN buku AS b ON p.idbuku = b.idbuku "+

"WHERE p.idpeminjaman LIKE '%"+keyword+"%' "+

"OR a.idanggota LIKE '%"+keyword+"%' "+

"OR b.idbuku LIKE '%"+keyword+"%' "+

"OR p.tanggalpinjam LIKE '%"+keyword+"%' "+

"OR p.tanggalkembali LIKE '%"+keyword+"%'");

try{

while(rs.next()){

Peminjaman pj = new Peminjaman();

pj.setIdpeminjaman(rs.getInt("idpeminjaman"));

pj.getBuku().setIdBuku(rs.getInt("idbuku"));

pj.getAnggota().setIdanggota(rs.getInt("idanggota"));

pj.setTanggalpinjam(rs.getString("tanggalpinjam"));

pj.setTanggalkembali(rs.getString("tanggalkembali"));

ListPeminjaman.add(pj);

}

} catch(Exception e) {

e.printStackTrace();

}

return ListPeminjaman;

}

public void save() {

String formattedTanggalPinjam = (tanggalpinjam != null) ? DATE\_FORMAT.format(tanggalpinjam) : null;

String formattedTanggalKembali = (tanggalkembali != null) ? DATE\_FORMAT.format(tanggalkembali) : null;

if(getById(idpeminjaman).getIdpeminjaman() == 0){

String SQL = "INSERT INTO peminjaman (idanggota, idbuku, tanggalpinjam, tanggalkembali) VALUES"

+"('"+this.getAnggota().getIdanggota()+"', '"+this.getBuku().getIdBuku()+

"', '"+formattedTanggalPinjam+"', '"+formattedTanggalKembali+"')";

this.idpeminjaman = DBHelper.insertQueryGetId(SQL);

} else {

String SQL = "UPDATE peminjaman SET "

+"idanggota = '"+this.getAnggota().getIdanggota()+"', idbuku = '"+this.getBuku().getIdBuku()+"', tanggalpinjam = '"

+formattedTanggalPinjam+"', tanggalkembali = '"+formattedTanggalKembali+"' WHERE idpeminjaman = '"+this.idpeminjaman+"'";

this.idpeminjaman = DBHelper.insertQueryGetId(SQL);

}

}

public void delete() {

String SQL = "DELETE FROM peminjaman WHERE idpeminjaman = '" + this.idpeminjaman + "'";

DBHelper.insertQueryGetId(SQL);

}

}

public ArrayList<Peminjaman> getAll() {

ArrayList<Peminjaman> ListPeminjaman = new ArrayList();

ResultSet rs = DBHelper.selectQuery("SELECT p.idpeminjaman, a.idanggota, "+

"a.nama, a.alamat, a.telepon, b.idbuku, "+

"b.idkategori, b.judul, b.penerbit, b.penulis, "+

"p.tanggalpinjam, p.tanggalkembali FROM peminjaman AS p "+

"INNER JOIN anggota AS a ON p.idanggota = a.idanggota "+

"INNER JOIN buku as b ON p.idbuku = b.idbuku");

try{

while(rs.next()){

Peminjaman pj = new Peminjaman();

pj.setIdpeminjaman(rs.getInt("idpeminjaman"));

pj.getBuku().setIdBuku(rs.getInt("idbuku"));

pj.getAnggota().setIdanggota(rs.getInt("idanggota"));

pj.setTanggalpinjam(rs.getString("tanggalpinjam"));

pj.setTanggalkembali(rs.getString("tanggalkembali"));

ListPeminjaman.add(pj);

}

} catch(Exception e) {

e.printStackTrace();

}

return ListPeminjaman;

}

public ArrayList<Peminjaman> search(String keyword) {

ArrayList<Peminjaman> ListPeminjaman = new ArrayList();

ResultSet rs = DBHelper.selectQuery("SELECT p.idpeminjaman, a.idanggota, b.idbuku, p.tanggalpinjam, "+

"p.tanggalkembali FROM peminjaman AS p INNER JOIN anggota AS a "+

"ON p.idanggota = a.idanggota INNER JOIN buku AS b ON p.idbuku = b.idbuku "+

"WHERE p.idpeminjaman LIKE '%"+keyword+"%' "+

"OR a.idanggota LIKE '%"+keyword+"%' "+

"OR b.idbuku LIKE '%"+keyword+"%' "+

"OR p.tanggalpinjam LIKE '%"+keyword+"%' "+

"OR p.tanggalkembali LIKE '%"+keyword+"%'");

try{

while(rs.next()){

Peminjaman pj = new Peminjaman();

pj.setIdpeminjaman(rs.getInt("idpeminjaman"));

pj.getBuku().setIdBuku(rs.getInt("idbuku"));

pj.getAnggota().setIdanggota(rs.getInt("idanggota"));

pj.setTanggalpinjam(rs.getString("tanggalpinjam"));

pj.setTanggalkembali(rs.getString("tanggalkembali"));

ListPeminjaman.add(pj);

}

} catch(Exception e) {

e.printStackTrace();

}

return ListPeminjaman;

}

public void save() {

String formattedTanggalPinjam = (tanggalpinjam != null) ? DATE\_FORMAT.format(tanggalpinjam) : null;

String formattedTanggalKembali = (tanggalkembali != null) ? DATE\_FORMAT.format(tanggalkembali) : null;

if(getById(idpeminjaman).getIdpeminjaman() == 0){

String SQL = "INSERT INTO peminjaman (idanggota, idbuku, tanggalpinjam, tanggalkembali) VALUES"

+"('"+this.getAnggota().getIdanggota()+"', '"+this.getBuku().getIdBuku()+

"', '"+formattedTanggalPinjam+"', '"+formattedTanggalKembali+"')";

this.idpeminjaman = DBHelper.insertQueryGetId(SQL);

} else {

String SQL = "UPDATE peminjaman SET "

+"idanggota = '"+this.getAnggota().getIdanggota()+"', idbuku = '"+this.getBuku().getIdBuku()+"', tanggalpinjam = '"

+formattedTanggalPinjam+"', tanggalkembali = '"+formattedTanggalKembali+"' WHERE idpeminjaman = '"+this.idpeminjaman+"'";

this.idpeminjaman = DBHelper.insertQueryGetId(SQL);

}

}

public void delete() {

String SQL = "DELETE FROM peminjaman WHERE idpeminjaman = '" + this.idpeminjaman + "'";

DBHelper.insertQueryGetId(SQL);

}

}

try{

while(rs.next()){

Peminjaman pj = new Peminjaman();

pj.setIdpeminjaman(rs.getInt("idpeminjaman"));

pj.getBuku().setIdBuku(rs.getInt("idbuku"));

pj.getAnggota().setIdanggota(rs.getInt("idanggota"));

pj.setTanggalpinjam(rs.getString("tanggalpinjam"));

pj.setTanggalkembali(rs.getString("tanggalkembali"));

ListPeminjaman.add(pj);

}

} catch(Exception e) {

e.printStackTrace();

}

return ListPeminjaman;

}

public void save() {

String formattedTanggalPinjam = (tanggalpinjam != null) ? DATE\_FORMAT.format(tanggalpinjam) : null;

String formattedTanggalKembali = (tanggalkembali != null) ? DATE\_FORMAT.format(tanggalkembali) : null;

if(getById(idpeminjaman).getIdpeminjaman() == 0){

String SQL = "INSERT INTO peminjaman (idanggota, idbuku, tanggalpinjam, tanggalkembali) VALUES"

+"('"+this.getAnggota().getIdanggota()+"', '"+this.getBuku().getIdBuku()+

"', '"+formattedTanggalPinjam+"', '"+formattedTanggalKembali+"')";

this.idpeminjaman = DBHelper.insertQueryGetId(SQL);

} else {

String SQL = "UPDATE peminjaman SET "

+"idanggota = '"+this.getAnggota().getIdanggota()+"', idbuku = '"+this.getBuku().getIdBuku()+"', tanggalpinjam = '"

+formattedTanggalPinjam+"', tanggalkembali = '"+formattedTanggalKembali+"' WHERE idpeminjaman = '"+this.idpeminjaman+"'";

this.idpeminjaman = DBHelper.insertQueryGetId(SQL);

}

}

public void delete() {

String SQL = "DELETE FROM peminjaman WHERE idpeminjaman = '" + this.idpeminjaman + "'";

DBHelper.insertQueryGetId(SQL);

}

}

GuiPeminjaman.java

public class GuiPeminjaman extends javax.swing.JFrame {

/\*\*

\* Creates new form GuiPeminjaman

\*/

public GuiPeminjaman() {

initComponents();

tampilkanData();

kosongkanForm();

}

public void kosongkanForm() {

txtIdBuku.setText("0");

txtIdAnggota.setText("");

txtIdBuku.setText("");

txtTglPinjam.setText("");

txtTglKembali.setText("");

}

public void tampilkanData() {

String[] kolom = {"ID", "ID Anggota", "ID Buku", "Tanggal Pinjam", "Tanggal Kembali"};

ArrayList<Peminjaman> list = new Peminjaman().getAll();

Object rowData[] = new Object[5];

tblPeminjaman.setModel(new DefaultTableModel(new Object[][] {}, kolom));

for(Peminjaman pj : list) {

rowData[0] = pj.getIdpeminjaman();

rowData[1] = pj.getAnggota().getIdanggota();

rowData[2] = pj.getBuku().getIdBuku();

rowData[3] = pj.getTanggalpinjam();

rowData[4] = pj.getTanggalkembali();

((DefaultTableModel)tblPeminjaman.getModel()).addRow(rowData);

}

}

public void cari(String keyword) {

String[] kolom = {"ID", "ID Anggota", "ID Buku", "Tanggal Pinjam", "Tanggal Kembali"};

ArrayList<Peminjaman> list = new Peminjaman().search(keyword);

Object rowData[] = new Object[5];

tblPeminjaman.setModel(new DefaultTableModel(new Object[][] {}, kolom));

for(Peminjaman pj : list) {

rowData[0] = pj.getIdpeminjaman();

rowData[1] = pj.getAnggota().getNama();

rowData[2] = pj.getBuku().getJudul();

rowData[3] = pj.getTanggalpinjam();

rowData[4] = pj.getTanggalkembali();

((DefaultTableModel)tblPeminjaman.getModel()).addRow(rowData);

}

}

public void cariAnggotaById(String keyword) {

ArrayList<Anggota> list = new Anggota().search(keyword);

if(!list.isEmpty()){

String namaAnggota = list.get(0).getNama();

cariAnggota.setText(namaAnggota);

} else {

cariAnggota.setText("Tidak ditemukan");

}

}

public void cariBukuById(String keyword) {

ArrayList<Buku> ListBuku = new Buku().search(keyword);

if(!ListBuku.isEmpty()){

String judulBuku = ListBuku.get(0).getJudul();

cariBuku.setText(judulBuku);

} else {

cariBuku.setText("Tidak ditemukan");

}

}

public void cariAnggotaById(String keyword) {

ArrayList<Anggota> list = new Anggota().search(keyword);

if(!list.isEmpty()){

String namaAnggota = list.get(0).getNama();

cariAnggota.setText(namaAnggota);

} else {

cariAnggota.setText("Tidak ditemukan");

}

}

public void cariBukuById(String keyword) {

ArrayList<Buku> ListBuku = new Buku().search(keyword);

if(!ListBuku.isEmpty()){

String judulBuku = ListBuku.get(0).getJudul();

cariBuku.setText(judulBuku);

} else {

cariBuku.setText("Tidak ditemukan");

}

}

private void btnCariActionPerformed(java.awt.event.ActionEvent evt) {

cari(txtCari.getText());

}

private void btnHapusActionPerformed(java.awt.event.ActionEvent evt) {

DefaultTableModel model = (DefaultTableModel)tblPeminjaman.getModel();

int row = tblPeminjaman.getSelectedRow();

Peminjaman pj = new Peminjaman().getById(Integer.parseInt(model.getValueAt(row, 0).toString()));

pj.delete();

kosongkanForm();

tampilkanData();

}

private void btnTambahBaruActionPerformed(java.awt.event.ActionEvent evt) {

kosongkanForm();

}

private void btnSimpanActionPerformed(java.awt.event.ActionEvent evt) {

try {

if (txtIdAnggota.getText().isEmpty()) {

throw new NumberFormatException("ID Anggota cannot be empty!");

}

if (txtIdBuku.getText().isEmpty()) {

throw new NumberFormatException("ID Buku cannot be empty!");

}

if (txtTglPinjam.getText().isEmpty()) {

throw new NumberFormatException("Tanggal Pinjam cannot be empty!");

}

if (txtTglKembali.getText().isEmpty()) {

throw new NumberFormatException("Tanggal Kembali cannot be empty!");

}

Peminjaman pj = new Peminjaman();

Anggota ag = new Anggota();

Buku bk = new Buku();

pj.setIdpeminjaman(Integer.parseInt(txtIdPeminjaman.getText()));

ag.setIdanggota(Integer.parseInt(txtIdAnggota.getText()));

bk.setIdBuku(Integer.parseInt(txtIdBuku.getText()));

pj.setAnggota(ag);

pj.setBuku(bk);

pj.setTanggalpinjam(txtTglPinjam.getText());

pj.setTanggalkembali(txtTglKembali.getText());

pj.save();

txtIdPeminjaman.setText(Integer.toString(pj.getIdpeminjaman()));

tampilkanData();

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void tblPeminjamanMouseClicked(java.awt.event.MouseEvent evt) {

DefaultTableModel model = (DefaultTableModel)tblPeminjaman.getModel();

int row = tblPeminjaman.getSelectedRow();

Peminjaman pj = new Peminjaman();

pj = pj.getById(Integer.parseInt(model.getValueAt(row, 0).toString()));

txtIdPeminjaman.setText(String.valueOf(pj.getIdpeminjaman()));

txtIdAnggota.setText(String.valueOf(pj.getAnggota().getIdanggota()));

txtIdBuku.setText(String.valueOf(pj.getBuku().getIdBuku()));

txtTglPinjam.setText(pj.getTanggalpinjam());

txtTglKembali.setText(pj.getTanggalkembali());

}

private void tblPeminjamanMouseClicked(java.awt.event.MouseEvent evt) {

DefaultTableModel model = (DefaultTableModel)tblPeminjaman.getModel();

int row = tblPeminjaman.getSelectedRow();

Peminjaman pj = new Peminjaman();

pj = pj.getById(Integer.parseInt(model.getValueAt(row, 0).toString()));

txtIdPeminjaman.setText(String.valueOf(pj.getIdpeminjaman()));

txtIdAnggota.setText(String.valueOf(pj.getAnggota().getIdanggota()));

txtIdBuku.setText(String.valueOf(pj.getBuku().getIdBuku()));

txtTglPinjam.setText(pj.getTanggalpinjam());

txtTglKembali.setText(pj.getTanggalkembali());

}

Output:

