

LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
MODUL 5
EXCEPTION HANDLING



Oleh :

Chris Hotasi Rajagukguk

2311103126

S1SI-07-C

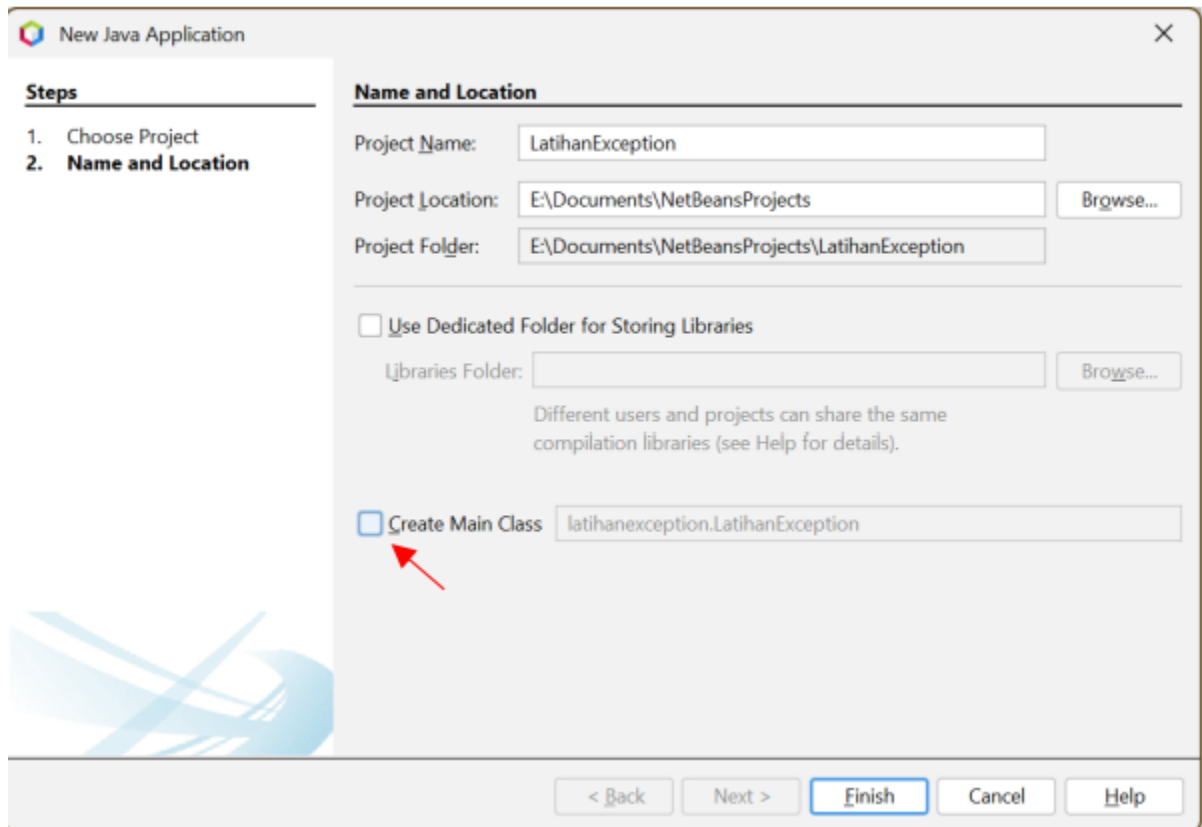
PROGRAM STUDI SISTEM INFORMASI
FAKULTAS REKAYASA INDUSTRI
UNIVERSITAS TELKOM PURWOKERTO
2024

A. Studi Kasus

Sistem Informasi Perpustakaan – Input Buku

B. GUIDED

1. Buat project baru pada Netbeans. Pada saat membuat project, hilangkan centang pada Create Main Class



2. Tambahkan package baru dengan nama perpustakaan dengan cara klik kanan pada Source Packages → New → Java Package...
3. Pada package perpustakaan, buat Class baru dengan nama Buku

Buku.java

```
package Perpustakaan;
```

```
/**
```

```
 * @author Chris Hotasi
```

```
public class Buku {  
    private String judul;  
    private String pengarang;  
    private String isbn;  
  
    public Buku(String judul, String pengarang, String isbn){  
        this.judul = judul;  
        this.pengarang = pengarang;  
        this.isbn = isbn;  
    }  
  
    public String getJudul() {  
        return judul;  
    }  
  
    public String getPengarang() {  
        return pengarang;  
    }  
  
    public String getIsbn() {  
        return isbn;  
    }  
}
```

Penjelasan : Kode di atas merupakan sebuah program Java yang mendefinisikan kelas **Buku** dalam paket **Perpustakaan**. Kelas ini digunakan untuk merepresentasikan objek buku dengan tiga atribut utama, yaitu **judul**, **pengarang**, dan **isbn**, yang semuanya bertipe data **String** dan bersifat privat. Untuk menginisialisasi nilai atribut-atribut tersebut, kelas ini memiliki sebuah konstruktor yang menerima tiga parameter, yakni **judul**, **pengarang**, dan **isbn**, lalu menetapkan nilainya ke atribut yang sesuai. Selain itu, kelas ini juga menyediakan tiga

metode akses publik (**getter**) bernama **getJudul**, **getPengarang**, dan **getIsbn**, yang masing-masing bertugas untuk mengembalikan nilai dari atribut **judul**, **pengarang**, dan **isbn**. Kode ini dirancang sederhana untuk menyimpan informasi dasar tentang buku secara terstruktur.

4. Tambahkan tampilan program untuk input data dengan cara klik kanan pada package perpustakaan → New → JFrame Form... , beri nama InputBuku
5. Buat tampilan seperti di bawah ini:

No.	Component	Variable Name
1	JPanel	
2	JScrollPane	
3	TextField	txtJudul
4	TextField	txtPengarang
5	TextField	txtIsbn
6	Button	btnSubmit
7	JTable	tblBuku
8	JLabel	lblKeterangan

6. Klik 2x pada tombol Tambah Data, tambahkan kode berikut

```
private void BtnSubmitActionPerformed(java.awt.event.ActionEvent evt) {
    String judul = txtJudul.getText();
    String pengarang = txtPengarang.getText();
    String isbn = txtIsbn.getText();

    // Membuat objek Buku baru
    Buku buku = new Buku(judul, pengarang, isbn);

    // Menambah data buku ke tabel
    DefaultTableModel data = (DefaultTableModel)
    tblBuku.getModel();
    data.addRow(new Object[]{buku.getJudul(), buku.getPengarang(),
    buku.getIsbn()});
}
```

```
// Reset form input
txtJudul.setText("");
txtPengarang.setText("");
txtIsbn.setText("");

}
```

Penjelasan : Kode di atas adalah metode **BtnSubmitActionPerformed** yang dipanggil saat tombol **Submit** diklik dalam aplikasi berbasis GUI menggunakan Java Swing. Metode ini pertama-tama mengambil data dari tiga input teks, yaitu **txtJudul**, **txtPengarang**, dan **txtIsbn**, dengan memanggil metode **getText** untuk mendapatkan teks yang diinputkan pengguna. Kemudian, kode ini membuat sebuah objek baru dari kelas **Buku** dengan nilai yang telah diambil dari input tersebut. Setelah itu, objek **Buku** digunakan untuk menambah baris data baru ke tabel **tblBuku** melalui model tabel **DefaultTableModel**, dengan menampilkan nilai atribut **judul**, **pengarang**, dan **isbn**. Setelah data berhasil ditambahkan ke tabel, form input (kolom teks) direset menjadi kosong menggunakan metode **setText** untuk mempersiapkan input berikutnya. Kode ini dirancang untuk mempermudah penambahan data buku ke dalam tabel secara dinamis.

7. Jalankan project lalu coba tambahkan data, maka akan muncul data pada tabel. Namun ketika form kosong lalu klik tambah data, maka akan ada data kosong yang masuk pada tabel. Hal ini bisa diantisipasi dengan membuat user defined exception
8. Buat sebuah class baru dengan nama **ValidasiInputException**

ValidasiInputException.java

```
package Perpustakaan;

/**
 * @author Chris Hotasi
 */
public class ValidasiInputException extends Exception {
    public ValidasiInputException(String message) {
        super(message);
    }
}
```

Penjelasan : Kode di atas mendefinisikan sebuah kelas bernama **ValidasiInputException** yang merupakan subclass dari kelas **Exception** dalam Java. Kelas ini dirancang sebagai custom exception, yang memungkinkan pengembang untuk membuat pengecualian khusus terkait validasi input. Kelas ini memiliki sebuah konstruktor yang menerima

parameter berupa pesan kesalahan (**String message**) dan meneruskannya ke konstruktor superclass (**Exception**) menggunakan kata kunci **super**. Dengan demikian, pesan kesalahan dapat digunakan untuk memberikan informasi spesifik tentang alasan pengecualian. Kelas ini berguna untuk menangani error validasi secara terstruktur dan memberikan pesan yang lebih kontekstual kepada pengguna atau sistem.

9. Ubah kode pada Class buku

Buku.java

```
package Perpustakaan;

/**
 * @author Chris Hotasi
 */
public class Buku {
    private String judul;
    private String pengarang;
    private String isbn;

    public Buku(String judul, String pengarang, String isbn) throws
ValidasiInputException {
        if (judul.isEmpty()) {
            throw new ValidasiInputException("Judul buku tidak boleh kosong!");
        }
        if (pengarang.isEmpty()) {
            throw new ValidasiInputException("Pengarang tidak boleh kosong!");
        }
        if (!isbn.matches("\\d{13}")) {
            throw new ValidasiInputException("ISBN harus 13 digit angka!");
        }

        this.judul = judul;
        this.pengarang = pengarang;
```

```

        this.isbn = isbn;

    }

    public String getJudul() {
        return judul;
    }

    public String getPengarang() {
        return pengarang;
    }

    public String getIsbn() {
        return isbn;
    }
}

```

Penjelasan : Penambahan pada kode di atas terdapat pada konstruktor kelas **Buku**, di mana kini terdapat validasi untuk memastikan bahwa data yang dimasukkan sesuai dengan ketentuan tertentu sebelum disimpan. Validasi ini menggunakan **ValidasiInputException** untuk menangani kasus-kasus berikut: jika **judul** atau **pengarang** kosong, maka akan dilemparkan pengecualian dengan pesan kesalahan yang relevan; jika **isbn** tidak memenuhi format 13 digit angka, juga akan dilemparkan pengecualian. Hal ini dilakukan dengan menggunakan metode **isEmpty** untuk memeriksa string kosong dan ekspresi reguler (**regex**) `\\d{13}` untuk memvalidasi format ISBN. Penambahan ini bertujuan untuk memastikan data yang dimasukkan valid dan mencegah kesalahan dalam pengelolaan data buku.

10. Klik 2x pada tombol Tambah Data, ubah kode berikut

```

private void BtnSubmitActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String judul = txtJudul.getText();
        String pengarang = txtPengarang.getText();
        String isbn = txtIsbn.getText();
    }
}

```

```

// Membuat objek Buku baru
Buku buku = new Buku(judul, pengarang, isbn);

// Menambah data buku ke tabel
DefaultTableModel data = (DefaultTableModel)
tblBuku.getModel();
data.addRow(new Object[]{buku.getJudul(),
buku.getPengarang(), buku.getIsbn()});

// Reset form input
txtJudul.setText("");
txtPengarang.setText("");
txtIsbn.setText("");
lblKeterangan.setForeground(Color.GREEN);
lblKeterangan.setText("Buku berhasil ditambahkan!");

} catch (ValidasiInputException e) {
    lblKeterangan.setForeground(Color.RED);
    lblKeterangan.setText("Error: " + e.getMessage());
}
}

```

Penjelasan : Penambahan pada kode metode **BtnSubmitActionPerformed** melibatkan pengelolaan validasi input menggunakan blok **try-catch** untuk menangani pengecualian **ValidasiInputException**. Jika validasi pada konstruktor kelas **Buku** gagal, pengecualian akan ditangkap di blok **catch**, dan pesan kesalahan yang relevan ditampilkan pada label **lblKeterangan** dengan warna teks merah (**Color.RED**) untuk menandakan adanya error. Sebaliknya, jika data berhasil divalidasi dan ditambahkan ke tabel, label **lblKeterangan** akan menampilkan pesan "Buku berhasil ditambahkan!" dengan warna hijau (**Color.GREEN**), memberikan umpan balik positif kepada pengguna. Penambahan ini

bertujuan untuk meningkatkan pengalaman pengguna dengan memberikan informasi yang jelas terkait keberhasilan atau kegagalan proses input.

11. Jalankan project, coba tambahkan data baru. Ketika form kosong maka akan muncul peringatan pada bagian bawah aplikasi

The image displays two screenshots of a web application interface for managing books. The top screenshot shows the 'Tambah Data' (Add Data) form with the following fields filled: Judul Buku (Harry Potter and the Cursed Child), Nama Pengarang (J. K. Rowling), and No ISBN (218726152126). Below the form is a table with columns Judul, Pengarang, and ISBN. A red error message at the bottom states 'Error: ISBN harus 13 digit angka!'. The bottom screenshot shows the same form with empty input fields. The table now contains one row of data: Judul (Harry Potter and the Curse...), Pengarang (J. K. Rowling), and ISBN (2187261521261). A green success message at the bottom states 'Buku berhasil ditambahkan!'.

Judul	Pengarang	ISBN
Harry Potter and the Cursed Child	J. K. Rowling	218726152126

Error: ISBN harus 13 digit angka!

Judul	Pengarang	ISBN
Harry Potter and the Curse...	J. K. Rowling	2187261521261

Buku berhasil ditambahkan!

C. UNGUIDED

Tambahkan inputan berikut pada aplikasi:

1. Tahun Terbit: Validasi untuk angka empat digit.

Buku.java

```
package Perpustakaan;

/**
 * @author Chris Hotasi
 */

public class Buku {

    private String judul;

    private String pengarang;

    private String isbn;

    private String tahunTerbit;

    public Buku(String judul, String pengarang, String isbn, String tahunTerbit) throws
ValidasiInputException {

        if (judul.isEmpty()) {

            throw new ValidasiInputException("Judul buku tidak boleh kosong!");

        }

        if (pengarang.isEmpty()) {

            throw new ValidasiInputException("Pengarang tidak boleh kosong!");

        }

        if (!isbn.matches("\\d{13}")) {

            throw new ValidasiInputException("ISBN harus 13 digit angka!");

        }

    }

}
```

```
}

if (!tahunTerbit.matches("\\d{4}")) {

    throw new ValidasiInputException("Tahun Terbit harus 4 digit angka!");

}


this.judul = judul;

this.pengarang = pengarang;

this.isbn = isbn;

this.tahunTerbit = tahunTerbit;


}


public String getJudul() {

    return judul;

}


public String getPengarang() {

    return pengarang;

}


public String getIsbn() {

    return isbn;

}
```

```

public String getTahunTerbit() {

    return tahunTerbit;

}

}

```

Penjelasan : Penambahan pada kode di atas terdapat pada atribut baru **tahunTerbit** yang ditambahkan ke kelas **Buku**, termasuk validasi dan metode aksesnya. Konstruktor kini menerima parameter tambahan **tahunTerbit** dan memvalidasi bahwa nilainya terdiri dari 4 digit angka menggunakan ekspresi reguler `\\d{4}`. Jika validasi gagal, akan dilemparkan pengecualian **ValidasiInputException** dengan pesan kesalahan yang relevan, misalnya "Tahun Terbit harus 4 digit angka!". Selain itu, metode akses baru **getTahunTerbit** ditambahkan untuk memungkinkan pengambilan nilai atribut **tahunTerbit**. Penambahan ini memungkinkan objek **Buku** untuk menyimpan informasi tambahan tentang tahun terbit buku secara terstruktur dan memastikan validitas datanya.

InputBuku.java

```

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

    try {

        String judul = txtJudul.getText();

        String pengarang = txtPengarang.getText();

        String isbn = txtIsbn.getText();

        String tahunTerbit = txtttahunTerbit.getText();

        // Membuat objek Buku baru

        Buku buku = new Buku(judul, pengarang, isbn, tahunTerbit);

        // Menambah data buku ke tabel

        DefaultTableModel data = (DefaultTableModel)

        tblBuku.getModel();
    }
}

```

```

        data.addRow(new Object[]{buku.getJudul(),buku.getPengarang(), buku.getIsbn(),
buku.getTahunTerbit()});

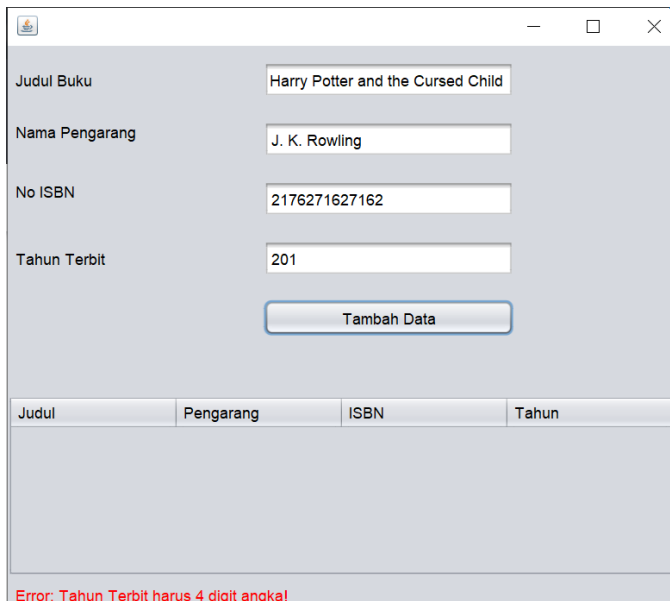
        // Reset form input
        txtJudul.setText("");
        txtPengarang.setText("");
        txtIsbn.setText("");
        txttahunTerbit.setText("");

        lblKeterangan.setForeground(Color.GREEN);
        lblKeterangan.setText("Buku berhasil ditambahkan!");
    } catch (ValidasiInputException e) {
        lblKeterangan.setForeground(Color.RED);
        lblKeterangan.setText("Error: " + e.getMessage());
    }
}

```

Penjelasan : Penambahan pada kode di atas terkait dengan atribut **tahunTerbit** yang diintegrasikan ke dalam metode **BtnSubmitActionPerformed**. Kini, nilai **tahunTerbit** diambil dari input teks **txttahunTerbit** menggunakan **getText** dan diteruskan sebagai parameter ke konstruktor kelas **Buku** saat membuat objek buku baru. Data **tahunTerbit** juga ditambahkan ke tabel dengan menyertakannya di dalam metode **addRow**, bersama atribut lain seperti **judul**, **pengarang**, dan **isbn**. Selain itu, form input **txttahunTerbit** di-reset menjadi kosong menggunakan **setText** setelah data berhasil ditambahkan. Penyesuaian ini memastikan bahwa atribut **tahunTerbit** dikelola dengan baik dalam proses input, validasi, dan penyimpanan data buku.

Output



Judul Buku: Harry Potter and the Cursed Child

Nama Pengarang: J. K. Rowling

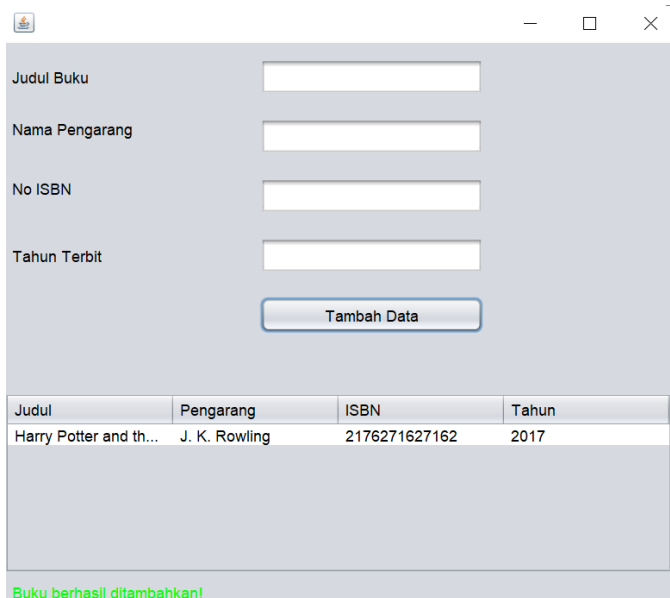
No ISBN: 2176271627162

Tahun Terbit: 201

Tambah Data

Judul	Pengarang	ISBN	Tahun
-------	-----------	------	-------

Error: Tahun Terbit harus 4 digit angka!



Judul Buku:

Nama Pengarang:

No ISBN:

Tahun Terbit:

Tambah Data

Judul	Pengarang	ISBN	Tahun
Harry Potter and th...	J. K. Rowling	2176271627162	2017

Buku berhasil ditambahkan!

2. Harga Buku: Validasi untuk angka positif.

Buku.java

```
package Perpustakaan;
```

```
/**
```

```
* @author Chris Hotasi
```

```
*/
```

```
public class Buku {
```

```
    private String judul;
```

```
    private String pengarang;
```

```
    private String isbn;
```

```
    private String tahunTerbit;
```

```
    private String harga;
```

```
    public Buku(String judul, String pengarang, String isbn, String tahunTerbit,  
    String harga) throws ValidasiInputException {
```

```
        if (judul.isEmpty()) {
```

```
            throw new ValidasiInputException("Judul buku tidak boleh kosong!");
```

```
        }
```

```
        if (pengarang.isEmpty()) {
```

```
            throw new ValidasiInputException("Pengarang tidak boleh kosong!");
```

```
        }
```

```
        if (!isbn.matches("\\d{13}")) {
```

```
            throw new ValidasiInputException("ISBN harus 13 digit angka!");
```

```
        }
```

```
        if (!tahunTerbit.matches("\\d{4}")) {
```

```
            throw new ValidasiInputException("Tahun Terbit harus 4 digit angka!");
```

```
        }
```

```
try {  
    long hargaNumber = Long.parseLong(harga);  
    if (hargaNumber < 0) {  
        throw new ValidasiInputException("Harga tidak boleh angka negatif!");  
    }  
} catch (NumberFormatException e) {  
    throw new ValidasiInputException("Input harus berupa angka!");  
}  
  
this.judul = judul;  
this.pengarang = pengarang;  
this.isbn = isbn;  
this.tahunTerbit = tahunTerbit;  
this.harga = harga;  
  
}  
  
public String getJudul() {  
    return judul;  
}  
  
public String getPengarang() {  
    return pengarang;
```



```
}

public String getIsbn() {

    return isbn;

}

public String getTahunTerbit() {

    return tahunTerbit;

}

public String getHarga() {

    return harga;

}

}
```

Penjelasan : Penambahan pada kode di atas adalah atribut baru **harga** yang merepresentasikan harga buku. Konstruktor kini menerima parameter tambahan **harga** dan memvalidasi nilainya. Validasi dilakukan dengan mencoba mengonversi **harga** menjadi tipe **long** menggunakan **Long.parseLong**. Jika nilai **harga** tidak dapat diubah menjadi angka (menimbulkan **NumberFormatException**) atau jika nilainya negatif, maka akan dilemparkan pengecualian **ValidasiInputException** dengan pesan kesalahan yang relevan, seperti "Input harus berupa angka!" atau "Harga tidak boleh angka negatif!". Atribut **harga** juga disimpan ke dalam properti kelas jika validasinya berhasil. Selain itu, metode akses baru **getHarga** ditambahkan untuk memungkinkan pengambilan nilai atribut **harga**. Penambahan ini memastikan bahwa data harga disimpan secara valid dan terstruktur dalam kelas **Buku**.

InputBuku.java

```
private void BtnSubmitActionPerformed(java.awt.event.ActionEvent evt) {  
  
    try {  
  
        String judul = txtJudul.getText();  
  
        String pengarang = txtPengarang.getText();  
  
        String isbn = txtIsbn.getText();  
  
        String tahunTerbit = txtTahunTerbit.getText();  
  
        String harga = txtHarga.getText();  
  
  
        // Membuat objek Buku baru  
  
        Buku buku = new Buku(judul, pengarang, isbn, tahunTerbit, harga);  
  
  
        // Menambah data buku ke tabel  
  
        DefaultTableModel data = (DefaultTableModel)  
tblBuku.getModel();  
  
        data.addRow(new Object[]{buku.getJudul(),  
buku.getPengarang(), buku.getIsbn(),  
buku.getTahunTerbit(), buku.getHarga()});  
  
  
        // Reset form input  
  
        txtJudul.setText("");  
  
        txtPengarang.setText("");  
  
        txtIsbn.setText("");  

```

```

txtTahunTerbit.setText("");

txtHarga.setText("");

lblKeterangan.setForeground(Color.GREEN);

lblKeterangan.setText("Buku berhasil ditambahkan!");


    } catch (ValidasiInputException e) {

        lblKeterangan.setForeground(Color.RED);

        lblKeterangan.setText("Error: " + e.getMessage());

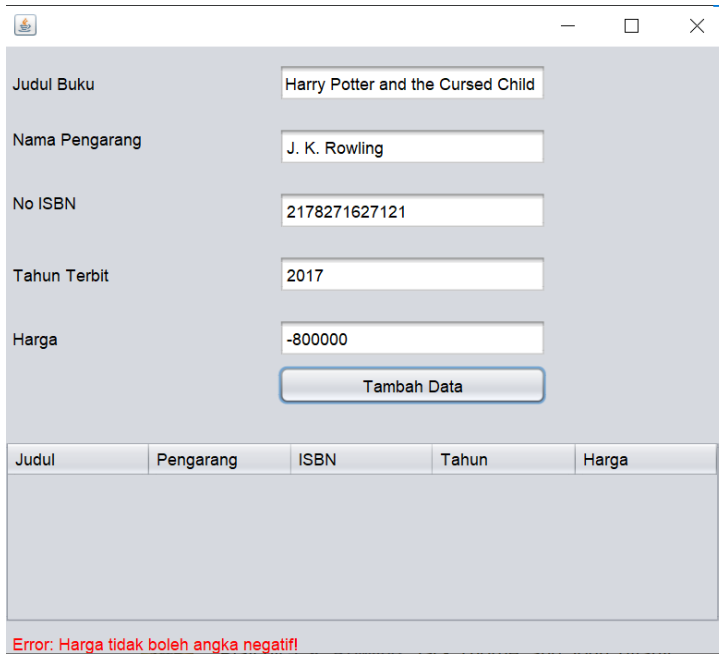
    }

}

```

Penjelasan : Penambahan pada kode di atas adalah atribut baru **harga** yang merepresentasikan harga buku. Konstruktor kini menerima parameter tambahan **harga** dan memvalidasi nilainya. Validasi dilakukan dengan mencoba mengonversi **harga** menjadi tipe **long** menggunakan **Long.parseLong**. Jika nilai **harga** tidak dapat diubah menjadi angka (menimbulkan **NumberFormatException**) atau jika nilainya negatif, maka akan dilemparkan pengecualian **ValidasiInputException** dengan pesan kesalahan yang relevan, seperti "Input harus berupa angka!" atau "Harga tidak boleh angka negatif!". Atribut **harga** juga disimpan ke dalam properti kelas jika validasinya berhasil. Selain itu, metode akses baru **getHarga** ditambahkan untuk memungkinkan pengambilan nilai atribut **harga**. Penambahan ini memastikan bahwa data harga disimpan secara valid dan terstruktur dalam kelas **Buku**.

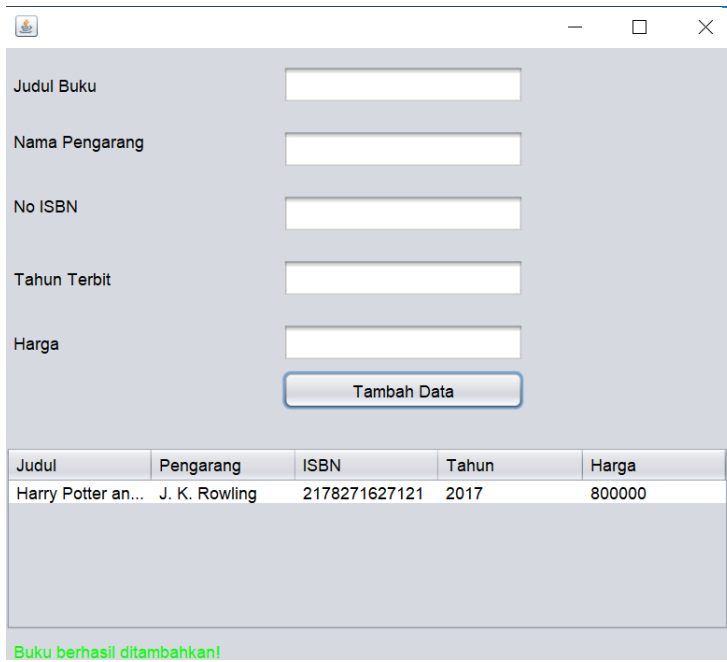
Output



A screenshot of a web application window titled "Output". It contains a form for adding book data. The form has five input fields: "Judul Buku" (filled with "Harry Potter and the Cursed Child"), "Nama Pengarang" (filled with "J. K. Rowling"), "No ISBN" (filled with "2178271627121"), "Tahun Terbit" (filled with "2017"), and "Harga" (filled with "-800000"). Below the inputs is a "Tambah Data" button. Under the button is a table with five columns: "Judul", "Pengarang", "ISBN", "Tahun", and "Harga". The table is currently empty. At the bottom, a red error message reads: "Error: Harga tidak boleh angka negatif!".

Judul	Pengarang	ISBN	Tahun	Harga
-------	-----------	------	-------	-------

Error: Harga tidak boleh angka negatif!



A screenshot of the same web application window after a successful submission. The input fields are now empty. The "Tambah Data" button is still present. The table below now contains one row of data: "Harry Potter an...", "J. K. Rowling", "2178271627121", "2017", and "800000". At the bottom, a green success message reads: "Buku berhasil ditambahkan!".

Judul	Pengarang	ISBN	Tahun	Harga
Harry Potter an...	J. K. Rowling	2178271627121	2017	800000

Buku berhasil ditambahkan!

3. Kategori Buku: Validasi untuk input tidak kosong. (dalam bentuk Dropdown)

Buku.java

```
package Perpustakaan;

/**
 * @author Chris Hotasi
 */
public class Buku {

    private String judul;

    private String pengarang;

    private String isbn;

    private String tahunTerbit;

    private String harga;

    private String kategori;

    public Buku(String judul, String pengarang, String isbn, String tahunTerbit,
String harga, String kategori) throws ValidasiInputException {

        if (judul.isEmpty()) {

            throw new ValidasiInputException("Judul buku tidak boleh kosong!");

        }

        if (pengarang.isEmpty()) {

            throw new ValidasiInputException("Pengarang tidak boleh kosong!");

        }

    }

}
```

```
}

if (!isbn.matches("\\d{13}")) {

    throw new ValidasiInputException("ISBN harus 13 digit angka!");

}

if (!tahunTerbit.matches("\\d{4}")) {

    throw new ValidasiInputException("Tahun Terbit harus 4 digit angka!");

}

try {

    long hargaNumber = Long.parseLong(harga);

    if (hargaNumber < 0) {

        throw new ValidasiInputException("Harga tidak boleh angka negatif!");

    }

} catch (NumberFormatException e) {

    throw new ValidasiInputException("Input harus berupa angka!");

}

if (kategori == null || kategori.isEmpty()) {

    throw new ValidasiInputException("Kategori harus dipilih!");

}


this.judul = judul;

this.pengarang = pengarang;

this.isbn = isbn;

this.tahunTerbit = tahunTerbit;
```

```
this.harga = harga;
```

```
this.kategori = kategori;
```

```
}
```

```
public String getJudul() {
```

```
    return judul;
```

```
}
```

```
public String getPengarang() {
```

```
    return pengarang;
```

```
}
```

```
public String getIsbn() {
```

```
    return isbn;
```

```
}
```

```
public String getTahunTerbit() {
```

```
    return tahunTerbit;
```

```
}
```

```
public String getHarga() {
```

```
    return harga;
```

```

    }

    public String getKategori() {

        return kategori;

    }

}

```

Penjelasan : Penambahan pada kode di atas adalah atribut baru **kategori**, yang merepresentasikan kategori buku, serta validasinya di dalam konstruktor. Konstruktor kini menerima parameter tambahan **kategori** dan memeriksa apakah nilai **kategori** kosong atau **null**. Jika validasi gagal, akan dilemparkan pengecualian **ValidasiInputException** dengan pesan "Kategori harus dipilih!". Selain itu, atribut **kategori** juga disimpan ke dalam properti kelas jika validasi berhasil. Untuk memungkinkan akses terhadap nilai **kategori**, metode **getKategori** ditambahkan. Penambahan ini memungkinkan pengelolaan data kategori buku secara terstruktur dan memastikan bahwa semua data yang penting sudah divalidasi sebelum digunakan.

InputBuku.java

```

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

    try {

        String judul = txtJudul.getText();

        String pengarang = txtPengarang.getText();

        String isbn = txtIsbn.getText();

        String tahunTerbit = txtTahunTerbit.getText();

        String harga = txtHarga.getText();

        String kategori = (String) cbxKategori.getSelectedItem();
    }
}

```



```
// Membuat objek Buku baru

Buku buku = new Buku(judul, pengarang, isbn, tahunTerbit, harga, kategori);


// Menambah data buku ke tabel

DefaultTableModel data = (DefaultTableModel)
tblBuku.getModel();

data.addRow(new Object[]{buku.getJudul(),
buku.getPengarang(), buku.getIsbn(),
buku.getTahunTerbit(), buku.getHarga(),
buku.getKategori()});


// Reset form input

txtJudul.setText("");

txtPengarang.setText("");

txtIsbn.setText("");

txtTahunTerbit.setText("");

txtHarga.setText("");

cbxKategori.setSelectedIndex(0);

lblKeterangan.setForeground(Color.GREEN);

lblKeterangan.setText("Buku berhasil ditambahkan!");


} catch (ValidasiInputException e) {

    lblKeterangan.setForeground(Color.RED);
```

```

        lblKeterangan.setText("Error: " + e.getMessage());
    }
}

```

Penjelasan : Penambahan terkait dengan **kategori** di kode di atas melibatkan pengambilan nilai **kategori** dari elemen combo box **cbxKategori**. Nilai yang dipilih oleh pengguna di combo box diakses menggunakan **getSelectedItem()**, yang mengembalikan objek yang dipilih (dalam hal ini, kategori buku). Nilai ini kemudian diteruskan sebagai parameter tambahan saat membuat objek **Buku**. Selain itu, setelah data berhasil ditambahkan ke tabel, pilihan kategori di-reset dengan **setSelectedIndex(0)**, yang mengembalikan combo box ke pilihan pertama (biasanya berupa default atau kosong). Penambahan ini memastikan bahwa data kategori buku ditangani dengan benar dalam proses input dan penyimpanan ke dalam tabel.

Output

The screenshot shows a Java Swing application window titled "Tambah Data". It contains a form with the following fields:

- Judul Buku: Harry Potter and the Cursed Child
- Nama Pengarang: J. K. Rowling
- No ISBN: 2167261762711
- Tahun Terbit: 2017
- Harga: 800000
- Kategori: (dropdown menu)

Below the form is a button labeled "Tambah Data". Below the button is a table with the following columns: Judul, Pengarang, ISBN, Tahun, Harga, and Kategori. The table is currently empty.

At the bottom of the window, a red error message is displayed: "Error: Kategori harus dipilih!".

Judul Buku

Nama Pengarang

No ISBN

Tahun Terbit

Harga

Kategori

Tambah Data

Judul	Pengarang	ISBN	Tahun	Harga	Kategori
Harry Potter...	J. K. Rowling	216726176...	2017	800000	Novel

Buku berhasil ditambahkan!