PROGRAM STUDI INFORMATIKA
UNIVERSITAS SANATA DHARMA

PRAKTIKUM PBO LANJUT
EVENT HANDLING

A. TUJUAN PRAKTIKUM

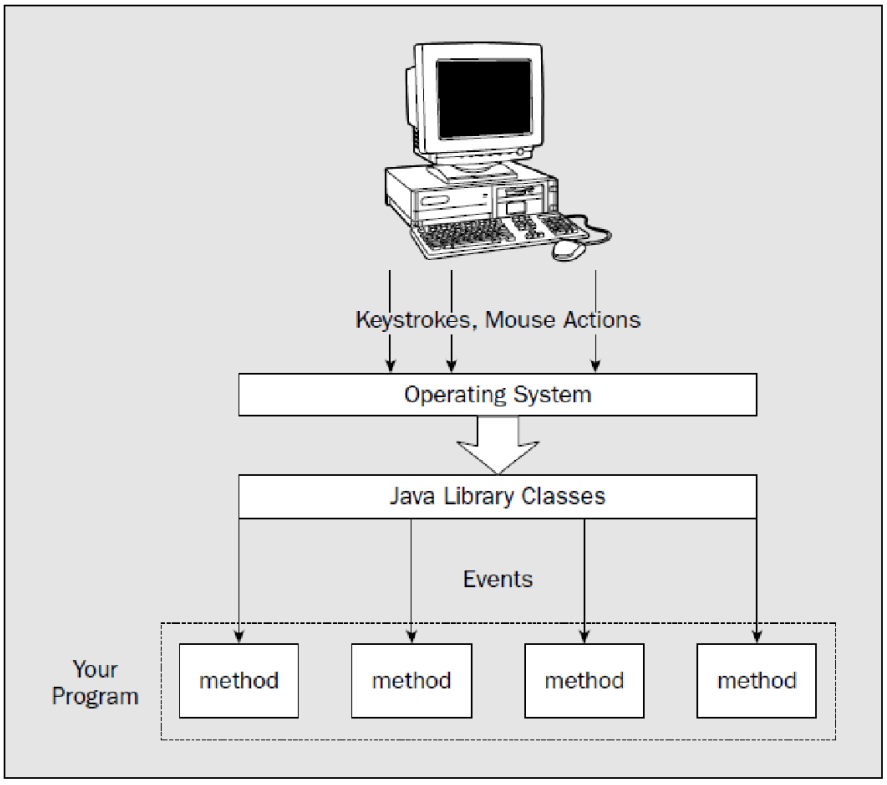
1. Mahasiswa mampu mengaplikasikan event handling .

B. MATERI PRAKTIKUM

1. Event

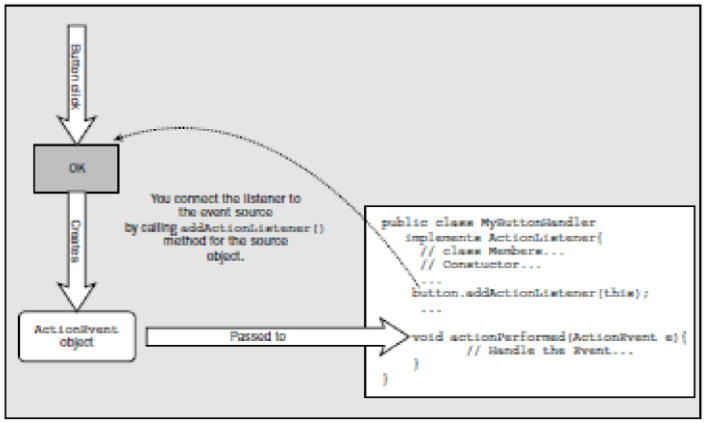
Pada program dengan menggunakan GUI, maka operasi program dikendalikan dari tampilan GUI tersebut. Dengan memilih item menu atau button menggunakan atau melalui keyboard akan menyebabkan kejadian tertentu pada program.

Ketika sebuah aksi dilakukan (misal, klik item menu, menggerakkan mouse dst) terhadap program berbasis window, maka aksi tersebut pertama-tama akan dikenali oleh sistem operasi. Sistem operasi akan mencatat program yang saat ini sedang berjalan dan melewatkan aksi tersebut ke program. Ketika user me-klik button mouse, maka SO akan mencatat posisi kursor mouse pada layar, kemudian menentukan aplikasi yang mengendalikan window dimana user menekan button mouse dan mengkomunikasikan penekanan button mouse tersebut ke program. Sinyal yang diterima program dari SO sebagai sebuah hasil dari aksi user disebut **events**. Komunikasi antara aksi dan events digambarkan sebagai berikut :



Sebuah program tidak harus merespon semua event, misal jika pergerakan mouse tidak akan menyebabkan sebuah aksi, maka event tersebut akan dihapus. Sebuah program berbasis window disebut **event-driven program** sebab sejumlah aksi dilakukan user sebagai interaksi

dengan layar GUI akan mengakibatkan terjadinya event yang dapat ditangani (*driven*) oleh program Java.
Jika sebuah event terjadi, maka Java akan melakukan mekanisme untuk memproses event, hal ini disebut **event handling**. Jika sebuah obyek JButton dikenai operasi click oleh user, maka obyek tersebut akan me-generate **action event**. Setelah event di-generate, system akan mencari obyek **event listener** yang relevan. Event listener adalah sebuah obyek yang memiliki method dan akan dieksekusi sebagai respons dari event yang digenerate. Event listener merupakan sebuah interface (apa itu interface ?).



Terdapat beberapa tipe event, diantaranya action event, window event, mouse event. Untuk setiap tipe event harus terdapat listener yang berkaitan secara langsung dengan event tersebut.
Terdapat beberapa cara untuk mengimplementasikan interface listener :

1. Listener dengan Satu Inner-Class

Cara ini akan membuat *inner-class* di dalam *main-class*. *Inner-class* ini akan mengimplementasikan interface *listener*.
Misalnya program berikut ini akan "mendengarkan" dengan obyek interface ActionListener. Dan "menangkap" ActionEvent untuk diolah dalam *event-handler* actionPerformed().

Cara ini, disatu sisi memiliki keuntungan, karena semua penanganan event dipusatkan dalam satu *method*. Namun disisi lain, cara seperti ini juga kurang praktis jika kita harus menangani banyak macam *event*. Jika demikian, maka akan banyak sekali interface yang harus diimplementasikan oleh *inner-class*.

2. Listener dengan Main Class

Cara ini akan memakai *keyword* "implements" pada *main-class*.
Misalnya program berikut ini akan "mendengarkan" dengan memakai interface ActionListener. Dan "menangkap" ActionEvent untuk diolah dengan *event-handler* actionPerformed().

Cara seperti ini kurang praktis, jika kita harus menangani banyak macam *event*. Jika demikian, maka akan banyak sekali interface yang harus diimplementasikan oleh *main-class*.

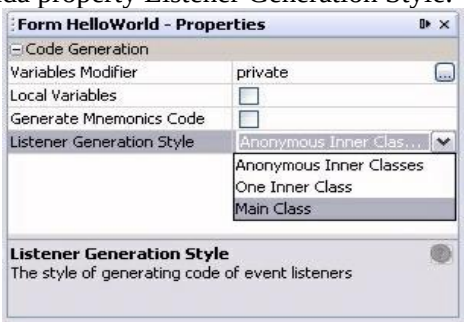
3. Listener dengan Anonymous Inner-Class

Cara ini akan membuat obyek dari interface *listener* secara anonim (dengan *anonymous-inner-class*). Misalnya program berikut ini akan "mendengarkan" dengan memakai obyek interface ActionListener anonim. Dan "menangkap" ActionEvent untuk diolah dalam *event-handler* actionPerformed().

Dengan cara ini, jika ada beberapa *listener* yang diperlukan, akan dapat ditambahkan dengan cara yang lebih terstruktur.
Namun cara ini membuat penanganan *event* menjadi terpecah dan terpisah dalam banyak bagian program. Misalnya, jika ada 10 komponen, dan setiap komponen harus menangani event

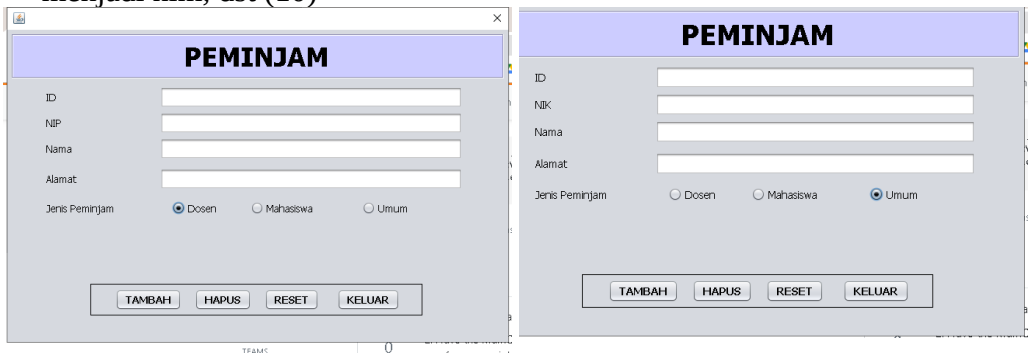
masing-masing. Maka akan ada pembuatan *anonymous inner-class* secara terpisah dalam 10 bagian program.

Terlepas dari kelebihan dan kekurangan dari ketiga cara di atas, Anda dapat memilih salah satu yang paling cocok untuk dipakai dalam program Anda.
Dalam IDE NetBeans, kita dapat menentukan cara mana yang akan dipakai dalam jendela Properties dari form. Yaitu pada property Listener Generation Style.

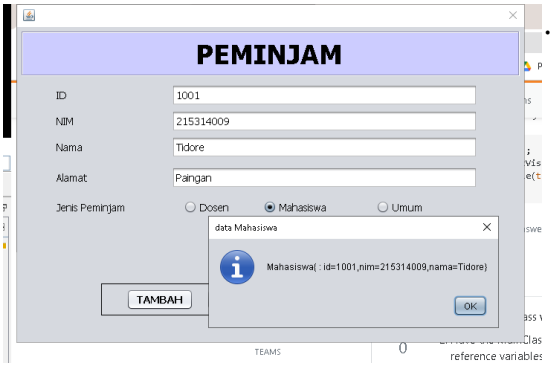


C. LANGKAH PRAKTIKUM

1. Berdasar kelas GUI pertemuan II, *handle*-lah menu (Edit Peminjam, Edit Koleksi, Transaksi Pinjam, Transaksi Kembali, serta menu Help) sehingga ketika menu dipilih akan menampilkan setiap JDialog yang bersesuaian yang dibuat di pertemuan II. (5 x @5)
2. Dalam kelas JDialog untuk Edit Peminjam:
   a. Gunakan JCombobox atau JRadioButton untuk memberikan pilihan mahasiswa/dosen/umum. Ketika dipilih mahasiswa, maka nomor akan berganti menjadi nim, dst (10)



   b. Untuk JButton "Tambah" ketika di-klik akan: (20)
      - Membuat obyek dari kelas Peminjam menyesuaikan apakah Mahasiswa/Dosen/Umum, kemudian mengeset data yang dimasukkan ke dalam obyek yang dibuat
      - menampilkan MessageDialog yang menampilkan data dari obyek kelas Peminjam



c. Kemudian buat juga JButton "Reset", yang akan mengosongkan semua JTextField. (5)
d. Beri JButton "Keluar" yang akan menutup JDialog. (5)

3. Lakukan seperti nomor 2 untuk kelas JDialog menambah data Koleksi.
   a. Untuk JButton "Tambah" akan berlaku sama seperti diatas. (20)
   b. Jenis koleksi (JcomboBox/JRadioButton) – Field selanjutnya akan berubah sesuai pilihan (3 x @5)
      i. Disk :
         - Format (JTextField)
         - ISBN (JTextField)
      ii. Majalah :
         - Volume (JTextField)
         - Seri (JTextField)
         - ISSN (JTextField)
      iii. Buku :
         - Halaman (JTextField)
         - ISBN (JTextField)

D. LAPORAN PRAKTIKUM

1. Capture GUI
2. Kode program yang bersesuaian dan analisa setiap baris program yang penting dan sesuai materi

++++++++