THCS 4: LẬP TRÌNH CƠ BẢN VỚI JAVA

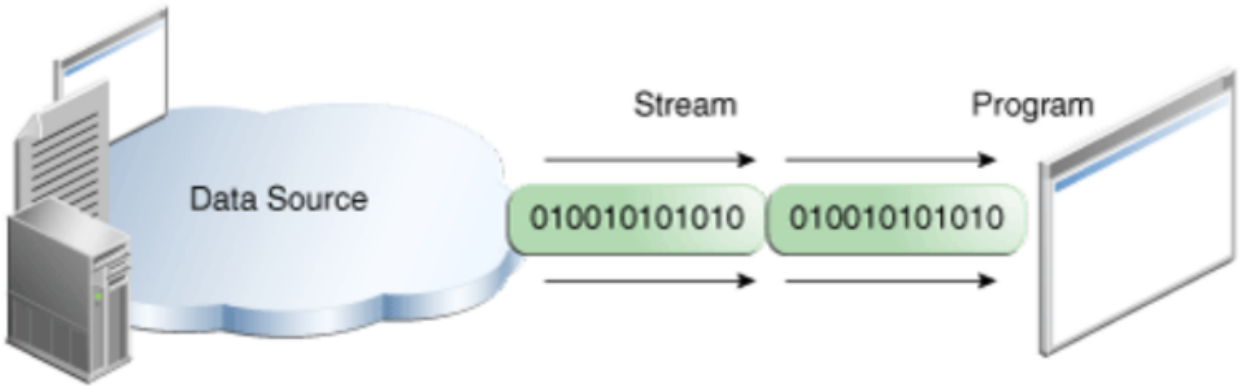
THƯ VIỆN I/O - THƯ VIỆN ĐỒ HOẠ

Nguyễn Thị Tâm
nguyenthitam.hus@gmail.com

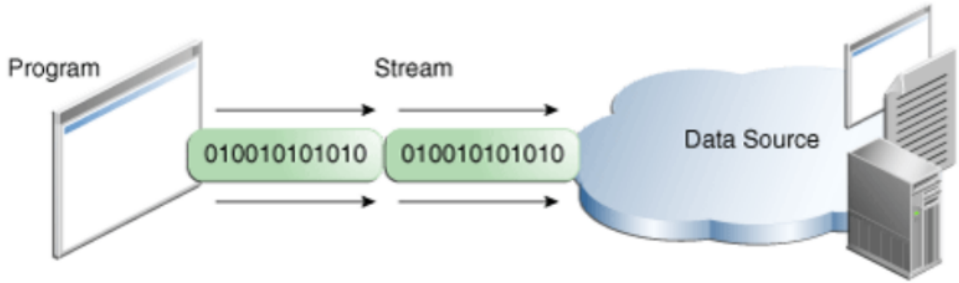
Ngày 1 tháng 12 năm 2024

Vào/Ra (I/O)

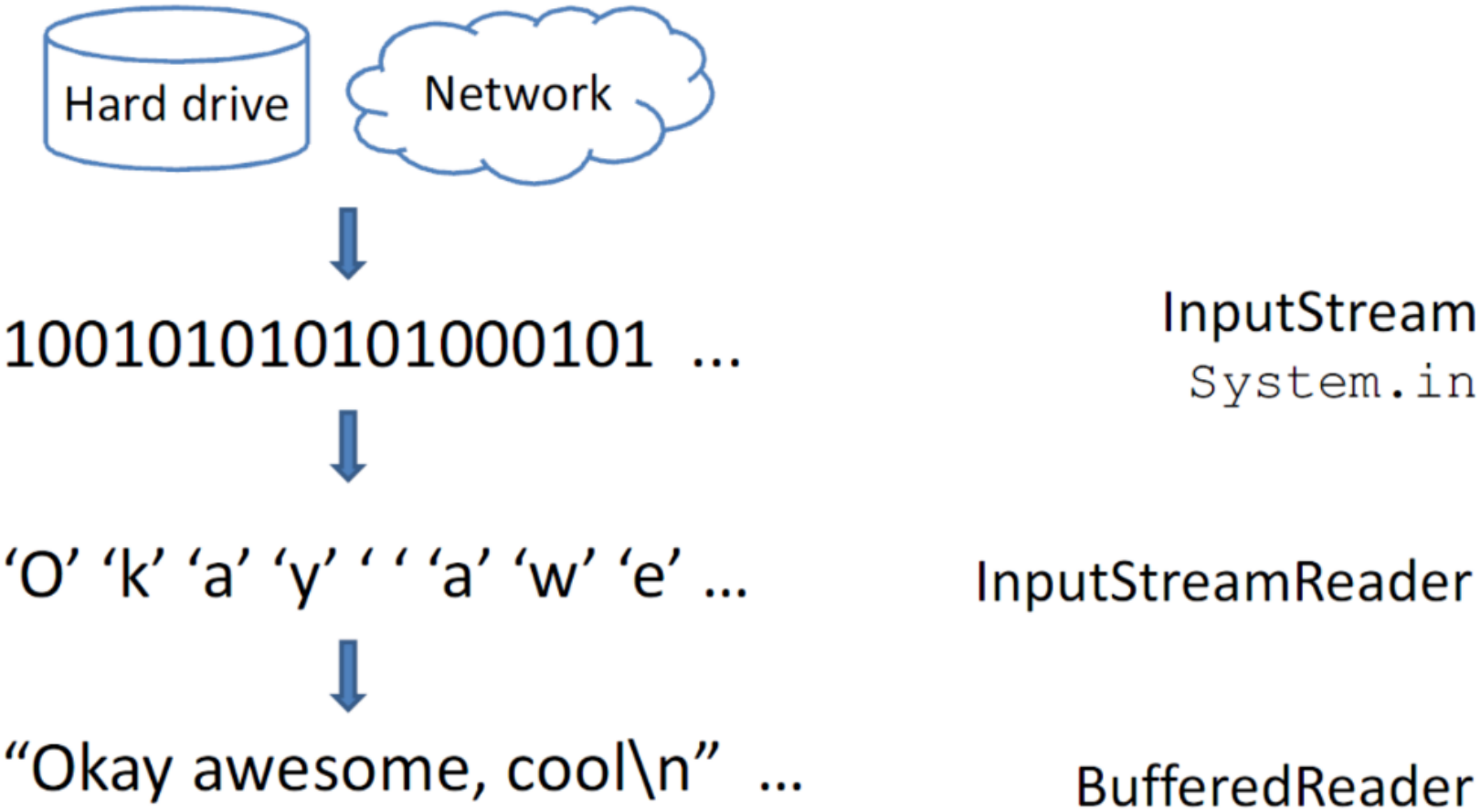
- Vào



- Ra



Ví dụ quản lý luồng kí tự



Luồng vào/ra (I/O) Stream

- Biểu diễn nguồn vào hoặc đích ra của dữ liệu trong chương trình
- Luồng có thể biểu diễn rất nhiều kiểu nguồn và đích: tệp trên đĩa, thiết bị ngoại vi, chương trình khác, bộ nhớ
- 2 kiểu tệp:
  - Tệp nhị phân (*binary file*): tạo từ luồng dữ liệu byte
  - Tệp văn bản (*text file*): tạo từ luồng dữ liệu kí tự

Xử lý tệp trong Java (1)

- Mở tệp
  - Tạo 01 đối tượng luồng gắn với một luồng byte hoặc luồng kí tự
  - Luồng cũng có thể được gắn với các thiết bị khác nhau.
  - Chương trình java tạo 03 đối tượng luồng khi bắt đầu thực hiện
    - System.in: Ngầm định gắn với bàn phím,
    - System.out, System.err: ngầm định gắn với màn hình
    - Có thể chuyển hướng các luồng này (gắn với thiết bị khác như tệp trên đĩa, đường truyền mạng, vv) bằng các phương thức tĩnh setIn, setOut, setErr của lớp System

Xử lý tệp trong Java (2)

- Xử lý tệp: Sử dụng các lớp trong gói java.io, ví dụ
  - Các luồng dữ liệu từ tệp
    - Luồng byte: FileInputStream/FileOutputStream, kế thừa các lớp tương ứng InputStream/OutputStream
    - Luồng kí tự: FileReader/FileWriter, kế thừa các lớp tương ứng Reader/Writer
  - Các luồng xử lí vào/ra các chuỗi dữ liệu đối tượng hoặc dữ liệu nguyên thủy: ObjectInputStream, ObjectOutputStream
- Việc vào/ra dữ liệu kí tự còn có thể được thực hiện nhờ các lớp java.util.Scanner/Formatter

Sử dụng các lớp trong gói java.io

Ví dụ

```
import java.io.*;
public class CopyFile {
    public static void main(String args[]) throws IOException{
        FileInputStream in = null;
        FileOutputStream out = null;
        in = new FileInputStream("input.txt");
        out = new FileOutputStream("output.txt");
        int c;
        while ((c = in.read()) != -1) {
                out.write(c);
        }
        in.close();
        out.close();
    }
}
```

Xử lý tệp trong Java

- Lấy thông tin về tệp/thư mục: tạo đối tượng thuộc lớp File
  - boolean canRead(), canWrite(), canExecute(), exists(), isFile(), isDirectory(), isAbsolute()
  - String getAbsolutePath(), getName(), getPath(), getParent()
  - long length(), lastModified()
  - String[] list(): mảng chứa các tên tệp/thư mục trong đối tượng thư mục hiện thời
  - ....
  - Ví dụ: FileDemonstration.java

Ví dụ

```
public void analyzePath( String path ) {
    File name = new File( path );
    if ( name.exists() ){
        System.out.printf("%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n%s",name.g
            name.isFile(), name.isDirectory(), name.isAbsolute(),
                    name.lastModified(), name.length(),
                     name.getPath(), name.getAbsolutePath(),
                    name.getParent() );
        if ( name.isDirectory() ){
            String directory[] = name.list();
            System.out.println( "\n\nNoi dung thu
            muc:\n" );
            for ( String directoryName : directory )
            System.out.printf( "%s\n",
            directoryName );
        }
    }
}
```

Tệp văn bản truy cập tuần tự

- Tạo tệp văn bản truy cập tuần tự: CreateTextFile.java
- Đọc tệp văn bản tuần tự: ReadTextFile.java
- Cập nhật tệp văn bản tuần tự: Ghi nội dung cập nhật vào tệp mới

Tạo tệp văn bản truy cập tuần tự (1)

Ví dụ

```
public class CreateTextFile{
    // object used to output text to file
    private Formatter output;
    public void openFile(){
    // commands
    }
    // add records to file
    public void addRecords(){
    // commands
    }
    public void closeFile(){
        if (output != null)
        output.close();
    }
} // end class CreateTextFile
```

Tạo tệp văn bản truy cập tuần tự (2)

Ví dụ

```
// private Formatter output;
public void addRecords(){
    AccountRecord record = new AccountRecord();
    Scanner input = new Scanner( System.in );
    int count = 5;
    for(int i; i< count; i++){
        record.setAcount( input.nextInt() );
        record.setFirstName( input.next() );
        record.setLastName( input.next() );
        record.setBalance( input.nextDouble() );
        output.format( "%d %s %s %.2f\n",
        record.getAccount(), record.getFirstName(),
        record.getLastName(), record.getBalance() );
// FormatterClosedException: Error writing to file
// NoSuchElementException: Invalid input
    }
}
```

Đọc tệp văn bản truy cập tuần tự (1)

Ví dụ

```
    // private Formatter output;
public void readRecords(){
    AccountRecord record = new AccountRecord();
    while ( input.hasNext() ){
        record.setAcount( input.nextInt() );
        record.setFirstName( input.next() );
        record.setLastName( input.next() );
        record.setBalance( input.nextDouble() );
        System.out.printf( "%-10d%-12s%-12s%10.2f\n",
        record.getAccount(), record.getFirstName(),
            record.getLastName(), record.getBalance() );
    // IllegalStateException: error reading from file
    // NoSuchElementException: File improperly formed
    }
}
```

Tệp nhị phân chứa chuỗi các đối tượng

- Lưu trữ, đọc dữ liệu kiểu đối tượng vào tệp nhị phân:
  - Điều kiện: đối tượng khả tuần tự hóa (serializable). Cài đặt giao diện Serializable (AccountRecordSerializable.java)
  - Ghi đối tượng vào tệp tuần tự: Tuần tự hóa đối tượng (object serialization). Ví dụ tệp CreateSequentialFile.java
  - Đọc đối tượng từ tệp tuần tự: Giải tuần tự hóa đối tượng (object deserialization) Ví dụ: ReadSequentialFile.java

Lưu trữ, đọc dữ liệu kiểu đối tượng trên tệp nhị phân (1)

Cài đặt giao diện Serializable (AccountRecordSerializable.java)

```
import java.io.Serializable;
public class AccountRecordSerializable implements
Serializable{
    // fields: account, firstName, lastName, balance
    public AccountRecordSerializable(){
        this(0, "", "", 0.0);
    }
    public AccountRecordSerializable(int acct, String first, String
         last, double bal ){
        setAccount( acct );
        setFirstName( first );
        setLastName( last );
        setBalance( bal );
    }
    // setAccount(), setFirstName(), setLastName(),
    setBalance( double bal ),...
}
```

Lưu trữ, đọc dữ liệu kiểu đối tượng trên tệp nhị phân (2)

Ghi đối tượng vào tệp tuần tự: Tuần tự hoá đối tượng (*object serialization*)

```
public class CreateSequentialFileTest{
    public static void main( String args[] ){
        CreateSequentialFile application = new
        CreateSequentialFile();
        application.openFile();
        application.addRecords();
        application.closeFile();
    } // end main
} // end class CreateSequentialFileTest
```

Lưu trữ, đọc dữ liệu kiểu đối tượng trên tệp nhị phân (3)

Đọc đối tượng từ tệp tuần tự: Giải tuần tự hóa đối tượng (object deserialization)

```
    public void readRecords(){
        AccountRecordSerializable record;
        System.out.printf( "%-0s%-s%-s%0s\n", "Account",
        "First Name", "Last Name", "Balance" );
        while ( true ){
            record = ( AccountRecordSerializable )
            input.readObject();
            System.out.printf( "%-0d%-s%-s%0.f\n",
            record.getAccount(), record.getFirstName(),
            record.getLastName(), record.getBalance() );
        }
    }
// ClassNotFoundException: Unable to create object
// IOException : Error during read from file
// EOFException: end of file was reached
```

Tệp truy cập ngẫu nhiên

- Cách đơn giản: tạo các đối tượng dữ liệu có cùng kích thước: RandomAccessAccountRecord.java
- Tạo tệp dữ liệu: WriteRandomFile.java
- Đọc dữ liệu từ tệp: ReadRandomFile.java

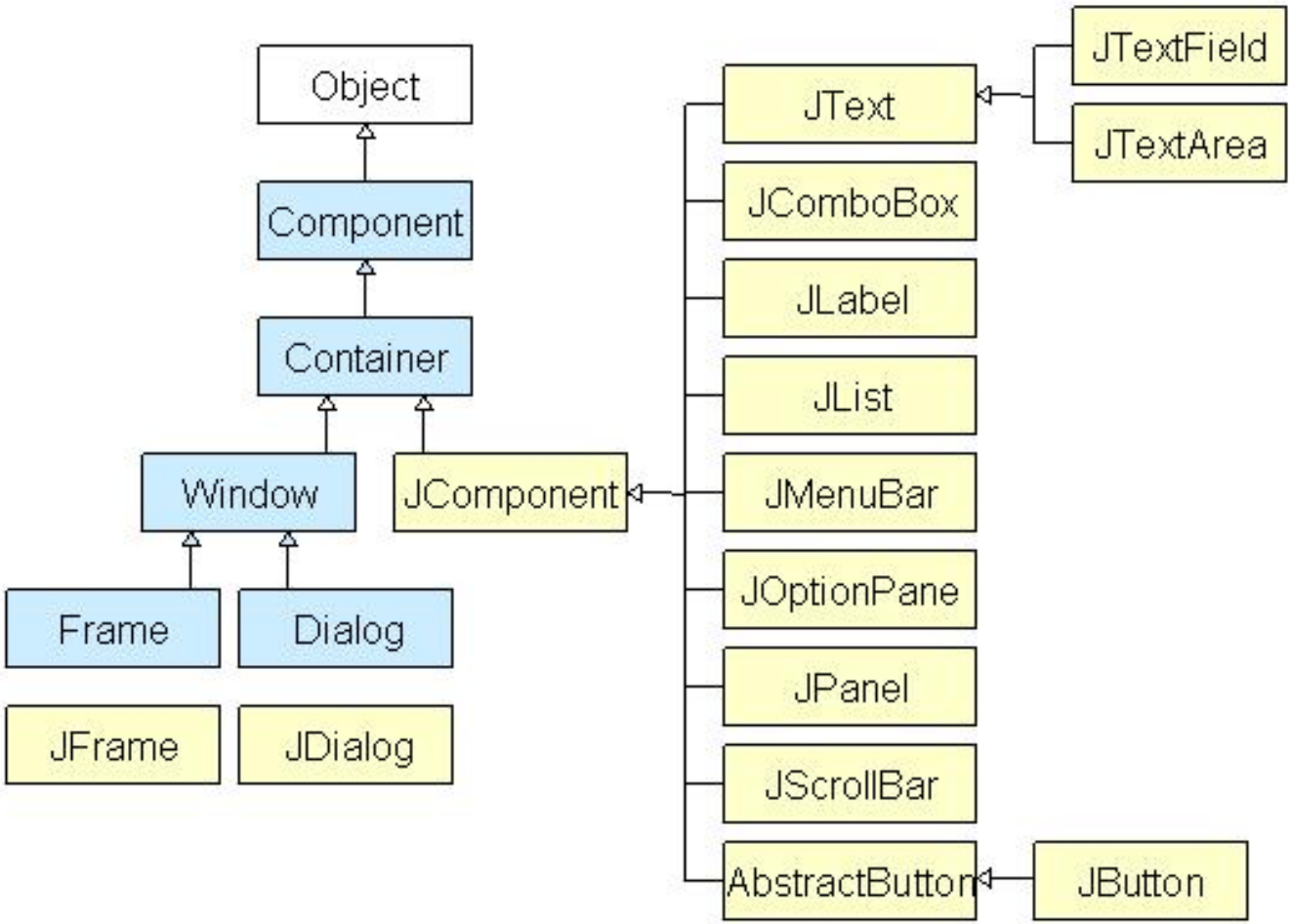
Giao diện chọn tệp

- FileChooserDemonstration.java

Thư viện đồ hoạ Java

- 2 gói: java.awt và javax.swing
- AWT: Abstract Window Toolkit
  - Hiển thị giao diện phù hợp với nền hệ điều hành (Windows, Linux, Mac OS)
  - Nặng nề và kém linh hoạt vì gắn chặt với hình thức và cách hoạt động (look and feel) của giao diện nền máy
- Swing: Ra đời từ Java 1.2
  - Hình thức và cách hoạt động độc lập với nền máy
  - Linh hoạt, được ưa chuộng hơn AWT.

Cây kế thừa từ các thành phần Swing



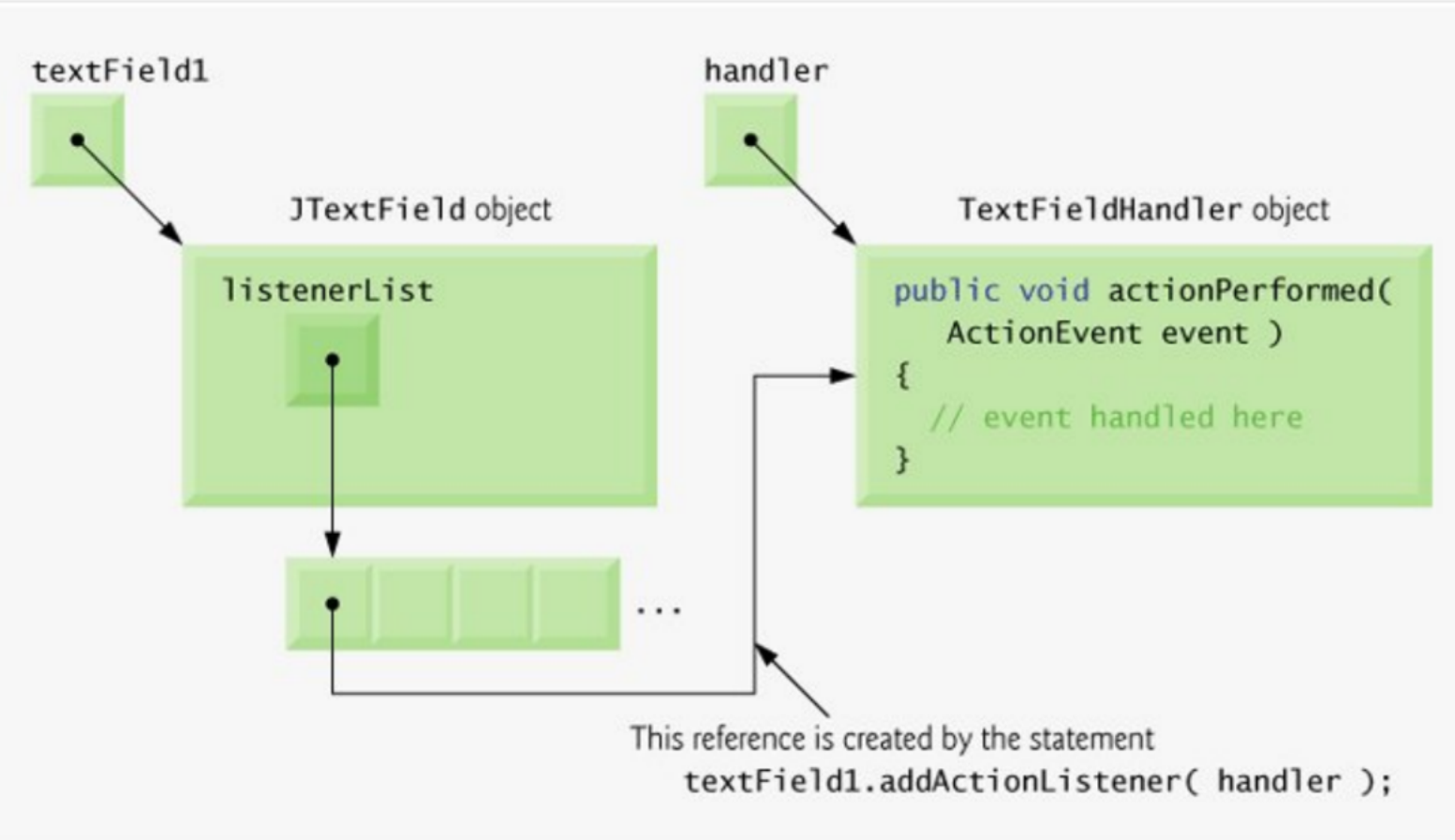
Một số thành phần Swing cơ bản

- JLabel: hiển thị nhãn văn bản hoặc biểu tượng
- JTextField: vùng văn bản soạn thảo được
- JButton: nút cho phép thực hiện 1 sự kiện khi có tín hiệu kích chuột
- JCheckbox: hộp chọn cho phép chọn/bỏ lựa chọn
- JComboBox: hộp chọn thả xuống
- JList: danh sách chọn (cho phép chọn nhiều mục)
- JPanel: vùng cho phép đặt các thành phần đồ hoạ.

Ví dụ minh hoạ

- LabelFrame.java, LabelTest.java
- TextFieldFrame.java, TextFieldTest.java

Quản lý sự kiện



Bố trí các thành phần đồ hoạ

- Quản lí bài trí: Giao diện LayoutManager: Sử dụng hàm setLayout(LayoutManager mgr)
- Các cách thiết lập bài trí
  - Định vị tuyệt đối: Người lập trình xác định vị trí tuyệt đối và kích thước của từng thành phần trong cửa sổ
  - Sử dụng LayoutManger đã cài đặt trong thư viện Java, VD: FlowLayout, GridLayout.
  - Sử dụng IDE hỗ trợ lập trình thị giác (visual programming):
  - Kéo/thả, thay đổi kích thước các thành phần trong cửa sổ. Công cụ IDE sinh tự động mã chương trình tương ứng

Các ví dụ minh hoạ

- Thư mục ví dụ Graphics
- Đọc danh sách các ví dụ trong tệp README.txt