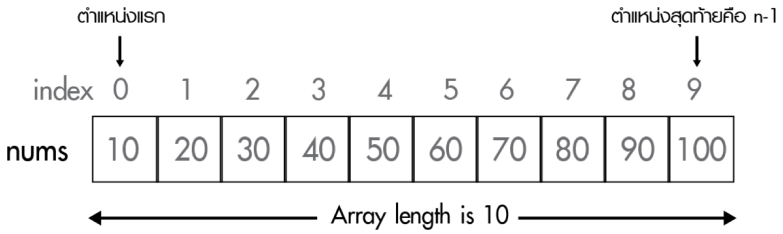
INT105 Programming II (FINAL)

Outlines

0. Adv.Concept OOP 1. Array 2. Exception 3. JDBC 4. GUI 5. Java I/O

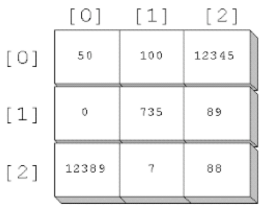
Array

1. Array 1 มิติ



int [ ] nums = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};

2. Array 2 มิติ




```
scores[0][0] = 50;
scores[0][1] = 100;
scores[0][2] = 12345;
scores[1][0] = 0;
scores[1][1] = 735;
scores[1][2] = 89;
scores[2][0] = 12389;
scores[2][1] = 7;
scores[2][2] = 88;
```

- การประกาศ:

  int [ ] [ ] scores = new int [row] [column];

3. ArrayList

```
ArrayList <DataType> myArrList = newArrayList <DataType>();

ArrayList <Student> stdList = newArrayList <Student>();
```

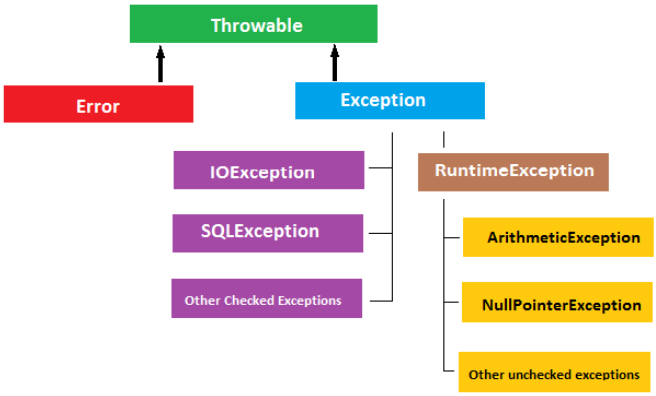
Method

- เพิ่ม : myArrList.add(“text”);
- ขนาด : myArrList.size();
- ดึงข้อมูล : myArrList.get(index);

หรือใช้ for each เพื่อแสดงข้อมูล

```
for (int show : myArrList){
      System.out.println(show);
}
```

Exception



- ต้นตระกูลคือ Throwable
- Error เป็นข้อผิดพลาดร้ายแรงที่เราไม่สามารถจัดการได้
- Exception เป็นความผิดพลาดที่แก้ไขได้
- **Runtime Exception (Unchecked Exception):**
  >ยอมให้ Developer ไม่ต้องจัดการได้
- **Other Exception (Checked Exception):**
  > ต้องจัดการ Exception เองเช่น IO, SQL

- ตัวอย่างใน Class

Class ArithmeticException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.lang.RuntimeException
        java.lang.ArithmeticException

Class IndexOutOfBoundsException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.lang.RuntimeException
        java.lang.IndexOutOfBoundsException

Class SQLException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.sql.SQLException

HOW TO HANDLER EXCEPTION ?

1. Try - Catch Block

```
try{
        line1
        line2   //เกิดปัญหา
        line3
}
catch(ArithmeticException ae){
        System.out.println(ae);
}
catch(RuntimeException re){
        System.out.println(re);
}
finally{
        line4
        line5
}
line 6
line 7
```

- **Try** 1 blocks/ ใส่ปัญหาที่น่าจะเกิด ควรใส่ทั้งหมดไม่ใช่แค่บริเวณบรรทัดนั้นเพราะอาจจะมีบรรทัดอื่นอ้างอิงและเรียกใช้งาน
- **Catch** (Optional) n blocks/จะเข้าโดยเรียงจากบนลงล่าง : Subclass ------> Class
  เช่น Arithmetic ------> RunTime
  **เหมือนเป็นตัวจัดการกับ Exception
- **Finally** (Optional) 1 blocks /จะถูกทำไม่ว่าจะเกิด Exception หรือไม่ เช่น ปิด Connection in DataBase
- Optional ต้องมีอย่างน้อย 1 อัน

กรณี 1 try-catch

1.1. ถ้าไม่มี exception
  TRYครบ-----> ข้าม catch----->ทำปกติ
1.2.ถ้ามี exception และหาเจอใน catch
  TRY ทำถึงบรรทัดที่เจอปัญหา ----->catch----->ทำปกติ
1.3.ถ้ามี exception แต่หาไม่เจอใน catch
  TRY ทำถึงบรรทัดที่เจอปัญหา ----->หาไม่เจอในcatch----->จบการทำงาน

กรณี 2 try-catch-finally

1.1. ถ้าไม่มี exception
  TRYครบ-----> ข้าม catch----->FINAL----->ทำปกติ
1.2.ถ้ามี exception และหาเจอใน catch
  TRY ทำถึงบรรทัดที่เจอปัญหา ----->catch----->FINAL----->ทำปกติ
1.3.ถ้ามี exception แต่หาไม่เจอใน catch
  TRY ทำถึงบรรทัดที่เจอปัญหา ----->หาไม่เจอ----->FINAL

2. Throw Exception

- เนื่องจาก caller (เรียกใช้) แต่ละคนอาจจัดการคนละรูปแบบ เราไม่ควรจัดการไว้ที่ methods ให้เพิ่ม throws ตามด้วยชื่อปัญหา ต่อท้าย method ที่มีความเสี่ยง เช่น

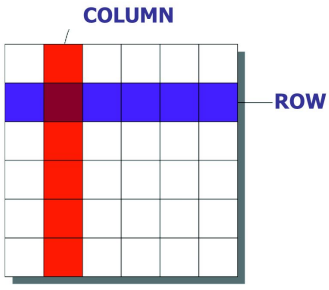
```
public static double methodName() throws ArithmeticException{

}
```

ตัวอย่างการสร้าง Throw

```
public class NegativeDividerException extends Exception{
    public NegativeDividerException(String message) {
        super(message);
    }
}
```

```
public static double divideByzero() throws ArithmeticException, NegativeDividerException {
    if(divider<0){
        throw new NegativeDividerException("Negative Divider");
    }
}
```

JDBC



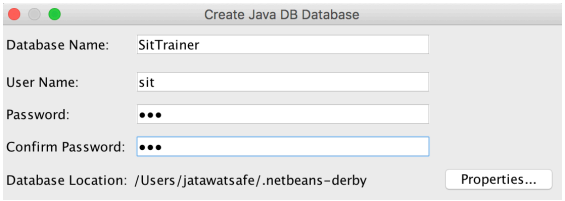
- ฐานข้อมูล (Database) อยู่ในระบบ DBMS (Database Management System)
- ใน 1 DB มี n tables โดยแต่ละตารางมี Relationship กัน
- 1 rows = 1 record
- Primary Key คือ ข้อมูลที่มีความ Unique เช่น id (unique)
- Foreign Key คือ interkey ใช้เชื่อมตารางเพื่อให้ข้อมูลเป็นข้อมูลเดียวกัน

เช่น

StdId (PrimaryKey)	stdName	stdLastName	DeptId (Foreign Key)
601301	AA	aa	2
601302	BB	bb	1

DeptId (Foreign Key)	DeptName
1	IT
2	Fibo

Create Database



- Database Name: SitDB
- User Name: sit
- Password: sit

SQL statement

1. สร้างตาราง

```
create table tableName (
name varchar(50), number integer, primary key(name)
);
```

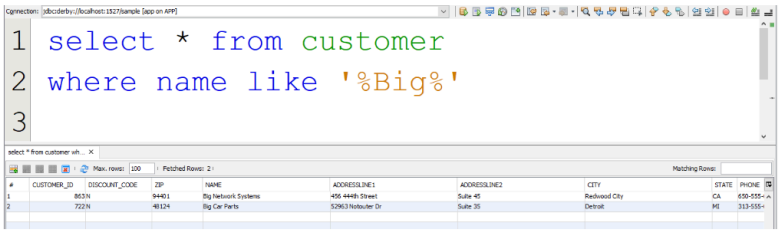
2. ดูข้อมูลลงตาราง

```
SELECT COLUMN FROM TABLE
```

- where (เพื่อ ‘ระบุ’)

ค้นหา Pattern โดยใช้ **LIKE**

```
1 select * from customer
2 where name like '%Big%'
3
```



- ใช้ % เป็น Wildcards เช่น A% คำที่ขึ้นต้นด้วย A

3. เพิ่มข้อมูลลงตาราง

```
INSERT INTO table (column1, column2, column3, ...)
VALUES (value1, value2, value3,...)
```

```
1 insert into Customer(customer_id, name, discount_code, zip)
2 values (954595,'Karakate','M','10095')
```

4. แก้ไขข้อมูล

```
UPDATE ชื่อตาราง  SET ข้อมูลอะไรบ้าง  WHERE ข้อมูลที่ไหน
```

- เช่น **UPDATE** customer **SET** zip='10095' **WHERE** city='New York'

5. เพื่อลบข้อมูล

```
DELETE FROM ชื่อตาราง
```

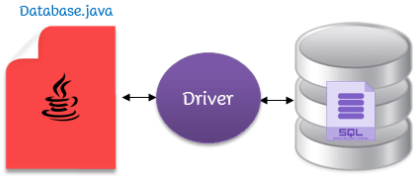
- เช่น **DELETE FROM** Customers **WHERE** CustomerName='Alfreds Futterkiste';

6. SQL Wildcard Characters

% : represents zero, one, or multiple characters (percent)

_ : represents a single character (underscore)

Basic Steps to Use a Database in Java



1. Loading Drivers
2. Establish a **connection**
3. Create and Execute **SQL** Statements
4. GET **ResultSet**
5. **Close** connections

ขั้นที่ 1 ทำการโหลด Driver

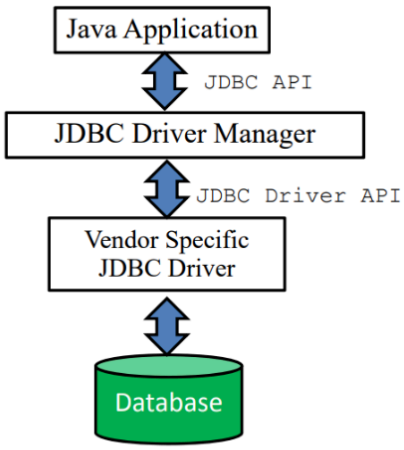
- JDBC Driver คือตัวเชื่อมต่อระหว่าง**โปรแกรมจาวา**กับ**ฐานข้อมูล** ดังนั้นจึงจำเป็นต้องโหลดมาให้เรียบร้อยก่อน ซึ่ง Driver มีหลายประเภท โดยที่ Library ของ JDBC Driver จะเป็น ไฟล์ .jar **(อย่าลืม add Java DB Library)** โดยการเรียก JDBC Driver ทำได้ดังนี้ Class.forName("ชื่อคลาสที่โหลด JDBC Driver");

ขั้นที่ 2 สร้างการเชื่อมต่อ

```
Connection cn=DriverManager.getConnection(url,username, psw);
```

ตัวอย่างการเขียน

```
public class ConnectionManager {
    public static Connection createConnection(String url, String username, String psw)
        //Optional for register driver name to DriverManager
        Class.forName("org.apache.derby.jdbc.EmbeddedDriver");

        //บอกที่อยู่ของ Database ตัวนั้นเพื่อ Connect
        Connection cn=DriverManager.getConnection(url,username, psw);
        return cn;
    }

    //Close Connection
    public static void closeConnection(Connection cn) throws SQLException{
        cn.close();
    }
}
```



--> ทำตัวควบคุมการเชื่อมต่อกับ Database ของเรา

```
public class PhoneController {
    private Connection con;

    public PhoneController(String user, String psw) throws ClassNotFoundException, SQLException{
        String url="jdbc:derby://localhost:1527/PhoneBook";
        con = ConnectionManager.createConnection(url, user, psw);
        System.out.println("Connection created Succesfully");
    }

    public void CloseConnection() throws SQLException{
        ConnectionManager.closeConnection(con);
        System.out.println("Close Connection Succesfully");
    }
```

--> ใน Driver Class เรียกตัว Controller มาทำงาน

- สร้างการเชื่อมต่อดังนี้ StudentController stdCtrl = new StudentController("sit","sit");
- อย่าลืม Try-Catch ใน Driver จะต้องรับผิดชอบ

ขั้นที่ 3 สร้างและ execute คำสั่ง

3.1. สร้าง Statement

- จะทำหน้าที่ส่ง SQL Statement ไปประมวลผล เชื่อมต่อ โดยการเอา ตัวแปรที่เชื่อมต่อ มาเรียก createStatement();
- เขียนได้ดังนี้ Statement stmt = con.createStatement();

3.2. ประมวลผล(execute) SQL statement

- การประมวลผลทำโดยนำตัวแปร Statement มาประมวลผล
- เขียนได้ดังนี้ stmt. executeQuery(SQL statement)

Execution

1. execute

- สำหรับไม่รู้ว่าเป็นประเภทไหน ให้ test ด้วย execute
- return boolean (true-select , false-update)

2. excuteUpdate

- ประมวลผลคำสั่ง SQL ทั้งหมด (INSERT, DELETE, UPDATE) ยกเว้น select
- return int (คืนค่าของแถวที่มีการเปลี่ยนแปลงไป)

3. executeQuery

- ใช้กับคำสั่ง Select // return เป็น ResultSet (ข้อมูลกลับมาในรูปแบบตาราง)

**ResultSet จะได้ข้อมูลเป็นตาราง การจะนำข้อมูลออกมาจะใช้คำสั่ง while loop และ resultSet.next() เพื่อวน 1 ครั้งจะได้ 1 RECORD และให้ใช้คำสั่ง resultSet.getString("COLUMN") หรือ resultSet.getInt("COLUMN") เพื่อดึงข้อมูลออกมา

```
ArrayList<PhoneBook> arrBook = new ArrayList<PhoneBook>();
Statement stmt = con.createStatement();
String sql = "SELECT * FROM food WHERE nickname LIKE '"+nickName+"%'";
ResultSet rs = stmt.executeQuery(sql);
while(rs.next()){
    String phone = rs.getString("telephone");
    String nickname = rs.getString("nickname");
    int gen = rs.getInt("gen");
    PhoneBook pb = new PhoneBook(phone,nickname,gen);
    arrBook.add(pb);
}
```

Statement

1. createStatement

2. prepareStatement

- เปลี่ยนเฉพาะค่าพารามิเตอร์ (parameter) ที่ส่งไป ในกระบวนการทำงาน มันจะแปลงคำสั่งรอไว้ แล้วเราจะ Set ค่าพารามิเตอร์ให้ที่หลังแทนที่ตำแหน่งเครื่องหมาย " ?" ที่เราใส่ไว้ในคำสั่ง SQL

- เช่น

```
//ทิ้งเครื่องหมาย ? แทนค่าที่จะมีการเปลี่ยน
String sq1="insert into student(stdId,firstname,lastname) "
           + "values (?,?,?)";
//เตรียม statement
PreparedStatement pStmt = con.prepareStatement(sq1);
```

```
//แทนค่าลง ? โดยระบุ index
pStmt.setInt(1, id);
pStmt.setString(2, firstname);
pStmt.setString(3, lastname);
//ทำการ executeUpdate
insertedRec+=pStmt.executeUpdate();
```

REVIEW LOOP, Scanner

- ปกติเราจะใช้ Scanner ในการอ่านค่าจาก Keyboard แต่เราสามารถอ่านได้จาก String หรือ ข้อความอีกด้วย
- การอ่านจากไฟล์

  File file = new File(“file.txt”);

  Scanner input = new Scanner(file);

  เช่น

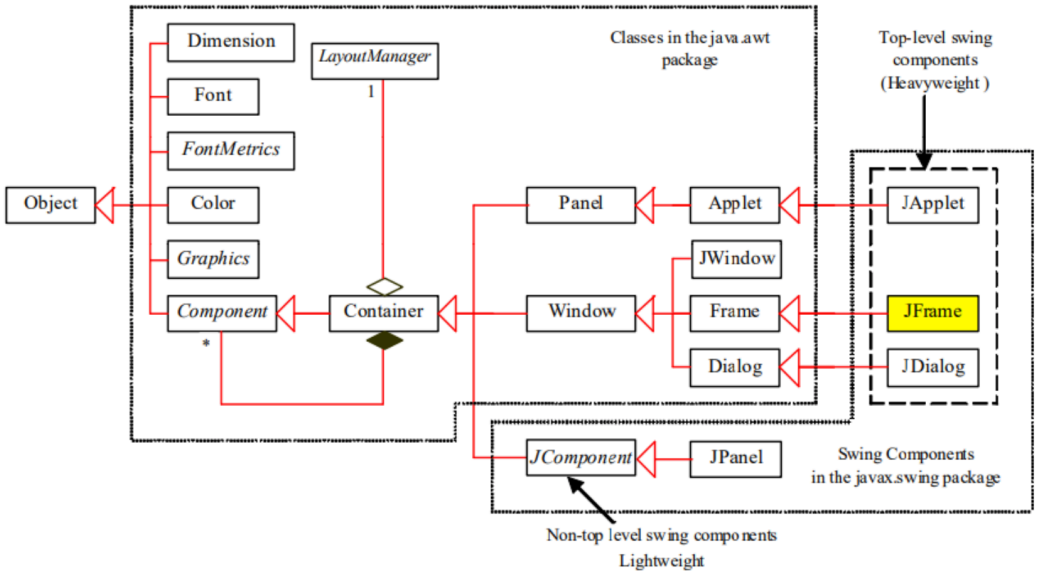
```
File myFile=new File(“StudentList.txt"); //ลักษณะไฟล์คือ 601 John Victor
Scanner sc = new Scanner(myFile);
while(sc.hasNextLine()){   //Loop บรรทัด   โดยผูกกับไฟล์
       String line=sc.nextLine();
       Scanner scStr=new Scanner(line);
       while(scStr.hasNext()){         //Loop คำ โดยผูกกับString
              Int id=scStr.nextInt();
              String firstname=scStr.next();
              String lastname=scStr.next();
              Student std=new Student(id,firstname,lastname); //1 Records
              System.out.println(std);
        }
 }
```

- การใช้ตัวคั่น (Delimiter)

```
String input = "1,2,red,blue";
Scanner s = new Scanner(input).useDelimiter(",");
System.out.println(s.next());
System.out.println(s.next());
System.out.println(s.next());
System.out.println(s.next());
s.close();
```

GUI (graphical user interface)

- เป็นการเขียนโปรแกรมJavaที่สร้างหน้าจอInterfaceที่เป็นแบบGraphic



- ประกอบด้วย 3 ส่วนหลัก ๆ คือ

1. Layout Managers

คือ ส่วนของการจัดวาง Layout ต่าง ๆ เช่น BorderLayout, FlowLayout, CardLayout, GridLayout บาง Layout จะจัดรูปแบบเหมือนกับ Table , Grid

2. Containers

คือพวก Component ที่สามารถนำมาสร้างเป็น Window Form หลักได้ เช่น Frame , Dialog และอื่น ๆ

3. Graphical Components

คือพวก Control ต่าง ๆ ที่เราจะนำมาใส่ใน Windows Form เช่น Label, TextBox , Button , Menu, Panel

MVC (Model View Controller)

1. Model → Entity Class
2. View → หน้าตาที่โชว์ เปลี่ยนจาก view เป็น GUI
3. Controller → หลังจาก Submit/Calculate จะทำงาน (ทำงานระหว่าง Model กับ View)

GUI COMPONENT

1. **Container**
   - *TOP Level :* JFrame (รันแอปยาว), JDialog(ติดต่อระยะสั้น)
   - *Non-Top Level : JPanel (ต้องวางบน Container อีกที)*
2. **Components**
   - JLabel, JTextField, Jbottom, JPanel, JTable

STEP TO CREATE GUI

1. สร้าง JFrame (กรอบแสดงผล)

```
JFrame f = new JFrame( "Frame Test");
//Default Size เป็น 0,0 / Visible เป็น false / ปุ่มปิด(DefaultCloseOperation)
//โดย FIX ลำดับดังนี้ size, Location, Visible !! ใส่หลัง add component

frameGrade.setSize(400, 400);
frameGrade.setLocationRelativeTo(null);   //เปิดมาหน้าจอโปรแกรมจะอยู่ตรงกลางของหน้าจอ
frameGrade.setVisible(true);
frameGrade.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

2. สร้าง JPanel (ส่วนจัดการแสดงผล)

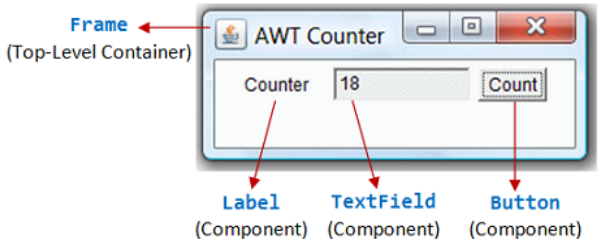
```
JPanel p = new JPanel();
//Default Layout Manager เป็น Flow Layout
```

- เพิ่มองค์ประกอบด้วยการ panel.add(component)

3. COMPONENTS

3.1. Button (ปุ่ม)

```
JButton buttonName = new JButton(text);
```



3.2. Label (แสดงผลข้อความ)

```
JLabel labelName = new JLabel();
```

3.3. TextField (รับข้อความจากผู้ใช้)

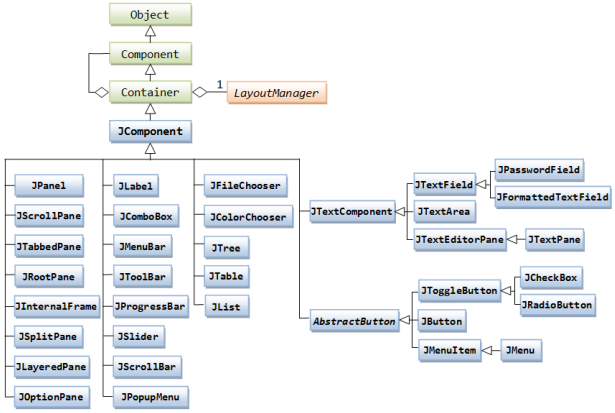
```
JTextField textName = new JTextField();
```

3.4. ScrollPane //มองเป็น Panel อยู่แล้ว

```
JScrollPane scroll = new JScrollPane();
```

3.4.1 Table

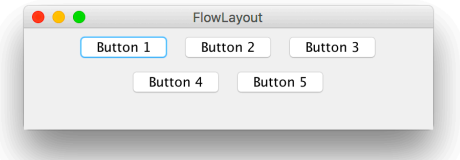
```
JTable table = new JTable();
JTable table = new JTable(data, columnNames);
```

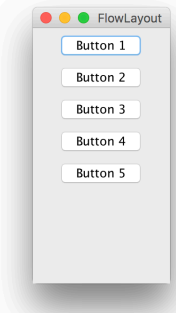


3. จัด Layout Manager

3.1. FlowLayout (Default)

- วาง Component จากบนลงล่าง ซ้ายไปขวา
- ถ้าไม่พอจะขึ้นบรรทัดใหม่ ตามขนาดความกว้าง

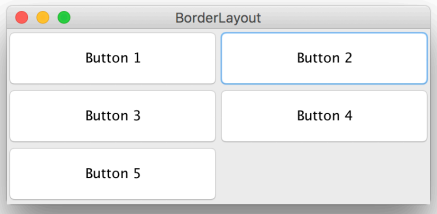





```
panel.add(component)
```

3.2. GridLayout

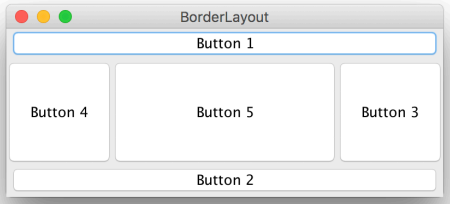
- แบ่งพื้นที่เป็น Grid ตามจำนวนแถวและคอลัมน์ที่กำหนด
- ใน 1 Grid วางได้ 1 component (จัดเรียงซ้ายไปขวา/บนลงล่าง)




```
panel.setLayout(new GridLayout(แถว,คอลัมน์))
```

3.3. BorderLayout

- แบ่งพื้นที่เป็น 5 ส่วนคือ north, south, center, east, west
- ใน 1 ส่วนวางได้ 1 component (จัดเรียงซ้ายไปขวา/บนลงล่าง)




```
panel.setLayout(new BorderLayout())
```

```
panel.add(component, BorderLayout.NORTH)
```

EVENT Handling

- การเขียนโปรแกรมแบบ GUI จะต้องจัดการในส่วนโต้ตอบระหว่างผู้ใช้กับ GUI โดยจะต้องตรวจจับเหตุการณ์ที่เกิดขึ้น เช่นกดปุ่ม (Button) หรือการกดปุ่ม Enter บนแป้นพิมพ์ที่ TextField โดยเป็นหน้าที่ของ Object ที่เป็น **EventListener**
- ทำได้โดยนำ ActionListener ไปผูกกับ component นั้นเพื่อตรวจจับเหตุการณ์

```
componentName.addActionListener(new ButtonListener())
```

ActionListener

- ต้องสร้าง class ที่ **implements ActionListener** โดยมี Method **actionPerformed(ActionEvent)**
- โดยสามารถมี parameter ที่เป็น JTextField ที่ต้องการรับค่าหรือเซ็ตค่า
- e.getActionCommand() จะบอกว่าปุ่มไหนถูกกด

```
public class ButtonListener implements ActionListener{
    private JTextField txtTest;

    public ButtonListener(JTextField txtTest) {
        this.txtTest = txtTest;
    }

    @Override
    public void actionPerformed(ActionEvent e) {

    }

}
```

JTextField Method

- getText() : เพื่ออ่านค่าจาก JTextField นั้น
- setText(string) : เพื่อกำหนดค่าจาก JTextField นั้น

ตัวอย่าง EventHandler

- ใน EventHandlerTest (Driver Class) จะทำการนำปุ่มไปผูกกับ ButtonListener Class และส่ง JTextField ไปให้

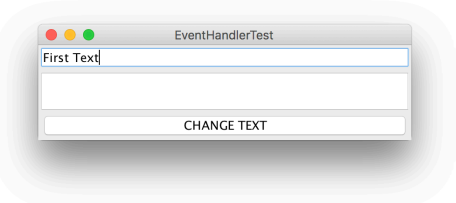
```
//--------Component---------
JTextField text1 = new JTextField("First Text",20);
JTextField text2 = new JTextField("",20);
JButton button = new JButton("CHANGE TEXT");

//------ActionListener-------
ButtonListener buttonEvent = new ButtonListener(text1,text2);
button.addActionListener(buttonEvent);
```

- ใน ButtonListener Class จะ implements ActionListener

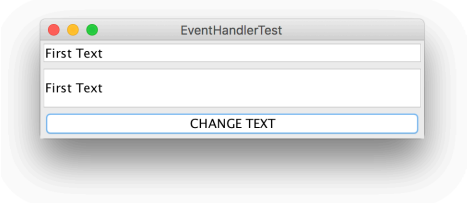
  โดยมี method actionPerformed ทำงาน รับค่าจาก text1 และนำไป set ที่ text2

```
public class ButtonListener implements ActionListener{
    private JTextField text1;
    private JTextField text2;

    public ButtonListener(JTextField text1, JTextField text2) {
        this.text1 = text1;
        this.text2 = text2;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        String message = text1.getText();
        text2.setText(message);
    }

}
```

- จะได้ผลลัพธ์ดังนี้



เมื่อกด Button “CHANGE TEXT” จะได้

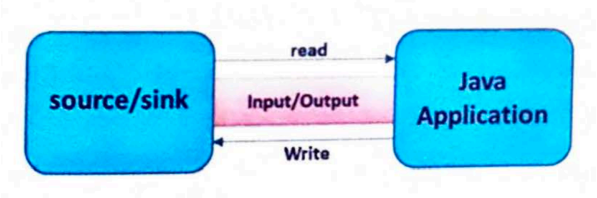


File I/O

- ในเนื้อหาเรียนที่ผ่านมาเป็นการเขียนโปรแกรมทำงานกับข้อมูลที่เก็บค่าไว้ในหน่วยความจำชั่วคราวเท่านั้น ในหัวข้อ File I/O จะเรียนรู้วิธีการเขียนโปรแกรมเพื่ออ่านและเขียนลงไฟล์

File Stream

- โปรแกรมจะอ่านหรือเขียนไฟล์ได้จะต้องมีสะพานส่งผ่านข้อมูลระหว่างกลางเรียกว่า STREAM โดยสะพานนี้จะใช้ส่งข้อมูลจากต้นทาง (source) ไปยังปลายทาง (sink)



- โดย source (ไฟล์) ——> sink (โปรแกรม) เรียกว่า Input Stream
  sink (โปรแกรม) ——> source (ไฟล์) เรียกว่า Output Stream
- File Stream เป็นท่อส่งข้อมูลในรูปแบบของไฟล์จากต้นทางไปยังปลายทาง ประกอบด้วยคลาส FileInputStream/ FileOutputStream สำหรับการอ่านและเขียนข้อมูล โดยรับส่งชนิดข้อมูลแบบ Byte หรือตัวเลข (เขียน/อ่านทีละ 1 byte) โดยข้อมูลในไฟล์จะถูกจัดเก็บเป็น Binary File

```
FileInputStream fileIn = new FileInputStream("fileName.dat");
FileOutputStream fileOut = new FileOutputStream("fileName.dat");
```

- FileOutputStream : เขียนข้อมูล byte ลงไฟล์ (แปลง byte stream ไปเป็น file)
- FileInputStream : การอ่านข้อมูล byte จากไฟล์ (แปลง file ไปเป็น byte stream)

DATA Stream

- สามารถใช้ Data Stream ทำการจัดเก็บข้อมูลที่ประกอบด้วยชนิดข้อมูลที่แตกต่างกันได้ โดยเป็นคลาสที่รองรับการเก็บข้อมูลชนิดพื้นฐานและเป็นคลาสที่รองรับการแปลงชนิดข้อมูลแบบ Byte เป็นชนิดอื่นๆได้

```
DataInputStream dataIn = new DataInputStream(fileIn);
DataOutputStream dataOut = new DataOutputStream(fileOut);
```

OBJECT Stream

- เป็นตัวแปลง byte stream ไปเป็น object และแปลง object กลับมาเป็น byte stream
- ObjectInputStream (ไว้แปลง object —-> byte stream)
- ObjectOutputStream (ไว้แปลง byte stream —-> object)

```
ObjectInputStream objectIn = new ObjectInputStream(fileIn);
ObjectOutputStream objectOut = new ObjectOutputStream(fileOut);
```

Method Input/Output Stream

- readInt() : เพื่ออ่านค่าข้อมูลชนิด int จากไฟล์ **read ตาม Datatypes
- writeInt(int) : เขียนข้อมูลชนิด int ลงไฟล์ **write ตาม Datatypes
- readUTF() : เพื่ออ่านค่าข้อมูลชนิด String จากไฟล์
- writeUTF(String) : เพื่อเขียนข้อมูลชนิด String ลงไฟล์
- readObject() : เพื่ออ่านค่าข้อมูลชนิด Object จากไฟล์
- writeObject(object) : เพื่อเขียนข้อมูลชนิด Object ลงไฟล์
- close() : ปิด stream

  **ทุกครั้งที่ต่อ Stream ไปยัง Resource ที่กำหนด Resource นั้นจะถูก lock ไว้ไม่ให้ส่วนอื่นๆ เข้าถึง

รูปแบบการเขียน

```
FileInputStream fileIn = new FileInputStream("fileName.dat");
DataInputStream dataIn = new DataInputStream(fileIn);
DataType varName = dataIn.methodName();
```

Class DataInputStream

java.lang.Object
  java.io.InputStream
    java.io.FilterInputStream
      java.io.DataInputStream

Class ObjectInputStream

java.lang.Object
  java.io.InputStream
    java.io.ObjectInputStream

Exception for Input Stream

- Signals that an end of file or end of stream has been reached unexpectedly during input.

  This exception is mainly used by data input streams to signal end of stream.

```
try{

}
catch(EOFException eof){
   try{
      ois.close();
   }catch(IOException ex){
      System.out.println(ex);
   }
}
```

Class EOFException

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.io.IOException
        java.io.EOFException

REF

หนังสือคู่มือเรียนเขียนโปรแกรมภาษา Java ฉบับสมบูรณ์ 2nd Edition

https://docs.oracle.com/javase/7/docs

https://expert-programming-tutor.com/tutorial/java/12_reading_file.php

http://na5cent.blogspot.com/2015/01/java-stream.html