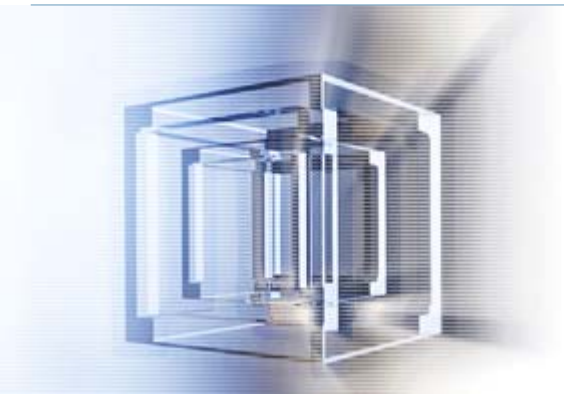
NETBEANS TUTORIALS



BY

P U P A
DEPARTMENT OF COMPUTER ENGINEERING
PRINCE OF SONGKLA UNIVERSITY

สารบัญ

1 Introduction to Netbeans 2

1.1 การติดตั้ง Netbeans 2

1.2 การติดตั้ง Plug-in 2

1.3 การสร้างโปรแกรมแรกด้วย NetBeans 2

1.3.1 โปรแกรม Hello World 2

1.3.2 โปรแกรม Counter 2

1.4 Code Completion 3

1.4.1 โปรแกรม Salary Calculator 3

1.5 Code Refactoring 5

1.6 Code Templates 5

1.6.1 การเพิ่ม Code Template 5

1.6.2 การใช้งาน Code Template 5

2 Third Party Libraries 6

2.1 JFreeChart 6

2.2 การสร้างไลบรารีด้วยตนเอง 9

2.2.1 สร้างไลบรารี 9

2.2.2 นำไลบรารีมาใช้งานกรณีไม่มีซอร์สโค้ด 9

2.2.3 นำไลบรารีมาใช้งานกรณีมีซอร์สโค้ด 9

3 UML 10

3.1 การออกแบบโปรแกรมด้วย UML 10

3.1.1 Entities 10

3.1.2 Relationships 11

3.2 Forward Engineering 11

3.3 Reverse Engineering 11

4 Subversion 12

4.1 การติดตั้ง Subversion 12

4.1.1 ติดตั้งโปรแกรม 12

4.1.2 สร้างเซอร์วิส Subversion บน Windows 12

4.2 การใช้งาน Subversion เบื้องต้น 13

4.3 การใช้งาน Subversion สำหรับการพัฒนาเป็นทีม 14

5 Ant script 15

5.1 การสร้างล็อกไฟล์ด้วย log4j 15

บทนำ

เอกสารฉบับนี้ จัดทำเพื่อเป็นเครื่องมือช่วยในการทำความเข้าใจ ความสามารถของ Netbeans ในการสร้างโปรแกรมประยุกต์ต่างๆ ด้วยภาษาจาวา โดยเนื้อหาเน้นไปที่การใช้เครื่องมือ เพื่อการพัฒนาตามหลักการที่ดีของวิศวกรรมซอฟต์แวร์ ตั้งแต่กระบวนการออกแบบ (Design) ไปจนถึงกระบวนการนำไปใช้งาน (Deployment)

ในกระบวนการการออกแบบ Netbeans สนับสนุน Forward Engineering โดยมีเครื่องมือในการวาด UML แล้วทำการสร้างโค้ดอัตโนมัติ และยังสนับสนุน Reverse Engineering ซึ่งสามารถสร้าง UML จากโค้ดที่มีอยู่แล้วได้

ในระหว่างการพัฒนา Netbeans มีเครื่องมือต่างๆ (1) ในการสร้างโค้ด ไม่ว่าจะเป็นการสร้างโค้ดอัตโนมัติ (Code Insertion) การจัดรูปแบบโค้ด (Code Refactoring) ต้นแบบของโค้ด (Code Template) และการเติมโค้ดอัตโนมัติ (Code Completion) เพื่อทำให้การเขียนโค้ดเป็นไปอย่างสะดวกและรวดเร็ว (2) การคอมไพล์ และการรันโปรแกรมที่พัฒนาขึ้นเป็นไปได้อย่างสะดวก โดยมี Ant เป็นเครื่องจักรสำคัญ เอกสารฉบับนี้จะได้แนะนำการเพิ่มเติมเกี่ยวกับ Ant Script เพื่อทำให้การพัฒนาเป็นไปอย่างสะดวกยิ่งขึ้น

ในการพัฒนาโปรแกรมในลักษณะ Desktop Application นั้น ทำได้สะดวกเนื่องจาก เฟรมเวิร์กที่มีความสามารถสูงหลายตัวได้ถูกรวบรวมเอาไว้กับ Netbeans อย่างสมบูรณ์ ไม่ว่าจะเป็น Beans Binding (JSR 295) และ Swing Application Framework (JSR 296) รวมไปถึง Free Design Layout (โค้ดเนมเดิมชื่อ Matisse) ที่ทำให้การจัดคอมโพเนนต์ต่างๆ เป็นไปได้โดยง่าย

ในการพัฒนาโปรแกรมที่มีการใช้งานฐานข้อมูล Netbeans ได้นำเสนอ Persistent API ซึ่งแนวทางที่มีประสิทธิผลสูง ไม่ว่าในแง่ของการพัฒนาโค้ด และประสิทธิภาพเมื่อโปรแกรทำงาน โดยเครื่องจักรเบื้องหลังคือ Oracle Toplink ทั้งนี้ Desktop Application ยังคงทำงานเชื่อมต่อกันอย่างสมบูรณ์ เมื่อทำงานร่วมกับ Persistent API

เอกสารฉบับนี้ ไม่ได้จัดทำขึ้นเพื่อเป็นเอกสารอ้างอิง แต่เป็นเพียงเครื่องมือนำทางในการเรียนรู้ Netbeans เท่านั้น ผู้สนใจศึกษาเรียนรู้ ควรใช้หนังสืออ้างอิงประกอบเพื่อความเข้าใจที่ถ่องแท้

Ch1

1 INTRODUCTION TO NETBEANS

ในบทนี้จะแนะนำเกี่ยวกับพื้นฐานที่สำคัญในการใช้งาน Netbeans เพื่อพัฒนาโปรแกรมประยุกต์ โดยจะเริ่มต้นจากการติดตั้ง Netbeans และ Plug-in พร้อมทั้งการสร้างโค้ดด้วยฟังก์ชันพิเศษต่างๆ

1.1 การติดตั้ง Netbeans

1. เลือกแพ็คเกจที่ต้องการติดตั้ง
2. ทำการติดตั้งตามคำแนะนำของตัวติดตั้ง

1.2 การติดตั้ง Plug-in

1. ติดตั้ง Plug-in ที่ชื่อว่า Eclipse Project Importer
2. ใช้ Plug-in เพื่ออิมพอร์ตโปรเจ็คที่สร้างขึ้นด้วยโปรแกรม Eclipse
3. ทำการรันโปรเจ็คที่อิมพอร์ตมา

1.3 การสร้างโปรแกรมแรกด้วย NetBeans

1.3.1 โปรแกรม Hello World

สร้างโปรเจ็คโดยให้ชื่อว่า *hello_world* และเขียนโปรแกรมโดยใช้ Code Template เพื่อแสดงผลลัพธ์

```
“Hello World !”
```

1.3.2 โปรแกรม Counter

1. สร้างโปรเจ็คโดยให้ชื่อว่า *counter*
2. สร้างคลาส *Counter*
3. เพิ่มฟิลด์ต่อไปนี้

```
private short c = 0;
```

4. เพิ่มเมธอดดังนี้

```
void increase()        //เพื่อเพิ่มค่าของ c อีก 1
void decrease()        //เพื่อลดค่าของ c อีก 1
int getValue()         //เพื่ออ่านค่าของ c
String toString()      //คืนค่าข้อความของ c
```

5. สร้าง JUnit Test ของคลาส *Counter*
6. ในโค้ดที่สร้างขึ้น ให้ลบเมธอด *testToString* และ *testGetValue*
7. ในเมธอด *testIncrease* ให้เพิ่มค่าของ *instance* ไป 10000 ครั้ง และทำการตรวจสอบว่า ค่าของ *instance* เป็น 10000 หรือไม่
8. ใน เมธอด *testDecrease* ให้ลดค่าของ *instance* ไป 10 ครั้ง และทำการเช็คว่า ค่าของ *instance* เป็น -10 หรือไม่
9. ทำการรัน JUnit Test
10. เปลี่ยนชนิดตัวแปร *c* ใน คลาส *Counter* ให้เป็น *int* และรัน JUnit Test อีกครั้ง
11. สร้างคลาส *Main* และกำหนดให้เมธอด *main* สร้างออบเจ็คจากคลาส *Counter* พร้อมทั้งสั่งลดค่า เพิ่มค่า และแสดงค่าของ c
12. ทำการรันโปรแกรม *counter*

1.4 Code Completion

1.4.1 โปรแกรม Salary Calculator

1. สร้างคลาส Employee โดยมีฟิลด์ดังนี้

```
int id;
String name ;
double salary ;
String department ;
```

2. สร้างคอนสตรัคเตอร์ โดยรับพารามิเตอร์ตามข้างต้น โดยใช้ Code Insertion
3. เพิ่มเมธอด getter และ setter สำหรับพารามิเตอร์ข้างต้น โดยใช้ Code Insertion
4. สร้างคลาส *SalaryCalculator*
5. เขียนโค้ดดังต่อไปนี้ โดยใช้ Code Completion

```
public class SalaryCalculator {
       ArrayList<Employee> employees = new ArrayList<Employee>();
    public void addEmployee(Employee e){
        employees.add(e);
    }

    public void calculate(){
        Hashtable<String, ArrayList<Employee>> departments =
            new Hashtable<String,ArrayList<Employee>>();
        for (Employee e : employees) {
            ArrayList<Employee> dept_employees =
                departments.get(e.getDepartment());
            if (dept_employees == null) {
                dept_employees = new ArrayList<Employee>();
                departments.put(e.getDepartment(), dept_employees);
            }
            dept_employees.add(e);
        }
       for(String department : departments.keySet()){
           double total = 0.0;
           for(Employee e: departments.get(department)){
               total += e.getSalary();
           }
           System.out.println("Department " + department + " has to pay " + total);
       }
    }
    public static void main(String[] args) {
        SalaryCalculator sc = new SalaryCalculator() ;
        sc.addEmployee(new Employee(1, "John", 120.0, "Engineer"));
        sc.addEmployee(new Employee(2, "Jane", 150.0, "Engineer"));
        sc.addEmployee(new Employee(3, "Marry", 50.0, "Cleaning"));
        sc.calculate();
    }
}
```

1.5 Code Refactoring

1. ใช้โค้ดจากการทดลองก่อนหน้า
2. เปลี่ยนชื่อตัวแปร *sc* โดยใช้ Code Refactoring
3. ใช้ Code Refactoring ในแยกการทำงานของเมธอด *calculate* ในส่วนของการจัดกลุ่มพนักงานตามแผนก ออกเป็นเมธอด *group_by_department* ซึ่งมี Access Modifier เป็น *private*

1.6 Code Templates

1.6.1 การเพิ่ม Code Template

ใช้ Editor Options ของ Netbeans เพื่อกำหนด Code Template ดังแสดงข้างล่าง

```
Abbreviation: news
Expanded text: String ${newVarName default="name"} = new String("${cursor}");
```

1.6.2 การใช้งาน Code Template

1. ทำการสร้างโปรเจ็ค โดยให้ชื่อว่า *code_template*
2. เขียนโค้ดต่อไปนี้โดย ใช้ Code Template

```
    public static void main(String[] args) { //psvm
        boolean boo = false ; //bo, fa
        String s1 = new String("code template"); //news
        for (int i = 0; i < 20; i++) { //fori
            if (boo) { //ifelse
                System.out.println("true"+s1);
                boo = false ;
            } else {
                System.out.println("false"); //sout
                boo= true ;
            }
        }
    }
```

2 THIRD PARTY LIBRARIES

ในบทนี้ จะเป็นการอธิบายแนวทางการพัฒนาโปรแกรมประยุกต์บน Netbeans เมื่อมีการใช้งานไลบรารีเพิ่มเติม โดยจะแบ่งออกเป็น 2 ช่วงหลัก คือ ช่วงที่ 1 เป็นการใช้งานไลบรารีเพิ่มเติมที่สร้างขึ้นโดยผู้อื่น และในช่วงที่ 2 จะเป็นการสร้างไลบรารีขึ้นมาเอง และนำมาใช้งาน

2.1 JFreeChart

1. ทำการสร้างโปรเจ็ค ให้ชื่อว่า *jfree*
2. ใช้ Project Properties เพื่อเพิ่มไฟล์จาร์ 2 ไฟล์ที่จำเป็นต้องใช้ในโปรแกรมที่พัฒนา คือ *jcommon-1.0.10.jar* และ *jfreechart-1.0.6.jar*
3. สร้างคลาส *BarChartFrame* โดยสืบทอดจาก *ApplicationFrame* ซึ่งเป็นคลาสของไลบรารี JFreeChart
4. เพิ่มคอนสตรัคเตอร์โดยใช้ Code Insertion

```
public BarChartFrame(String title) {
        super(title);
}
```

5. เขียนโค้ด เพื่อให้ได้ผลลัพธ์สุดท้ายดังแสดง

```
public class BarChartFrame extends ApplicationFrame {

    public BarChartFrame(String title) {
        super(title);
        JFreeChart chart = createChart(createDataset());
        ChartPanel chartPanel = new ChartPanel(chart, false);
        chartPanel.setPreferredSize(new Dimension(500, 270));
        setContentPane(chartPanel);
    }

    private static CategoryDataset createDataset() {

        // row keys...
        String series1 = "First";
        String series2 = "Second";
```

```
        // column keys...
        String category1 = "Category 1";
        String category2 = "Category 2";
        String category3 = "Category 3";

        // create the dataset...
        DefaultCategoryDataset dataset = new DefaultCategoryDataset();

        dataset.addValue(1.0, series1, category1);
        dataset.addValue(4.0, series1, category2);
        dataset.addValue(3.0, series1, category3);

        dataset.addValue(5.0, series2, category1);
        dataset.addValue(7.0, series2, category2);
        dataset.addValue(6.0, series2, category3);

        return dataset;
    }

    private static JFreeChart createChart(CategoryDataset dataset) {
        // create the chart...
        JFreeChart chart = ChartFactory.createBarChart(
                "Bar Chart Demo 1", // chart title
                "Category", // domain axis label
                " Value", // range axis label
                dataset, // data
                PlotOrientation.VERTICAL, // orientation
                true, // include legend
                true, // tooltips?
                false // URLs?
                );

        // NOW DO SOME OPTIONAL CUSTOMISATION OF THE CHART...

        // set the background color for the chart...
        chart.setBackgroundPaint(Color.white);
```

```
        // get a reference to the plot for further customisation...
        CategoryPlot plot = (CategoryPlot) chart.getPlot();
        plot.setBackgroundPaint(Color.lightGray);
        plot.setDomainGridlinePaint(Color.white);
        plot.setDomainGridlinesVisible(true);
        plot.setRangeGridlinePaint(Color.white);

        // set the range axis to display integers only...
        final NumberAxis rangeAxis = (NumberAxis) plot.getRangeAxis();
        rangeAxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits());

        // disable bar outlines...
        BarRenderer renderer = (BarRenderer) plot.getRenderer();
        renderer.setDrawBarOutline(false);

        // set up gradient paints for series...
        GradientPaint gp0 = new GradientPaint(0.0f, 0.0f, Color.blue,
                0.0f, 0.0f, new Color(0, 0, 64));
        GradientPaint gp1 = new GradientPaint(0.0f, 0.0f, Color.green,
                0.0f, 0.0f, new Color(0, 64, 0));

        renderer.setSeriesPaint(0, gp0);
        renderer.setSeriesPaint(1, gp1);

        CategoryAxis domainAxis = plot.getDomainAxis();
        domainAxis.setCategoryLabelPositions(
                CategoryLabelPositions.createUpRotationLabelPositions(
                Math.PI / 6.0));
        // OPTIONAL CUSTOMISATION COMPLETED.

        return chart;
    }
}
```

6. เขียนโค้ดต่อไปนี้ในเมธอด *main* ในคลาส *Main*

```
BarChartFrame frame = new BarChartFrame("my chart");
frame.pack();
frame.setVisible(true);
```

7. ทำการ Build โปรเจ็ค แล้วนำโฟลเดอร์ *dist* ของโปรเจ็ค *jfree* มาวางไว้บน Desktop
8. รัน ไฟล์ *jfree.jar*

2.2 การสร้างไลบรารีด้วยตนเอง

2.2.1 สร้างไลบรารี

1. ทำการสร้างโปรเจ็ค โดยกำหนดให้เป็น Java Class Library และให้ชื่อว่า *counter*
2. สร้างคลาส โดยให้ชื่อว่า *Counter* โดยให้อยู่ในแพ็กเกจ *name.test*
3. ทำการ Build และนำไฟล์ *counter.jar* ในโฟลเดอร์ *dist* ของโปรเจ็ค *counter* มาวางไว้บน Desktop

2.2.2 นำไลบรารีมาใช้งานกรณีไม่มีซอร์สโค้ด

1. สร้าง โปรเจ็ค โดยให้ชื่อว่า *test_counter*
2. ใช้ Project Properties เพื่อเพิ่มไฟล์จาร์ *counter.jar* จาก Desktop
3. ทดลองเขียนโค้ดสร้างออบเจ็คของ คลาส *Counter* ทำการเรียกเมธอด *count* และแสดงผลลัพธ์จากเมธอด *getCount*

2.2.3 นำไลบรารีมาใช้งานกรณีมีซอร์สโค้ด

1. ทำการลบไฟล์จาร์ *counter.jar* ออกจาก Project Properties
2. เพิ่มโปรเจ็ค *counter* ให้กับรายการไลบรารีโดยตรง
3. ทดลองรันโปรเจ็คอีกครั้ง

3 UML

Ch3

ในการทำงานกับภาษาจาวานั้นเราสามารถสร้าง UML มาช่วยในการออกแบบได้โดย Netbeans จะมีความสามารถในการเปลี่ยนแปลง UML เป็นโค้ดในภาษาจาวาได้ รวมถึงความสามารถในการทำ Reverse Engineering ด้วย

3.1 การออกแบบโปรแกรมด้วย UML

3.1.1 Entities

1. สร้างโปรเจ็คชื่อ my_uml โดยกำหนดชนิดของโปรเจ็คเป็น Java Platform Model
2. สร้างแพ็กเกจ *bankpack*
3. สร้างคลาส *BankAccount* โดยมีแอตริบิวต์ และโอเปอร์เรชันสอดคล้องกับโค้ดที่แสดงข้างล่าง

```
***Attribute
private int balance

***Operations
public BankAccount()
public int getBalance()
public void setBalance(int val)
public void withdraw(int amount)
```

4. สร้างคลาส *Checking* โดยมี โอเปอร์เรชัน ดังนี้

```
public Checking()
```

5. สร้างคลาส *AccountTest* โดยมี โอเปอร์เรชัน ดังนี้

```
public AccountTest()
```

6. สร้างอินเตอร์เฟส *Bank* โดยให้มี โอเปอร์เรชัน ดังนี้

```
public void deposit()
```

3.1.2 Relationships

สร้างความสัมพันธ์ในแต่ละคลาสได้จาก Component Palette ของ Netbeans โดยให้สร้างความสัมพันธ์ดังนี้

- ให้คลาส และอินเตอร์เฟสทั้งหมดอยู่ภายในแพ็คเก็จ *bankpack*
- ให้คลาส *BankAccount* อิมพลีเมนต์ อินเตอร์เฟส *Bank*
- ให้คลาส *Checking* สืบทอดมาจากคลาส *BankAccount*

3.2 Forward Engineering

1. สร้างโปรเจ็ค *my_java*
2. ทำการสร้างโค้ดของ UML ไปยังโปรเจ็ค *my_java* ที่สร้างขึ้น
3. แก้ไฟล์ที่ได้ทำการสร้างขึ้น โดยที่คลาส *BankAccount* ทำการเพิ่มโค้ดที่เมธอด *deposit* ดังนี้

```
setBalance(getBalance() + amount) ;
```

4. ทำการเพิ่มเมธอด *main* ในคลาส *Main* ดังนี้

```
public static void main (String args[]) {
    Ckecking myChecking = new Checking() ;
    mychecking.deposit(1000) ;
    System.out.prinntln("Checking Balance is : " + myChecking.getBalance()) ;
}
```

3.3 Reverse Engineering

ใช้ Reverse Engineer ในการสร้าง UML จากโปรเจ็ค *my_java*

4 SUBVERSION

Ch4

ในบทนี้จะกล่างถึงการใช้ ใน Netbeans ร่วมกับ Subversion เพื่อการจัดการด้านการจับเก็บโค้ดเวอร์ชันล่าสุดและก่อนหน้า รวมทั้งเครื่องมือที่เกี่ยวข้องอื่นๆ ที่จะทำให้การพัฒนาโปรแกรมที่มักจะถูกพัฒนาโดยทีมงาน เกิดความสะดวก และมีประสิทธิภาพสูงสุด

4.1 การติดตั้ง Subversion

4.1.1 ติดตั้งโปรแกรม

1. ทำการติดตั้งโปรแกรมชื่อ *svn-1.4.4setup.exe*
2. ทำการติดตั้งตามคำแนะนำของตัวติดตั้ง

4.1.2 สร้างเซอร์วิส Subversion บน Windows

1. สร้างโฟลเดอร์ *C:\svn*
2. ทำการรัน Command Prompt ขึ้นมาจากนั้นพิมพ์คำสั่งดังนี้ที่ *C:\svn*

```
svnadmin create test_svn
```

3. จากนั้นให้ทำการพิมพ์คำสั่งดังนี้

```
sc create svn binpath= "\"C:\program files\Subversion\bin\svnserve.exe\" --service
-r C:\svn" displayname= "Subversion" depend= Tcpip start= auto
```

**หลังเครื่องหมาย = มีเว้นวรรคหนึ่งครั้ง

4. ทำการเริ่มต้นเซอร์วิส *Subversion* ขึ้นมาโดยไปที่ Control Panel -> Administrative Tools -> Services
5. แก้ไขไฟล์กำหนดค่าการทำงานต่างๆ ในโฟลเดอร์ *test_svn* ที่สร้างขึ้นโดยคำสั่ง *svnadmin*
   - ไฟล์ *svnserve.conf* ทำการตั้งค่าเริ่มต้นให้กับการควมคุม Repository ดังนี้
     - กำหนดให้ผู้ที่เข้าใช้ต้องล็อกอินเข้ามาทุกครั้งที่เข้าใช้งาน
     - กำหนดไฟล์ซึ่งเป็นฐานข้อมูลรหัสผ่าน
     - กำหนดไฟล์ซึ่งกำหนดสิทธิการเข้าใช้งาน Repository
   - ไฟล์ *passwd* ซึ่งกำหนดชื่อผู้ใช้ และรหัสผ่าน

- ไฟล์ *authz* ซึ่งกำหนดสิทธิการเข้าใช้งาน Repository

4.2 การใช้งาน Subversion เบื้องต้น

1. สร้างโปรเจ็คขึ้นมาโดยชื่อว่า *TestSvn* และสร้างคลาสชื่อ *TestSvn* โดยมีโค้ดดังต่อไปนี้

```
public class TestSvn {
    public static void main(String[] args) {
        System.out.println("Test Subversion");
    }
    public void foo () {
        System.out.println("Foo");
    }
}
```

2. ใช้ฟังก์ชัน Versioning เพื่อ Import โปรเจ็คนี้เข้าสู่ Repository โดยกำหนด Repository URL เป็น

```
svn://localhost/test_svn
```

3. ทำการ Checkout โปรเจ็คออกมาโดยกำหนดโฟล์เดอร์ที่ใช้เก็บโปรเจ็คเป็นที่ใหม่
4. แก้ไขคลาส *TetstSvn* ที่ Checkout ออกมา ดังนี้

```
public class TestSvn {
    public static void main(String[] args) {
        System.out.println("Test Subversion") ;
    }
    public void testCommit () {
    System.out.println("testCommit") ;
    }
}
```

5. ทำการ Diff เพื่อดูความเปลี่ยนแปลงของโค้ด

6. เพิ่มคลาส *Test* ลงไปในโปรเจ็ค โดยกำหนดให้มีเมธอด *main* ดังนี้

```
public static void main (String args[]) {
  System.out.println("Test Diff") ;
}
```

7. สร้างไฟล์ test.log แล้วใช้ฟังก์ชัน Ignore
8. ทำการ Commit ซอร์สโค้ดทั้งหมด
9. ทำการเปลี่ยนแปลงคลาส *Test* เล็กน้อยแล้วทำการบันทึก
10. ใช้ฟังก์ชัน Revert Modification

4.3 การใช้งาน Subversion สำหรับการพัฒนาเป็นทีม

1. ทำการเปลี่ยนแปลงโค้ดโดยผู้ใช้หลายคน
2. ทดสอบฟังก์ชัน Update เพื่ออ่านไฟล์โค้ดล่าสุด
3. ทดสอบฟังก์ชัน Resolve Conflicts เมื่อมีผู้ใช้หลายคนแก้ไขไฟล์โค้ดเดียวกัน

Ch5

5 ANT SCRIPT

Ant เป็นเครื่องจักรสำคัญของ Netbeans ในการคอมไพล์ และรันโปรแกรม แม้ว่าในหลายครั้งที่ผู้ใช้ Netbeans ไม่มีความจำเป็นต้องรับรู้การทำงานของ Ant อย่างไรก็ตาม การรับรู้ และเข้าใจ Ant จะช่วยทำให้การพัฒนาเป็นไปอย่างราบรื่น และสง่างาม

5.1 การสร้างล็อกไฟล์ด้วย log4j

1. สร้างโปรเจ็คขึ้นมาใหม่โดยตั้งชื่อว่า *Log4j*
2. ทำการเพิ่มจาร์ไฟล์ชื่อ *Log4j-1.2.13jar* ในโปรเจ็ค *Log4j*
3. ในโปรเจ็ค *Log4j* ทำการพิมพ์โค้ดดังนี้

```
package log4j;
import org.apache.log4j.BasicConfigurator;
import org.apache.log4j.Logger;

public class Main {
    public static final Logger logger  = Logger.getLogger(Main.class);
    public static void main(String[] args) {
        BasicConfigurator.configure();
        logger.info("hello world");
        logger.error("Hellow another world");
    }
}
```

4. จากนั้นทำการรันโปรเจ็ค *Log4j*
5. ทำการสร้างไฟล์ใหม่เป็นแบบ Properties File โดยตั้งชื่อว่า *Log4j* โดยทำการพิมพ์โค้ดดังนี้

```
log4j.rootLogger=DEBUG, stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%p - %m%n
```

6. จากนั้นทำการ Build โปรเจ็ค *Log4j*
7. ทำการรัน Command Prompt ขึ้นมาจากนั้นไปที่ โปรเจ็ค *Log4j* และไปที่โฟล์เดอร์ *dist* จากนั้นพิมพ์คำสั่ง

```
java –jar log4j.jar
```

5.2 Ant Script

1. เลือก *build*.xml ในโปรเจ็คก่อนหน้า เพิ่ม Ant Script ดังนี้

```
<target name="-post-compile">
        <copy todir="${dist.dir}" file="log4j.properties"/>
    </target>
</project>
```

2. ทำการ Clean and Build โปรเจ็ค *Log4j*
3. ทำการรัน Command Prompt ขึ้นมาจากนั้นไปที่ โปรเจ็ค *Log4j* และไปที่โฟล์เดอร์ *dist* จากนั้นพิมพ์คำสั่ง

```
java –jar log4j.jar
```

Ch6

6 PERSISTENT API

การทดลองในบทนี้เป็นการทดลองใช้งาน Persistent API ซึ่งเป็นไลบรารีที่มีไว้สำหรับเก็บข้อมูลลงในฐานข้อมูล โดยข้อมูลดังกล่าวจะไม่ถูกลบทิ้งไปแม้ว่าโปรแกรมจะถูกปิดตัวไปแล้วก็ตาม สำหรับการทดลองนี้จะแบ่งออกเป็นสองส่วน คือส่วนที่ใช้เก็บค่าและแสดงผลข้อมูลจากฐานข้อมูล และส่วนที่ใช้ความสามารถของ Persistent API ช่วยในการเก็บและแสดงผลข้อมูลจากฐานข้อมูล

6.1 การอ่านและเก็บข้อมูลโดยใช้ Persistent API

1. สร้างโปรเจ็ค โดยตั้งชื่อว่า *test_persistent*
2. สร้างคลาสชนิด Persistence Unit โดยตั้งชื่อว่า *TestPersistPU* และกำหนดฐานข้อมูลคอนเนคชั่น
3. สร้าง Entity คลาสชื่อ *Employee* โดยมีฟิลด์ดังนี้

```
String name;
Float salary;
```

6. สร้างคอนสตรัคเตอร์ โดยรับพารามิเตอร์ตามข้างต้น โดยใช้ Code Insertion
7. เพิ่มเมธอด getter และ setter สำหรับพารามิเตอร์ข้างต้น โดยใช้ Code Insertion
8. เพิ่มเมธอด *persist* (Persistence) ในคลาส *Employee*
9. แก้เมธอด *toString* ให้แสดงข้อมูล *name* และ *salary* ของ *Employee*
10. ทดสอบการเก็บและแสดงผลข้อมูลจากฐานข้อมูลโดยการเพิ่มโค้ดดังต่อไปนี้ลงในคลาส *Main*

```
static EntityManagerFactory emf =
javax.persistence.Persistence.createEntityManagerFactory("TestPersistPU");

 public static void main(String[] args) {
    EntityManager em = emf.createEntityManager();
    Employee e =em.find(Employee.class, 2L);
    Employee e1 = new Employee("aErOn",30.0f);
    em.getTransaction().begin();
    em.persist(e1);
    em.getTransaction().commit();
    List<Employee> list = em.createQuery("select e from Employee").getResultList();
    for(Employee employee: list){
         System.out.println(employee);
```

```
    }
        em.close();
}
```

11. สั่งคอมไพล์และรัน จากนั้นสังเกตุการแสดงผลของโปรแกรมใน Netbeans เปรียบเทียบกับข้อมูลที่อยู่ในฐานข้อมูล

6.2 การใช้ความสามารถเพิ่มเติมของ Persistent API

1. สร้าง Entity คลาสชื่อว่า *Department* โดยมีฟิลด์ดังนี้

```
String name;
List<Employee> employees;
```

2. สร้างคอนสตรัคเตอร์ โดยรับพารามิเตอร์ตามข้างต้น โดยใช้ Code Insertion
3. เพิ่มเมธอด getter และ setter สำหรับพารามิเตอร์ข้างต้น โดยใช้ Code Insertion
4. เพิ่มแอนโนเทชั่นก่อนการประกาศคลาส ดังนี้

```
@NamedQueries(
    @NamedQuery(name="all_departments", query="select d from Department d")
)
```

5. เพิ่มโค้ดต่อไปนี้บนเมธอด *getEmployees*

```
@OneToMany(cascade=CascadeType.ALL, mappedBy="department")
@OrderBy(value="salary desc")
```

6. เปลี่ยนแปลงโค้ดในคลาส *Main* ให้สามารถเก็บตัวแปรชนิดลิสต์ลงในฐานข้อมูลโดยผ่านทางเมธอด *Persist* ได้ ดังต่อไปนี้

```
public static void main(String[] args) {
        EntityManager em = emf.createEntityManager();
        Department d1 = new Department();
        d1.setName("Engineer");
        List<Employee> list1 = new ArrayList<Employee>();
        list1.add(new Employee("A", 1000.0f, d1));
        list1.add(new Employee("B", 1200.0f, d1));
```

```
        d1.setEmployees(list1);
        Department d2 = new Department();
        d2.setName("Finance");
        List<Employee> list2 = new ArrayList<Employee>();
        list2.add(new Employee("C", 1000.0f, d2));
        d2.setEmployees(list2);
        em.getTransaction().begin();
        em.persist(d1);
        em.persist(d2);
        em.getTransaction().commit();
        em.close();*/
}
```

7. สั่งคอมไพล์และรัน
8. เปลี่ยนแปลงคลาส main ดังต่อไปนี้ เพื่อนำข้อมูลมาแสดงผล โดยสังเกตุว่าวิธีการเรียกข้อมูลจะกระทำผ่านทางเมธอด *createNameQuery* ซึ่งต่างจากการทดลองก่อนหน้าที่ใช้เมธอด *createQuery*

```
EntityManager em = emf.createEntityManager();
List<Department> depts = em.createNamedQuery("all_departments").getResultList();
for(Department dept : depts){
   System.out.println(dept.getName());
   List<Employee> es = dept.getEmployees();
   for(Employee e: es){
       System.out.println(e);
       }
 }
em.close();
```

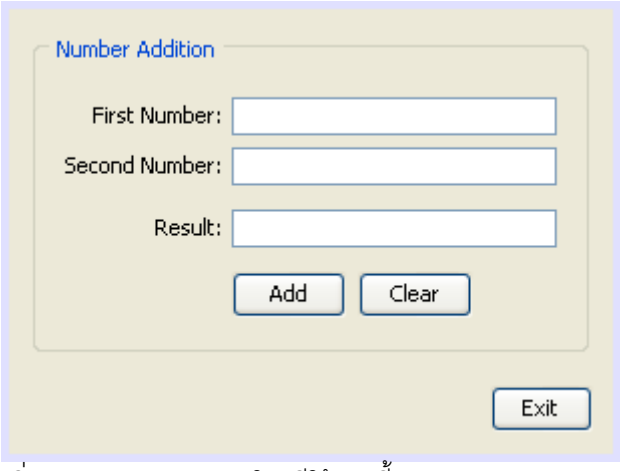
สั่งคอมไพล์และรัน หลังจากนั้นสังเกตุการแสดงผลข้อมูลของโปรแกรม Netbeans เทียบกับข้อมูลที่อยู่ในฐานข้อมูล

7 SWING

ในบทนี้จะเป็นการพัฒนาโปรแกรมที่มี Graphic User Interface (GUI) โดยใช้งาน Swing Application Framework ซึ่ง Netbeans มีเครื่องมือที่ช่วยให้การพัฒนาโปรแกรมในลักษณะที่มีการทำงาร่วมกับ GUI ได้ง่าย และสะดวกมากยิ่งขึ้น ทั้งในส่วนที่เป็นการสร้าง components ต่างๆ และการจัดหน้าต่างโปรแกรม

7.1 โปรแกรม Add

1. ทำการสร้างโปรเจ็คให้ชื่อว่า *number_addition*
2. สร้าง *JFrame Form* ชื่อว่า *NumberAdditionUI* และใส่ *my.numberaddition* ในช่องแพ็คเก็จ
3. สร้าง *JPanel* ขึ้นมา โดยเซ็ต Border เป็น Titled Border และให้มี title เป็น *Number Addition*
4. สร้าง *JLabel*, *JTextField* และ *JButton* ดังรูป โดยในส่วนของ *JTextField* นั้นให้มีชื่อเป็น *firstTextField*, *secondTextField* และ *resultTextField* ตามลำดับ



5. ในปุ่ม Exit ให้เพิ่ม *ActionPerformed* โดยมีโค้ดดังนี้

```
System.exit(0);
```

6. ในปุ่ม Clear ให้เพิ่ม *ActionPerformed* โดยมีโค้ดดังนี้

```
firstTextField.setText("");
secondTextField.setText("");
resultTextField.setText("");
```

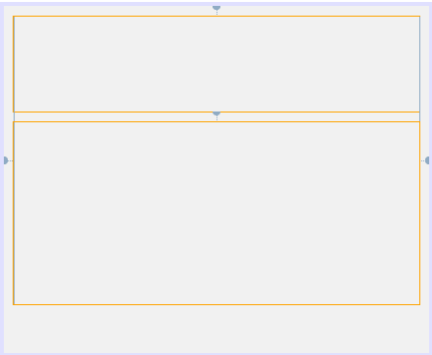
7. ในปุ่ม Add ให้เพิ่ม *ActionPerformed* โดยมีโค้ดดังนี้

```
float num1, num2, result;
num1 = Float.parseFloat(firstTextField.getText());
num2 = Float.parseFloat(secondTextField.getText());
result = num1+num2;
resultTextField.setText(String.valueOf(result));
```

8. ทำการรันโปรเจ็ค และทดสอบการทำงานของโปรแกรม

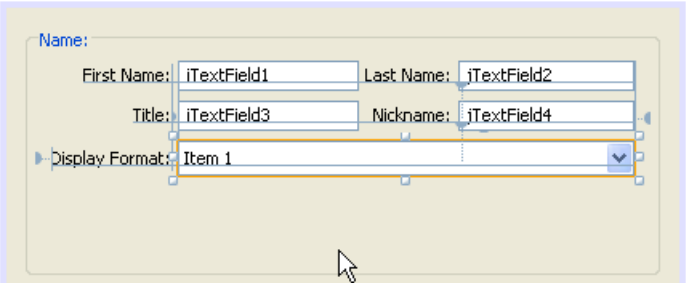
7.2 โปรแกรม CONTACT EDITOR

1. สร้างโปรเจ็คแบบ Java Application ชื่อ *contact_editor* และให้แพ็คเกจเป็น *my.contacteditor*
2. สร้าง *JFrame Form* ชื่อว่า *ContactEditorUI*
3. สร้าง *JPanel* 2 อัน ดังรูป



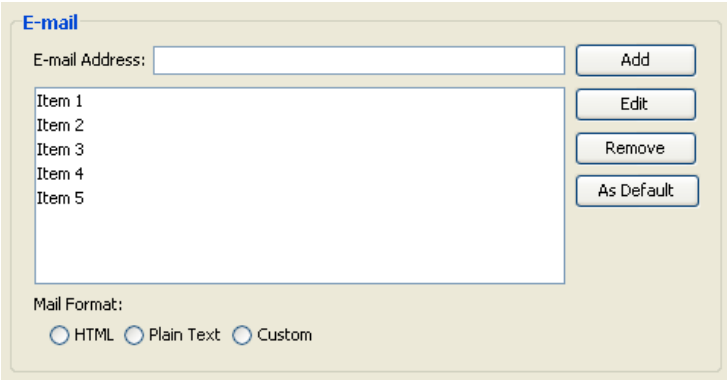
4. ทำการเซ็ต Border ทั้งสองอันให้เป็น Titled Border และให้มี Title เป็น *Name* และ *E-mail* ตามลำดับ
5. ในส่วนของ *JPanel Name* เพิ่มส่วนต่างๆ ดังนี้
   - *JLabel* โดยมีข้อความว่า *First Name :*
   - *JLabel* โดยมีข้อความว่า *Last Name :*
   - *JLabel* โดยมีข้อความว่า *Title :*
   - *JLabel* โดยมีข้อความว่า *Nick Name :*
   - เพิ่ม *JTextField* หลัง *JLabel* แต่ละอัน
   - *JLabel* โดยมีข้อความว่า *Display Format :*
   - เพิ่ม *JComboBox* ด้านหลัง *Display Format :*

   โดยผลลัพธ์สุดท้ายจะได้ดังรูป



6. ในส่วนของของ *JPanel E-mail* เพิ่มส่วนต่างๆ ดังนี้
   - *JLabel* โดยมีข้อความว่า *E-mail Address :*
   - เพิ่ม *JTextField* ต่อด้านหลัง *E-mail Address :*
   - เพิ่ม *JButton* ต่อด้านหลังจาก *JTextField* โดยให้ชื่อว่า *Add*
   - เพิ่ม *JList* ด้านล่างของ *JLabel* ที่ชื่อว่า *E-mail Address* และ *JTextField*
   - เพิ่ม *JButton* จำนวน 3 อันเรียงในแนวตั้งโดยให้ชื่อว่า *Edit, Remove* และ *As Default* ตามลำดับ ด้านล่างของ *JButton Add*
   - เพิ่ม *JLabel* โดยพิมพ์ว่า *Mail Format* : ด้านล่างของ *JList*
   - เพิ่ม *JRadioButton* จำนวน 3 อันในแนวนอนโดยพิมพ์ว่า *HTML, Plain Text* และ *Custom* ตามลำดับ ด้านล่างของ *JLabel*
7. เพิ่ม *JButton* ด้านล่าง *JPanel E-mail* จำนวน 2 อันในแนวนอน โดยพิมพ์ว่า *OK* และ *Cancel*

จัดตำแหน่งของแต่ละ Component ให้เป็นระเบียบ โดยผลลัพธ์สุดท้ายจะได้ ดังรูป



8 JAVA DESKTOP APPLICATION

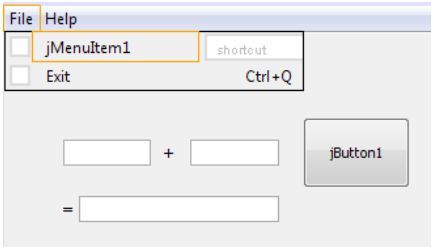
Ch8

จะเห็นได้ว่าการพัฒนาโปรแกรมที่เกี่ยวข้องกับ GUI ในบทข้างต้นนั้น เป็นการแนะนำให้ผู้พัฒนาได้รู้จักกับเครื่องมือที่ใช้ในการพัฒนา GUI เบื้องต้น ดังนั้นในบทนี้จะกล่าวถึงการพัฒนาโปรแกรมในลักษณะ Desktop Application ที่มีการใช้งาน Swing Application Framework ที่มีการใช้เทคนิคที่ทำให้การพัฒนาโปรแกรมทำได้ง่ายและสะดวกยิ่งขึ้น

8.1 การเตรียม GUI

1. สร้างโปรเจ็คชนิด Java Desktop Application ให้ชื่อว่า *desktop*

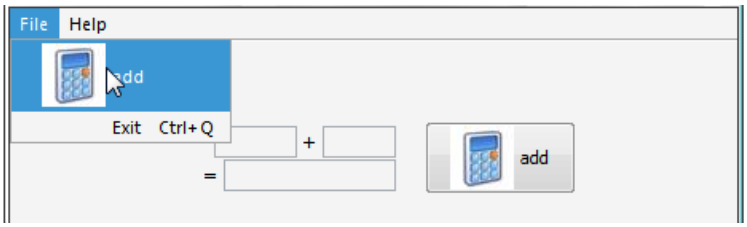
2. ทำการเพิ่มส่วนต่างๆ ดังรูป โดยให้มี *JTextfield* จำนวน 3 อัน โดยมีชื่อว่า *aTextField*, *bTextField* และ *resultTextField* ตามลำดับ และเพิ่ม *JButton* ให้มีข้อความว่า *Add*



3. ทำการสร้างเมนูบาร์ และเพิ่ม *Menu Item* โดยใช้ฟังก์ชัน Add from Palette

8.2 กำหนด Action

1. ให้ทำสร้าง Action ให้กับปุ่ม และเมนู โดยกำหนดชื่อของ Action เป็น *doAdd* ซึ่งจะได้ผลลัพธ์ ดังรูป



2. กำหนดโค้ดให้กับ action ดังนี้

```
int a = Integer.parseInt(aTextField.getText());
int b = Integer.parseInt(bTextField.getText());
int c = a+b ;
resultTextField.setText(String.valueOf(c));
```

3. ทำการรันโปรแกรม ทดสอบการบวกเลข จากทั้งปุ่ม และเมนูไอเทม

8.3 กำหนด Action ให้ทำงานเบื้องหลัง

1. ทำการ Set Action ของปุ่ม Add โดยให้ชื่อ Action's method ว่า *doAddInBackground* และช่อง Text ให้ใส่ข้อความว่า *add*
2. กำหนดให้เมนูไอเทม *add* ใช้งานแอคชั่น *doAddInBackground*
3. เพิ่มฟิลด์ต่อไปนี้ในคลาส *DoInBackgroundTask*

```
int c ;
```

4. ในคอนสตรัคเตอร์ของคลาส *DoInBackgroundTask* ให้เพิ่มโค้ดต่อไปนี้

```
int a = Integer.parseInt(aTextField.getText());
int b = Integer.parseInt(bTextField.getText());
c = a + b ;
```

5. เพิ่มโค้ดต่อไปนี้ลงในเมธอด *doInBackground*

```
try{
    Thread.sleep(5000);
} catch(Exception e){}
```

6. ในเมธอด *succeeded* ให้เพิ่มโค้ดต่อไปนี้

```
resultTextField.setText(String.valueOf(c));
```

7. ทดลองรันเพื่อดูผลลัพธ์

8.4 กำหนด Enable Property

1. กำหนด Enabled Property ของ *doAddInBackground* เป็น *ready* และเปลี่ยนค่าบูลีนของ *ready* เป็น true
2. เพิ่มโค้ดต่อไปนี้ลงในเมธอด *doAddInBackground*

```
setReady(false);
```

3. ในเมธอด *succeeded* ให้เพิ่มโค้ดต่อไปนี้

```
setReady(true);
```

ทำลองรันเพื่อดูผลลัพธ์

9 JAVA DESKTOP DATABASE APPLICATION

Ch9

การพัฒนาพัฒนาโปรแกรมในลักษณะ Desktop Application นั้นในบางครั้งก็หลีกเลี่ยงไม่ได้ที่จะต้องทำการติดต่อกับฐานข้อมูล ไม่ว่าจะเป็นในเรื่องของการสร้าง การอ่าน การปรับปรุงและการลบข้อมูลในฐานข้อมูล ดังนั้นบทนี้จะกล่าวถึงการใช้ Netbeans ในการพัฒนาโปรแกรมลักษณะ Desktop Application ที่มีการทำงานติดต่อกับฐานข้อมูล

9.1 เตรียมพร้อมฐานข้อมูล

1. ทำการตรวจสอบว่าได้มีการกำหนด Database Path แล้วหรือยัง โดยในหน้าต่าง Services เลือก Databases > Java DB เลือก Properties
2. ทำการสตาร์ทเซริฟเวอร์ฐานข้อมูล โดยหากไม่เกิดข้อผิดพลาด ผลที่ได้จะเป็นดังนี้

```
Apache Derby Network Server - 10.2.2.0 - (485682) started and ready to accept
connections on port 1527 at 2007-09-05 10:26:25.424 GMT
```

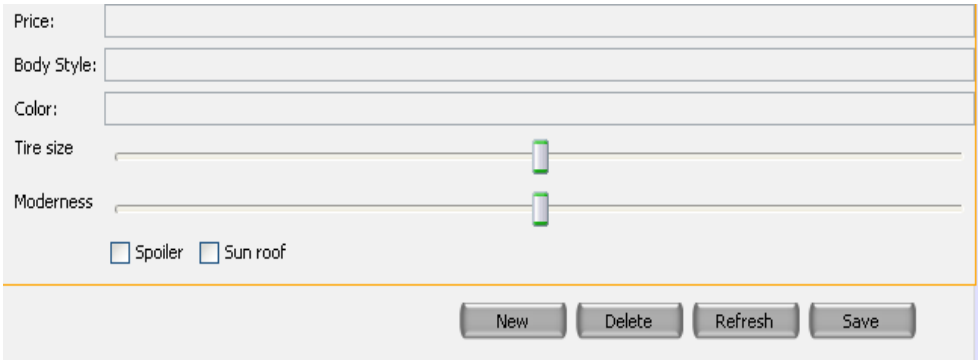
3. สร้างฐานข้อมูลโดยใช้ฟังก์ชัน Create Database บน Java DB และให้ชื่อเป็น *car_database* และกำหนดชื่อผู้ใช้ และรหัสผ่าน
4. ทำการเชื่อมต่อกับฐานข้อมูล *car_database* โดยกำหนด Connection URL เป็น *jdbc:derby://localhost:*1527/*car_database*
5. เลือกที่โหนด Table คลิ๊กขวา และเลือก Execute Command โดยสำเนาโค้ดที่อยู่ใน *car.sql*
6. สร้างโปรเจ็คโดยให้ชื่อว่า *cars_app* โดยเลือกเป็นแบบ Java Desktop Application จากนั้นเลือก Database Application เมื่อ Next แล้วเลือกเชื่อมต่อกับ *car_database* โดยเลือกอันที่มีลักษณะคล้ายๆกับตัวอย่างนี้

```
jdbc:derby://localhost:1527/car_database[root on ROOT]
```

7. ให้ช่อง Available Columns มีสมาชิกตั้งแต่ *SUN_ROOF* ไปจนถึง *MODERNNESS*
8. สั่งรันโปรเจ็คจะขึ้นหน้าต่างขึ้นมา
   - ในหน้าต่างนั้นลองเลือกเรคคอร์ดแรก และปรับเปลี่ยน Price
   - คลิ้กปุ่ม New เพื่อสร้างเรคคอร์ดเพิ่ม

9.2 การเพิ่มคอนโทรลขั้นต้น

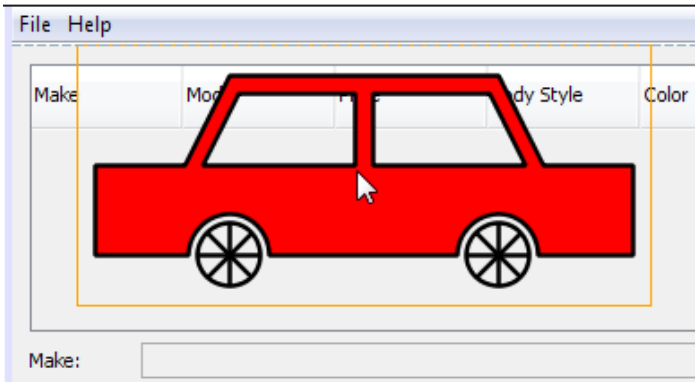
1. เพิ่มสไลด์เดอร์ 2 อัน และให้มีชื่อว่า *Tire size* และ *Moderness*
2. เพิ่มเช็คบ๊อก 2 อัน ให้ชื่อว่า *Spoiler* และ *Sun Roof* ดังในรูป



3. เลือกสไลด์เดอร์ Tire size และใช้ฟังก์ชัน Bind > value ทำการเลือก Binding Source เป็น *masterTable* และในส่วนของเลือก Binding Expression เป็น *tireSize*
4. ทำเช่นเดียวกันกับสไลด์เดอร์ *Morderness* แต่เลือก Binding Expression เป็น *morderness*
5. เลือกเช็คบ๊อก Spoiler ใช้ฟังก์ชัน Bind > selected เลือก Binding Source เป็น *masterTable* และเลือก Binding Expression เป็น *spoiler*
6. ทำเช่นเดียวกันกับเช็คบ๊อก Sun roof แต่เลือก Expression เป็น *sunRoof*

9.3 การเพิ่มคอนโทรลขั้นสูง

1. ทำการ Open Project ที่มีชื่อว่า *CarPreview*
2. ลากคลาส *CarPreview.java* มาใส่ในฟอร์มเหนือตารางดังรูป



3. คลิ๊กขวาที่รูปรถ และใช้คำสั่ง Bind > color เลือก Binding Source เป็น *masterTable* และเลือก Binding Expression เป็น *color*
4. ทำเช่นเดียวกันกับข้อ 3 กับ *morderness*, *tireSize*, *spoiler* และ *sunRoof* โดยเลือก Binding Expression ตามการ Bind ของแต่ละอัน
5. สั่งคอมไพล์ และรันโปรเจ็ค *cars_app* แล้วทดลองปรับ ค่าต่างๆ ที่ได้ทำการ Bind เอาไว้ข้างต้น

10 การใช้งาน PROFILING

1. ทำการแตกไฟล์ *treasure.zip* ไปที่ c:\
2. ทำการเปิดโปรเจ็ค *google_hunt* ขึ้นมา
3. ที่โปรเจ็ค *google_hunt* เปิดหน้าต่าง Profile จากนั้นเลือก Enable threads monitoring จากเมนู Monitor และรันโปรแกรมจากปุ่ม Run ภายในหน้าต่าง Profile หลังจากนั้นเลือกแท็บต่อไปนี้
    - Thread (Timeline)
    - Threads (Details)
4. ที่โปรเจ็ค *google_hunt* เปิดหน้าต่าง Profile จากนั้นกำหนด Filter: ให้มีค่าเป็น Profiles All Classes จากเมนู CPU และรันโปรแกรมจากปุ่ม Run ภายในหน้าต่าง Profile หลังจากนั้นเลือกแท็บต่อไปนี้
    - Call Tree
    - Hot Spots
    - Combined
5. ที่โปรเจ็ค *google_hunt* เลือก Profile ไปที่ Memory และรันโปรแกรมจากปุ่ม Run ภายในหน้าต่าง Profile