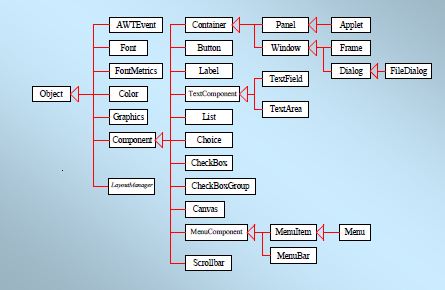
**SWING**

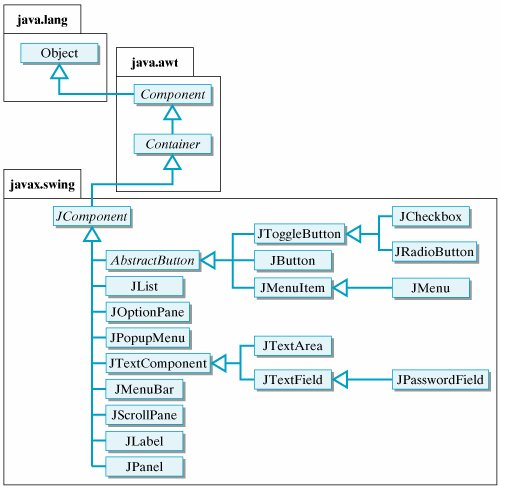
Бид өмнө AWT –ын классуудаар энгийнээр хэрхэн график дүрслэлийг үүсгэхийг үзсэн. Цаашид бид AWT –ын component –үүдийг график дүсрлэлийг хийхдээ өргөнөөр ашиглахгүй. Яагаад гэвэл одоо үед Swing болон JavaFX өргөн хэрэглэж байгаа билээ. Доор AWT –н component –үүдийн шаталсан харьцааг үзүүлэв.



Awt классийн компонентуудыг хүнд буюу **heavyweight** гэдэг бол **Swing** компонентуудыг **lightweight** буюу хөнгөн компонент гэдэг. Ингэж нэрлэсний учир нь swing компонентуудын код нь цэвэр **java** ашиглан бичигдсэн бол **awt** компонентуудын кодонд **native code** оролцсон байдаг (үйлдлийн системийн код).

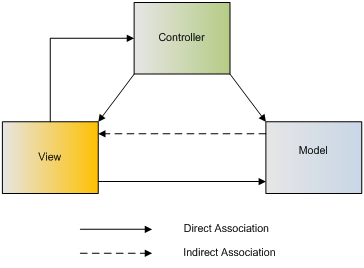
Swing нь 1997 онд Sun компани нь Swing нэртэй төсөл хэрэгжүүлснээр swing үүссэн юм. Swing нь маш энгийн JLabel –с эхлээд нэлээд төвөгтэй JTable, JTree зэрэг компонентуудыг агуулдаг. Swing нь ажиллаж буй үйлдлийн системээсээ огт хамааралгүйгээр 100% java дээр бичигдсэн тул бүх үйлдлийн системүүд дээр ижилхэн харагдаж ажиллах чадвартай байдаг. Awt component -үүдийн нэрний өмнө J үсэг залгасан байвал Swing-ийн компонент гэж таньж болно. Доор Swing –н зарим пакежууд харууллаа.

* **javax.swing:** хамгийн үндсэн swing компонентуудыг агуулна. Ихэнх классууд энд агуулагддаг.
* **javax.swing.border:** Хүрээг тодорхойлоход хэрэглэгдэх класс болон интерфэйсүүдийг агуулна. Хүрээнүүд нь өөрсдөө компонент биш тул хэдэн ч тооны компонент дунд нэг хүрээ хэрэглэгдэж болно.
* **javax.swing.colorchooser:** JColorChooser өнгө сонгогч класст хэрэглэгдэх классуудыг агуулна.
* **javax.swing.event:** Swing холбоотой бүх төрлийн event-үүдийг агуулна.
* **javax.swing.filechooser**: Файл сонгоход хэрэглэгддэг JFileChooser компонентод хэрэглэгдэх класс болон интерфэйсүүдийг агуулна.
* **javax.swing.plaf:** Програмын харагдах байдлыг тодорхойлогч абстракт класс болон интерфэйсүүд агуулагддаг (look and feel). Эдгээр Metal, Basic, Motif зэрэг хувилбаруудад хэрэглэгддэг. Програмист энэ классуудыг имплемент хийн шинэ харагдах байдлыг өөрөө үүсгэж болно.
* **javax.swing.plaf.basic:** Дээрх абстракт классуудыг имплемент хийсэн Basic хувилбар нь энд байдаг. Програмист эдгээр классуудаас удамшуулан өөрийн хүссэн харагдах байдлыг үүсгэж авч болно.
* **javax.swing.metal:** Metal хувилбар энд байрлана. Java-н default харагдах байдал юм.
* **javax.swing.multi:** Multiplexing харагдах байдал. Энэ нь шинэ харагдах байдал биш бөгөөд харин бусад харагдах байдлуудыг зэрэгцүүлэн хэрэглэх боломжийг олгодог.
* **javax.swing.table:** Хүснэгт буюу JTable компонентод хэрэглэгдэх класс болон интерфэйсүүдийг агуулна.
* **javax.swing.text:** Текст компонентуудад хэрэглэгдэх классуудыг агуулдаг.
* **javax.swing.text.html:** HTML документ үүсгэх, задлах, харах зэрэгт хэрэглэгдэх класс болон интерфэйсүүдийг агуулна.
* **javax.swing.text.html.parser:** HTML –г задлахад шаардлагатай классуудыг агуулна.
* **javax.swing.text.rtf:** Rich Text Document буюу сайжруулсан текст документ компонентод хэрэглэгдэх класс интерфэйсүүдйи агуулна. Энэ компонент дотор байгаа текстүүд нь янз бүрийн өнгөтэй, ташуу, зураастай гэх мэтээр олон хэлбэрээр бичих боломжтой байдаг.
* **javax.swing.tree:** Үечилсэн хэлбэртэй өгөгдлийг дүрслэхэд хэрэглэгддэг мод буюу JTree компонентод хэрэгтэй класс болон интерфэйсүүдийг агуулдаг.
* **javax.swing.undo:** Undo, Redo функцүүдийг хийхэд шаардлагатай класс болон интерфэйсүүдийг агуулна.



**MVC –(Model View Controller)**

MVC загвар нь 1970 оноос эхлэлтэй обьект хандалтат дэлгэцийн загварчлал юм. Энэ загвар ёсоор бол компонентийг гурван хэсэгт хуваадаг(model, view, controller). Swing-н бүх компонентууд нь энэ загварын сайжруулагдсан хэлбэрээр бүтээгдсэн.



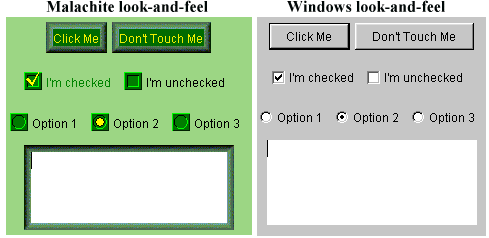
**Model:** Model нь компонентийн төлөвийг хариуцдаг. Жишээ нь товч дарагдсан байгаа эсэх, radio товч сонгогдсон эсэх, list-д ямар элементүүд байгаа аль нь сонгогдсон байна гэх мэт. Model нь view болон controller-тойгоо шууд бусаар харилцдаг. Шууд бус гэсний учир нь компонентийн модел нь өөрийн view болон controller-н талаарх мэдээллийг өөртөө хадгалдаггүй. Компонентийн төлөв өөрчлөгдвөл уг өөрчлөлтийн талаар мэдээлэл ерөнхийд нь цацдаг (event). Тасархай шугамаар энэ харилцааг харуулсан байна.

**View:** View нь компонентийн гаднаасаа ямар харагдахыг тодорхойлдог (“look”). Өөрөөр хэлбэл гаднах байдал нь юм. Жишээ нь компонентийн одоогийн өнгө, хүрээ, фонт, товойсон байдал зэргийг дүрсэлнэ. View нь моделоос ирсэн шууд бус мэдээ болон Controller –с ирсэн шууд мэдээнд үндэслэн харагдах байдлыг тааруулж өөрчлөх үүрэгтэй.

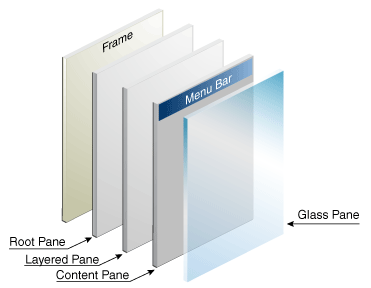
**Controller:** Controller нь гаднаас орж ирсэн мэдээнд (хулгана, гар г.м) компонент хариу үйлдэл үзүүлэх эсэхийг тодорхойлж өгдөг. Тиймээс Controller –г компонентийн мэдрэхүй (“feel”) гэж хэлж болно. Энэ нь компонентийг хэрэглэх үед ямар үйлдэл хариу үзүүлэхийг тодорхойлно. Controller нь model-с шууд бус мэдээ, view-с шууд мэдээ хүлээн авч болно.

Жишээ нь checkbox-г check хийлээ гэж бодъё. Хэрэв контроллер үүнийг мэдрэх юм бол view рүү энэ тухай мэссэж илгээж болно. Хэрэв view үүнийг мэдэрвэл тэрээр модел руу мэссэж илгээнэ. Модел нь уг мэссэжийг хүлээж аваад тохирох ёсоор өөрийн төлөвийг өөрчлөнгүүт уг өөрчлөлтийг зарладаг. Үүнийг нь хүлээж аваад view нь өөрийн байдлыг шинэ төлөв дээрээс үндэслэн өөрчилнө.

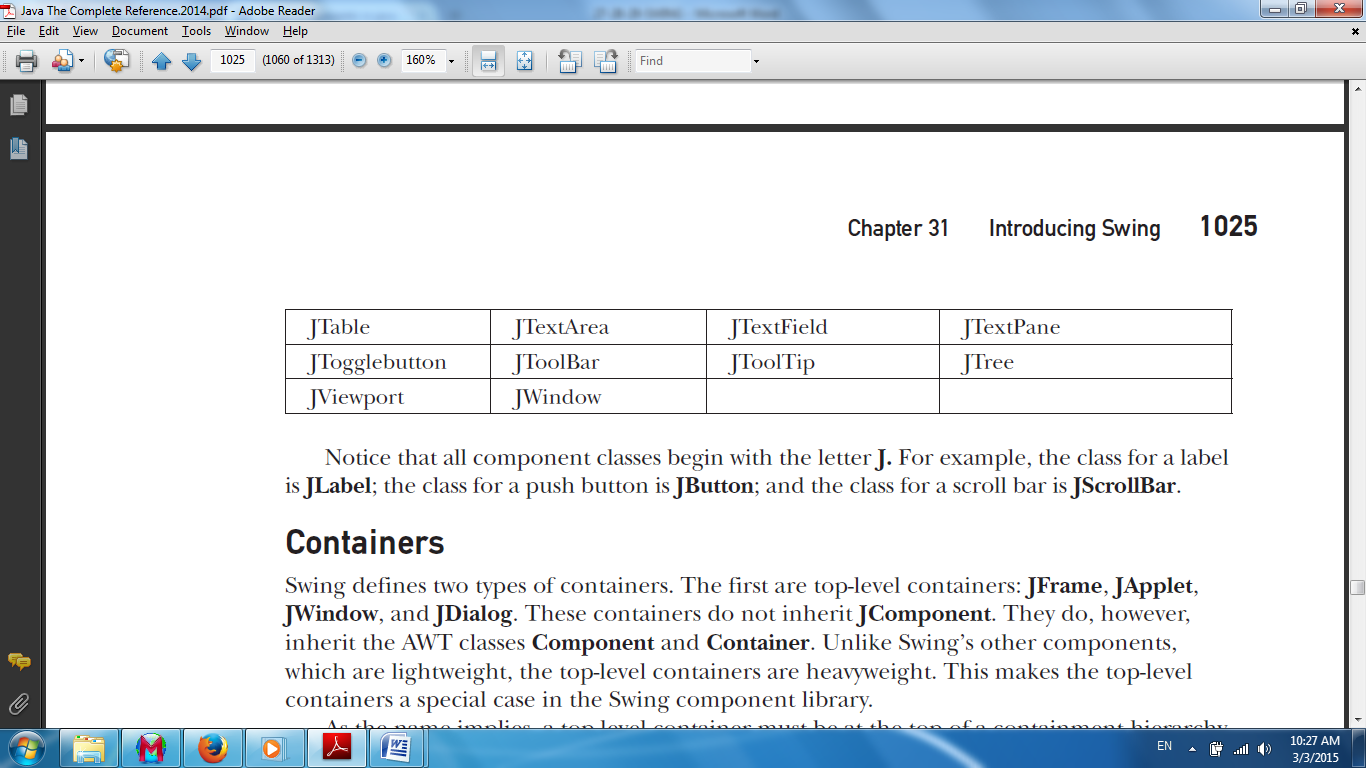
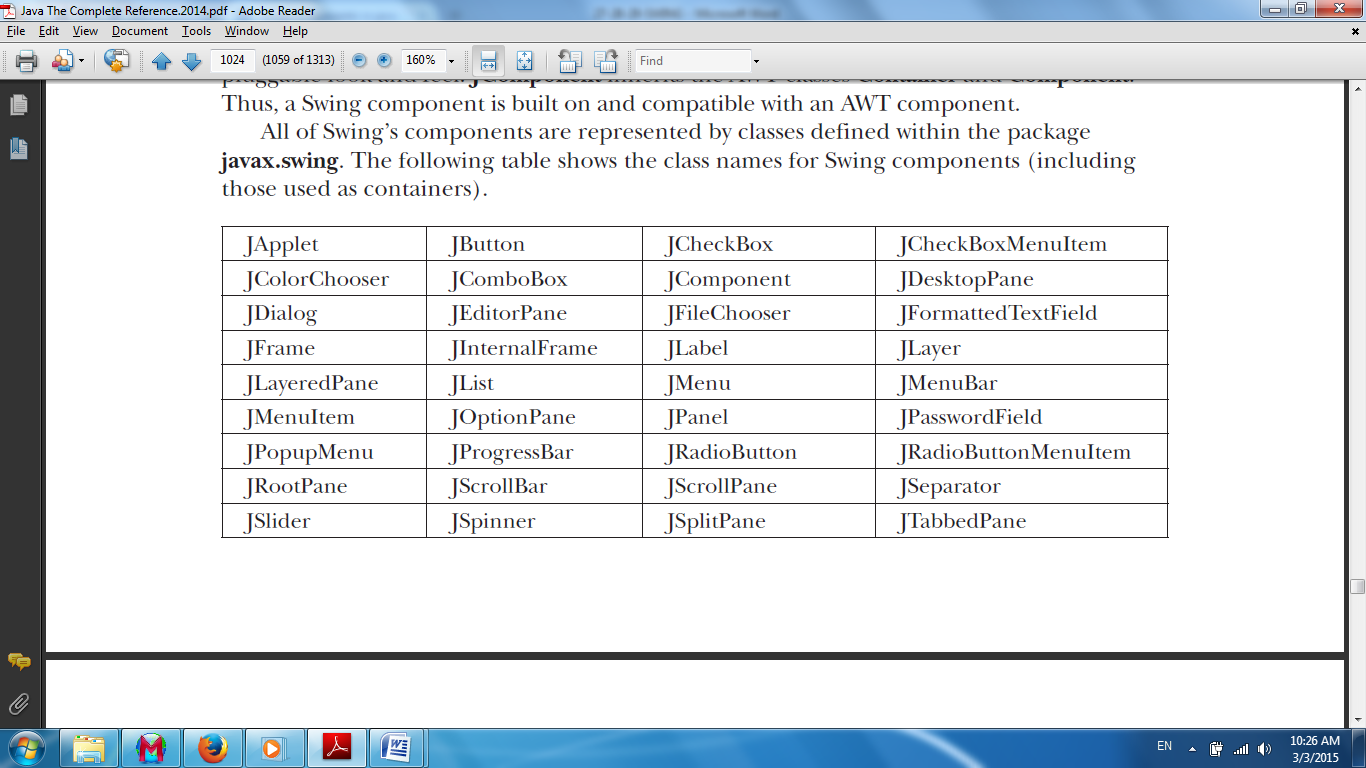
MVC загвар нь model, view, controller-г бие даасан байх боломжийг олгож байгаа учраас нэг модел дээр олон view, controller байх боломжийг олгодогоороо давуу талтай. Swing-ийн MVC загварийн ачаар бид компонентийн “харагдах байдал болон мэдрэмж”-г (“look and feel”) моделийг нь өөрчлөлгүйгээр өөрчлөх боломжтой болсон. Swing-н компонент бүр моделтой бөгөөд түүнийг нь getModel() функцээр авч ажиллаж болно. Компонентийн моделтой ажиллах замаар компонентийн дотор яг юу байгаа яаж ажиллаж байгааг бүрэн мэдрэх боломжтой. Мөн бид компонентуудын моделийг өөрчлөн setModel() функцээр өөрийн хийсэн моделийг хийж өгөх замаар компонентийн үйл ажиллагааг өөрийнхөөрөө өөрчлөх боломжтой. Үүгээрээ Java нь маш гайхамшигтай уян хатан байдлыг бий болгодог. Java-г ашиглан түүнд байгаа бүх зүйлийг бид хүссэнээрээ өөрчлөх боломжтой байдаг нь түүний том давуу тал юм.



Swing нь components болон container гэсэн хоёр үндсэн хэсэгээс бүрддэг. Энэ хоёрын хоорондын ялгаа нь гэвэл Components нь удирдлагаас биеэ даасан зүйл, Containers нь бүлэг component –үүдийг дотроо агуулдагт оршиж байга юм. Container нь top-level-container болон general-purpose container гэсэн хоёр төрлийн Container –тэй. Swing нь JFrame, JDialog, JApplet гэсэн гурван төрлийн top-level container –тэй. Үүн дээр нэмэгдээд JWindow container тэй. JWindow container нь ямар нэг тодорхой цонхыг харуулдаггүй. general-purpose container нь жишээ нь JPanel, JLayeredPane, JDesktopPane, JInternalFrame байдаг. Swing component –үүд нь JComponent классаас үүсэлтэй ба энэ класс нь бүх component –үүдийг ерөнхийд нь бэлтгэж өгдөг. JComponent класс нь AWT –н классууд болох Container болон Component классуудаас удамшдаг. Swing –н бүх component –үүд нь javax.swing гэсэн пакежд агуулагддаг. Тиймээс бид swing –н component –үүдийг хэрэглэхдээ swing –н пакежийг импорт хийж оруулж өгөх ёстой. top-level container болгон root pane гэж дууддаг дундын container –тэй. Root pane нь дөрвөн хэсэгтэй байдаг: GlassPane, LayeredPane, ContentPane, MenuBar гэсэн. RootPane нь бусад бүх агуулагчдыг дотроо агуулдаг. LayeredPane энэ дотроо ContentPane болон MenuBar –г агуулдаг. Цонхонд MenuBar –г нэмхэд ContentPane –д нэмдэггүй учир нь contentPane болон menuBar хоёр нь нэг төвшинд байдаг. Тиймээс MenuBar –г нэмэхдээ LayeredPane –д нэмдэг. ContentPane нь цонхонд дүрслэгдэх үндсэн component –үүдийг (menuBar –с бусад) агуулах үүрэгтэй. GlassPane нь бусад бүх агуулагчдын дээр оршиж шил мэт байдаг. Харагддаггүй боловч Paint функцийг нь дахин тодорхойлсноор энэ агуулагчийг харагддаг болгож болно. Үүний хэрэглээ нь component –үүд дээрх хулганы үзэгдлийг түрүүлж барих, component дээр өөр component байрлуулах зэрэгт хэрэглэж болно. Доор зураг байдлаар давхаргуудыг үзүүллээ.



Доор бүх swing –н component –үүдийг хүснэгт хэлбэрээр үзүүллээ.



**Жишээ**

**package** swing;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.SwingUtilities;

**class** SwingDemo {

SwingDemo(){

// Create a new JFrame container.

JFrame jfrm = **new** JFrame("A Simple Swing Application");

// Give the frame an initial size.

jfrm.setSize(275, 100);

// Terminate the program when the user closes the application.

jfrm.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

// Create a text-based label.

JLabel jlab = **new** JLabel(" Swing means powerful GUIs.");

// Add the label to the content pane.

jfrm.add(jlab);

// Display the frame.

jfrm.setVisible(**true**);

}

**public** **static** **void** main(String args[]) {

// Create the frame on the event dispatching thread.

SwingUtilities.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**new** SwingDemo();

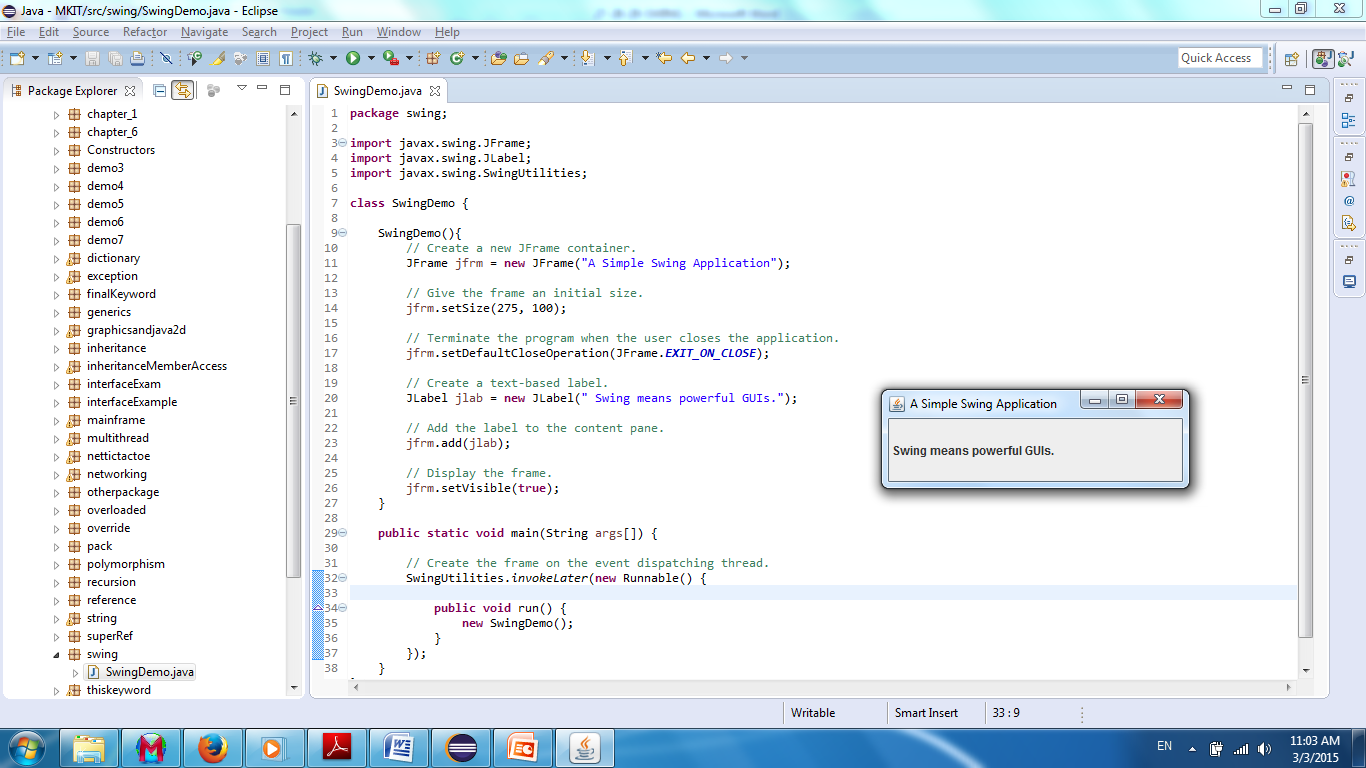
}

});

}

}

**Үр дүн**



Тайлбар:

Доорх код нь гарчигтай, нээх, хаах, томсгох, багасгах төлөвтэй jfrm нэртэй тэгш өнцөгт агуулагч цонх үүсгэж байна.  
JFrame jfrm = **new** JFrame("A Simple Swing Application");

Тухайн цонхыг 275 –н өргөнтэй, 100 –н өндөртэй хэмжээ өгч байна.  
jfrm.setSize(275, 100);

Тухайн цонхыг хааж програмыг дуусгаж байна. Тухайн цонхыг хаахад  
jfrm.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
setDefaultCloseOperation() функц доторх дараах дөрвөн тохиргооны аль нэгийг өгч болно. **DISPOSE\_ON\_CLOS** – Container –н доторх component –үүдийг болон уг Container –г санах ойгоос чөлөөлнө.  
**DO\_NOTHING\_ON\_CLOSE** – Цонхыг хаах товчийг дархад юу ч хийгдэхгүй.   
**HIDE\_ON\_CLOSE** – Container –г үзэгдэхгүй болгоно.  
**EXIT\_ON\_CLOSE** – Цонхыг хааж, програмыг дуусгана.

Доорх код нь swing JLabel component –г үүсгэж байна.  
JLabel jlab = **new** JLabel(" Swing means powerful GUIs.");

Тухайн цонхонд тухайн frame –н content pane дээр JLabel –г нэмж байна.  
jfrm.add(jlab);

Анхааруулахад JDK5 өмнөх хувилбаруудад content pane дээр component –г нэмхэд JFrame дээрээс шууд add() функцийг дуудаж болдоггүй байсан. Харин үүний оронд JFrame –н объектийн content pane дээр add() функцийг дуудах хэрэгтэй. Тухайн content pane –г JFrame –н тухайн нэг тохиолдол дээр getContentPane() гэсэн функцээр гаргаж ирж болно. бичигдэх хэлбэр нь: Container getContentPane() энэ код нь content pane –д Container –н заагчийг буцаадаг. Тиймээс өмнөх бичиглэл нь jfrm.getContentPane( ).add( jlab );

Дээрх бичиглэлийн getContentPane( ) функц нь эхлээд content pane –н заагчийг хүлээн авах ба үүний дараагаар add( ) функц нь тухайн pane –дээр container –рүү холбогдон component –г нэмдэг. Энэ арга нь content pane –д зориулж layout manager –г тавихад setLayout( ) болон component –г арилгахын тулд remove( ) функцийг дуудаж ажиллуулах шаардлагатай байдаг байсан. Танд JDK5 -аас өмнөх хувилбараас getContentPane( ) функцийг дуудах аргыг тодорхой харагдах болно. Тиймээс getContentPane( ) –нь цаашид бидэнд шаардлаггүй. Та энгийнээр шууд JFrame дээрээс add( ), remove( ), setLayout( ) функцүүдийг дуудаж болно яагаад гэвэл эдгээр функцүүд нь content pane автоматаар үйлдэл хийгдэхэд өөрчлөгддөг юм.

Доорх бичиглэл нь цонхыг гаргах боломжийг бий болгодог.  
jfrm.setVisible(**true**);

setVisible( ) функц нь AWT Component классаас удамшдаг. Хэрвээ аргумент нь true байвал цонхыг гаргах болно үгүй бол цонхыг нуух болно. Энэ функц нь анхын утгаараа харагддаггүй байдаг. Тиймээс бид энэ функцийг setVisible(true) –ээр дуудах ёстой. Main ( ) функц доторх код нь SwingDemo объектийг үүсгэж, цонхон дээр label –г үзүүлнэ. Анхааруулахад SwingDemo байгуулагч функцийг дараах мөр кодуудаар дуудаж ажиллуулна.

SwingUtilities.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**new** SwingDemo();

}

});

Main( ) функц нь эхлээд main тридийг ажиллуулдаг. Шууд SwingDemo –н объектийг үүсгэдэггүй. Оронд нь энэ объектийг үүсгэх Runnable объектийг үүсгэх ёстой. Тридийн үзэгдлийг дамжуулахдаа дараах SwingUtilities классын хоёр функцийн аль нэгийг нь хэрэглэх ёстой. Эдгээр функцүүд: **static void invokeLater(Runnable *obj*)**, **static void invokeAndWait(Runnable *obj*)**

**Event Handling**

Дээрх жишээнд энгийн Swing цонх үүсгэсэн. Гэвч бид нэг чухал зүйл болох event handling –г орхигдуулсан байна. Яагаад гэвэл JLabel бол хэрэглэгчээс оролтын ямар нэг өгөгдөл авдаггүй ба цонхон дээр зүгээр тогтмол текст болон зурагийг гаргах зориулалттай. Тийм учираас энэ Component –д ямар нэг event хэрэггүй ба event үүсгэдэггүй юм. Swing –н бусад component –үүд нь event –тэй байдаг ба эдгээр swing –н event –тэй холбоотой классууд нь javax.swing.event гэсэн пакежд агуулагддаг. Swing –н event –үүдийг барьж авахын тулд AWT –н event –г барьж авдагтай адилаар кодчилох юм. Санах зүйл: бид цаашид график дүрслэлийг хийхдээ Swing –н классуудыг ашиглана.

**Жишээ**

**package** swing;

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** javax.swing.\*;

**class** EventDemo {

JLabel jlab;

EventDemo() {

// Create a new JFrame container.

JFrame jfrm = **new** JFrame("An Event Example");

// Specify FlowLayout for the layout manager.

jfrm.setLayout(**new** FlowLayout());

// Give the frame an initial size.

jfrm.setSize(220, 90);

// Terminate the program when the user closes the application.

jfrm.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

// Make two buttons.

JButton jbtnAlpha = **new** JButton("Alpha");

JButton jbtnBeta = **new** JButton("Beta");

// Add action listener for Alpha.

jbtnAlpha.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent ae) {

jlab.setText("Alpha was pressed.");

}

});

// Add action listener for Beta.

jbtnBeta.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent ae) {

jlab.setText("Beta was pressed.");

}

});

// Add the buttons to the content pane.

jfrm.add(jbtnAlpha);

jfrm.add(jbtnBeta);

// Create a text-based label.

jlab = **new** JLabel("Press a button.");

// Add the label to the content pane.

jfrm.add(jlab);

// Display the frame.

jfrm.setVisible(**true**);

}

**public** **static** **void** main(String args[]) {

// Create the frame on the event dispatching thread.

SwingUtilities.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**new** EventDemo();

}

});

}

}

**Үр дүн**

