Object-Oriented Software Engineering

Team Work - State Diagram Editor

Instructor: Chuen-Min Huang

Group 3

B10223005　林俊安　Jimmy

B10223031　李健銘　Solomon

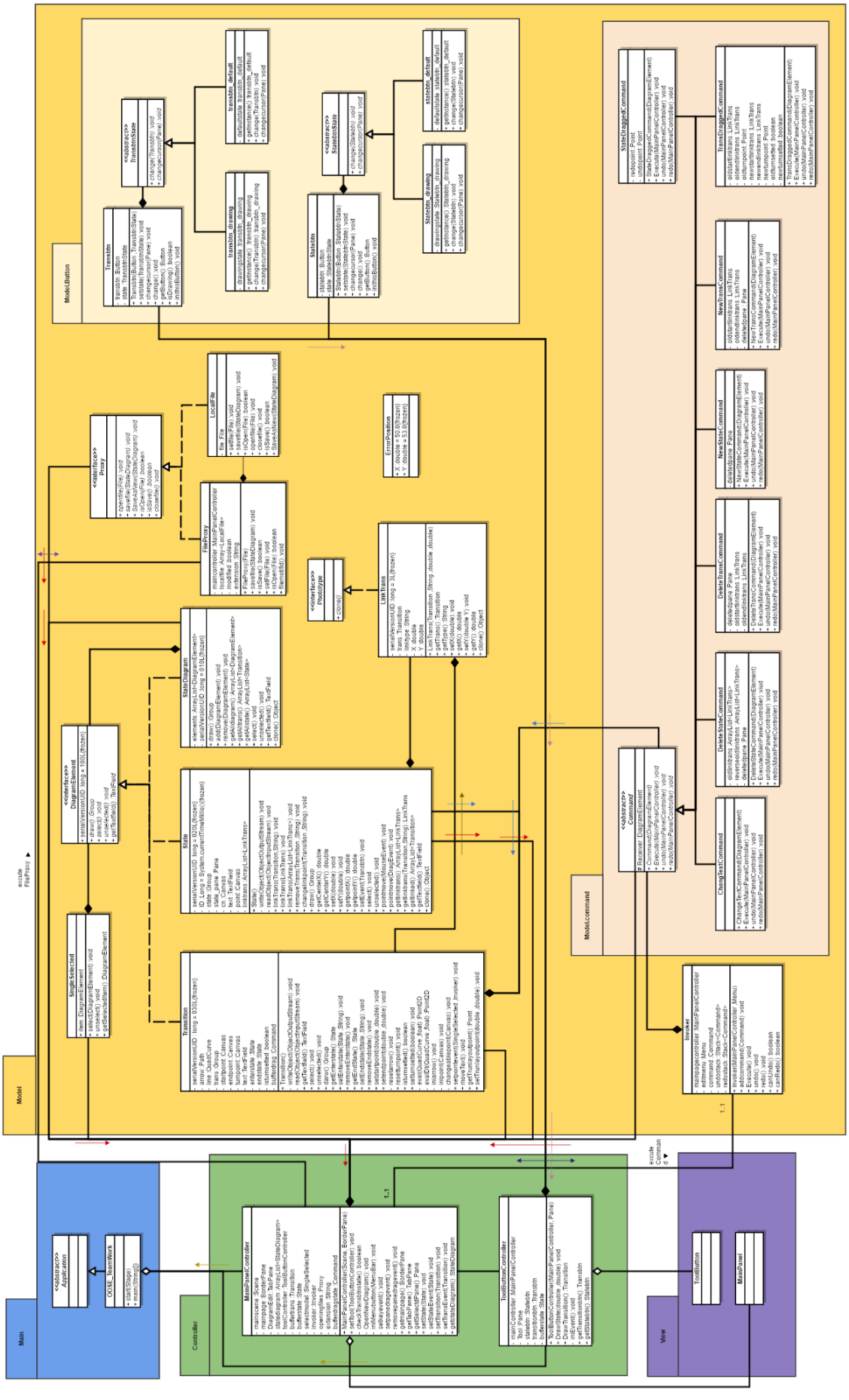
B10223041　黃正文　Dylan

B10223047　張藝亭　Heidi

B10223059　甘宇成　Ray

B10223062　艾正宇　Ashton

Structure


Model
Model.Button
Model.Command
Main
Controller
View
Transition
TransitionState
transition_default
transition_drawing
State
StateState
statebtn_default
statebtn_drawing
StateDiagram
LinkTrans
ErrorPosition
Phototype
LocalFile
FileProxy
Proxy
SingleSelected
DiagramElement
Application
OOSE_TeamWork
MainPanelController
ToolButtonController
Toolbutton
MainPanel
Invoker
Receiver
StateDraggedCommand
NewTransCommand
NewStateCommand
DeleteTransCommand
DeleteStateCommand
ChangeTextCommand
TransDraggedCommand

Problem & Solution :

- Prototype

  Problem : If we have a lot of states and transitions, there will be too much object about the links of those states and transitions that have similar behavior and relationships.

  If we want to undo states or transition , that will be too difficult.

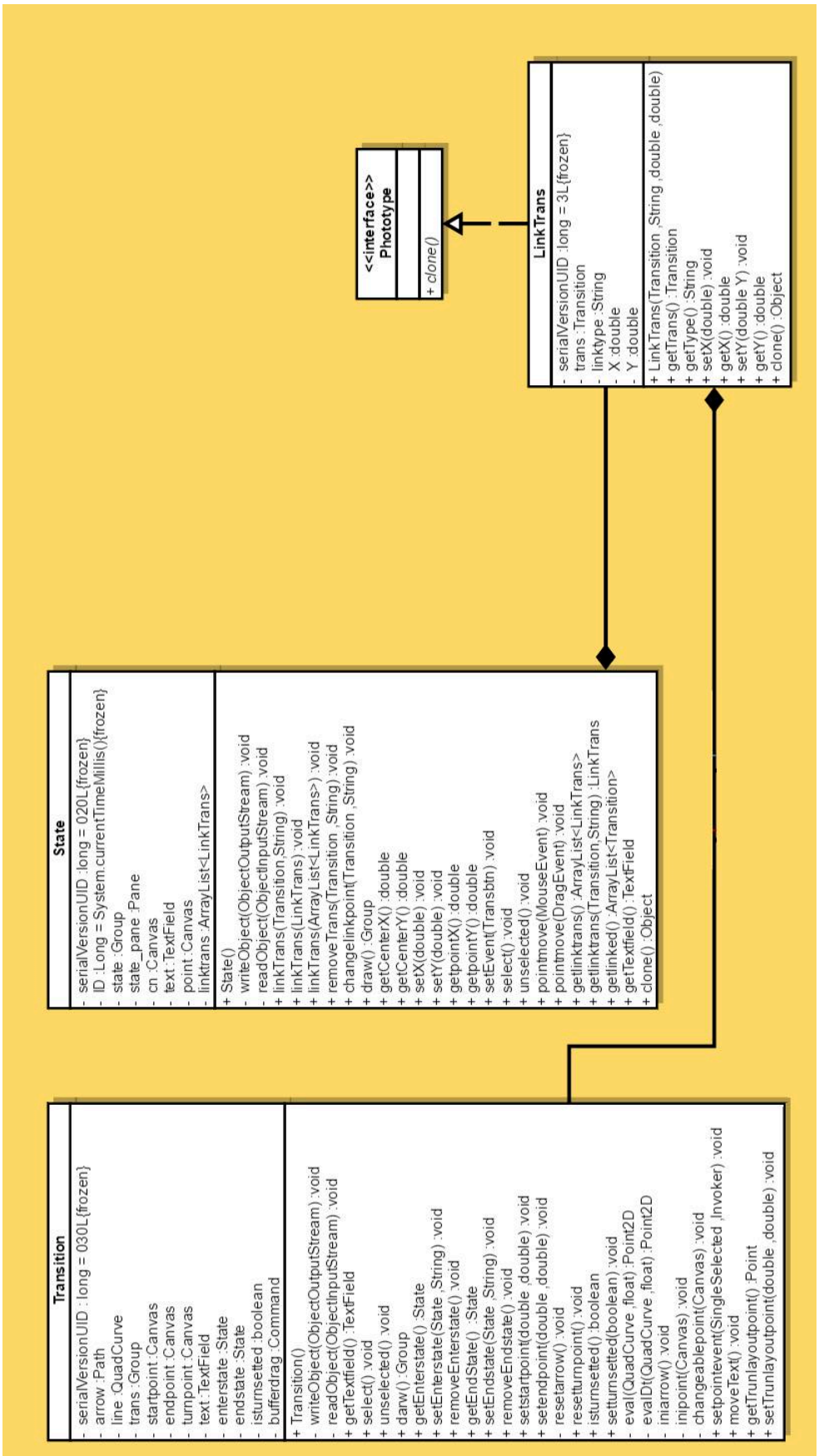
  Solution : Define a class “Prototype” as a interface of the concrete prototype “LinkTrans” to implement the simply operation “clone()”.

  If we want to undo state or transition , just reset LinkTrans that are cloned

Code(Concrete Prototype) :

```
14 public class LinkTrans implements Phototype,Serializable {
15     private static final long serialVersionUID = 3L;
16     private Transition trans;
17     private String linktype;
18     private double X;
19     private double Y;
20     public LinkTrans(Transition t,String s,double X,double Y){
21         trans=t;
22         linktype=s;
23         this.X=X;
24         this.Y=Y;
25     }
26 
27     public Transition getTrans(){
28         return trans;
29     }
30     public String getType(){
31         return linktype;
32     }
33     public void setX(double X){
34         this.X=X;
35     }
36     public double getX(){
37         return X;
38     }
39     public void setY(double Y){
40         this.Y=Y;
41     }
42     public double getY(){
43         return Y;
44     }
45     public Object clone(){
46         String lt=linktype;
47         double x=X;
48         double y=Y;
49         return new LinkTrans(trans,lt,x,y);
50     }
51 }
```

Structure :


Transition
- serialVersionUID : long = 030L{frozen}
- arrow :Path
- line :QuadCurve
- trans :Group
- startpoint :Canvas
- endpoint :Canvas
- turnpoint :Canvas
- text :TextField
- enterstate :State
- endstate :State
- isturnsetted :boolean
- bufferdrag :Command
+ Transition()
- writeObject(ObjectOutputStream) :void
- readObject(ObjectInputStream) :void
+ getTextfield() :TextField
+ select() :void
+ unselected() :void
+ darw() :Group
+ getEnterstate() :State
+ setEnterstate(State ,String) :void
+ removeEnterstate() :void
+ getEndState() :State
+ setEndstate(State ,String) :void
+ removeEndstate() :void
+ setstartpoint(double ,double) :void
+ setendpoint(double ,double) :void
- resetarrow() :void
+ resetturnpoint() :void
+ isturnsetted() :boolean
+ setturnsetted(boolean) :void
- eval(QuadCurve ,float) :Point2D
- evalDt(QuadCurve ,float) :Point2D
- iniarrow() :void
- inipoint(Canvas) :void
- changeablepoint(Canvas) :void
+ setpointevent(SingleSelected ,Invoker) :void
+ moveText() :void
+ getTrunlayoutpoint() :Point
+ setTrunlayoutpoint(double ,double) :void
State
- serialVersionUID :long = 020L{frozen}
- ID :Long = System.currentTimeMillis(){frozen}
- state :Group
- state_pane :Pane
- cn :Canvas
- text :TextField
- point :Canvas
- linktrans :ArrayList<LinkTrans>
+ State()
- writeObject(ObjectOutputStream) :void
- readObject(ObjectInputStream) :void
+ linkTrans(Transition,String) :void
+ linkTrans(LinkTrans) :void
+ linkTrans(ArrayList<LinkTrans>) :void
+ removeTrans(Transition ,String) :void
+ changeLinkpoint(Transition ,String) :void
+ draw() :Group
+ getCenterX() :double
+ getCenterY() :double
+ setX(double) :void
+ setY(double) :void
+ getpointX() :double
+ getpointY() :double
+ setEvent(Transbtn) :void
+ select() :void
+ unselected() :void
+ pointmove(MouseEvent) :void
+ pointmove(DragEvent) :void
+ getlinktrans() :ArrayList<LinkTrans>
+ getlinktrans(Transition,String) :LinkTrans
+ getlinked() :ArrayList<Transition>
+ getTextfield() :TextField
+ clone() :Object
<<interface>>
Phototype
+ clone()
LinkTrans
- serialVersionUID :long = 3L{frozen}
- trans :Transition
- linktype :String
- X :double
- Y :double
+ LinkTrans(Transition ,String ,double ,double)
+ getTrans() :Transition
+ getType() :String
+ setX(double) :void
+ getX() :double
+ setY(double Y) :void
+ getY() :double
+ clone() :Object

- Command

  Problem : It’s high coupling about the operations of “Transition” and “State”, like new/delete a state, and it will cause the execution not flexible anymore.

  Solution : Create a abstract class “Command” to provide a uniform interface for all concrete command, and the “Command” will invokes the operation on receiving active from controller.
  The concrete command like “DeleteStateCommand” will extends the operation of the original object class.
  “Invoker” will store all the history of command, and provide “undo()” and “redo()”.

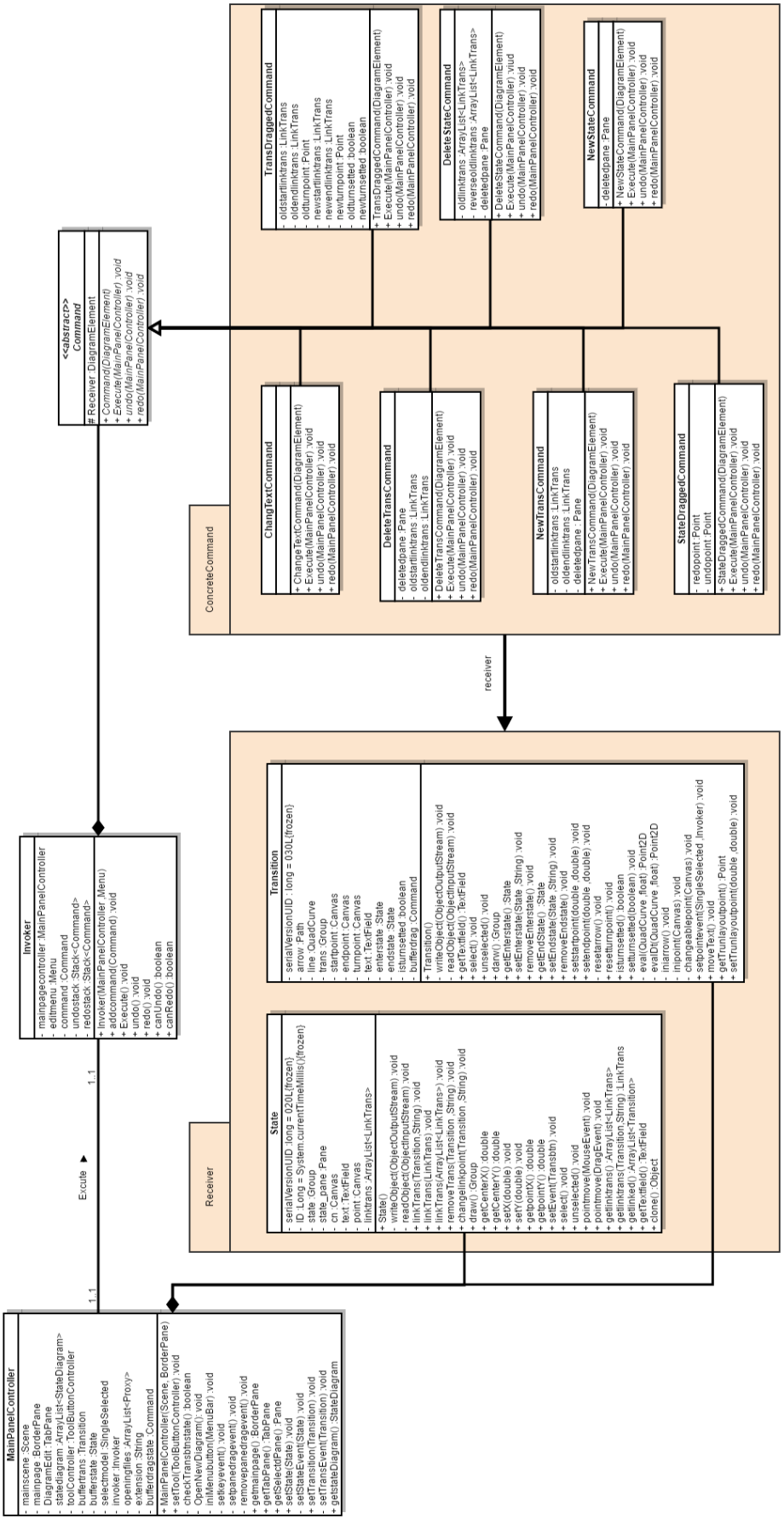
Code :

```java
public abstract class Command {
    protected DiagramElement Receiver;
    public Command(DiagramElement e){
        Receiver=e;
    }
    public abstract void Execute(MainPanelController controller);
    public abstract void undo(MainPanelController controller);
    public abstract void redo(MainPanelController controller);
}
```

```java
public class DeleteStateCommand extends Command {

    private ArrayList<LinkTrans> oldlinktrans=new ArrayList<>();
    private ArrayList<LinkTrans> reverseoldlinktrans=new ArrayList<>();
    private Pane deletedpane;
    public DeleteStateCommand(DiagramElement e){
        super(e);
    }
    @Override
    public void Execute(MainPanelController controller) {
        State s=(State)Receiver;
        deletedpane=controller.getSelectdPane();
        s.getlinktrans().forEach(l->{
            oldlinktrans.add((LinkTrans)l.clone());
            if (l.getType().equals("start")){
                reverseoldlinktrans.add((LinkTrans)l.getTrans().getEndState().getlinktrans(l.getTrans(), "end").clone(
            }else{
                reverseoldlinktrans.add((LinkTrans)l.getTrans().getEnterstate().getlinktrans(l.getTrans(), "start").cl
            }
        });
        //System.out.println(oldlinktrans.size()+" "+reverseoldlinktrans.size());
        oldlinktrans.forEach(link->{
            link.getTrans().getEnterstate().removeTrans(link.getTrans(),"start");
            link.getTrans().getEndState().removeTrans(link.getTrans(), "end");
            controller.getstateDiagram().remove(link.getTrans());
            Group tc=link.getTrans().draw();
            Group g=(Group)controller.getSelectdPane().getChildren().get(0);
            g.getChildren().remove(tc);
        });
        controller.getstateDiagram().remove(Receiver);
        Group c=Receiver.draw();
        Group g=(Group)controller.getSelectdPane().getChildren().get(0);
        g.getChildren().remove(c);

        System.out.println("DeleteState after statediagram:"+controller.getstateDiagram().getAlldiagram().size());
    }
```

```
public class Invoker {
    private MainPanelController mainpagecontroller;
    private Menu editmenu;
    private Command command;
    private Stack<Command> undostack=new Stack<>();
    private Stack<Command> redostack=new Stack<>();
    public Invoker(MainPanelController c,Menu edit){
        mainpagecontroller=c;
        editmenu=edit;
    }
    public void addcommand(Command c){
        command=c;
    }
    public void Execute(){
        command.Execute(mainpagecontroller);
        undostack.push(command);
        editmenu.getItems().get(0).setDisable(false);
        command=null;
        redostack.clear();
        editmenu.getItems().get(1).setDisable(true);
    };
    public void undo(){
        Command c;
        if (canUndo()){
            c=undostack.pop();
            if (!canUndo()){editmenu.getItems().get(0).setDisable(true);}
            c.undo(mainpagecontroller);
            redostack.push(c);
            editmenu.getItems().get(1).setDisable(false);
        }
    }
    public void redo(){
        Command c;
        if (canRedo()){
            c=redostack.pop();
            if (!canRedo()){editmenu.getItems().get(1).setDisable(true);}
            c.redo(mainpagecontroller);
```

Structure :


MainPanelController
Invoker
<<abstract>>
Command
ConcreteCommand
ChangeTextCommand
DeleteTransCommand
NewTransCommand
StateDraggedCommand
TransDraggedCommand
DeleteStateCommand
NewStateCommand
Receiver
State
Transition
Excute
receiver
1..1

- Composite

Problem : The methods of storing states, transitions and state diagrams are dependent. It’s too unstructured, and we can’t treat they as the same, like select either a state or a transition, but they are all element of the diagram.

Solution : Create an interface “DiagramElement“ to let the state or transition as a DiagramElement , so that “StateDiagram” can treat all the states and transition as the same , and store the states , transitions, state diagram by using the Arraylist<DiagramElement>

Code :

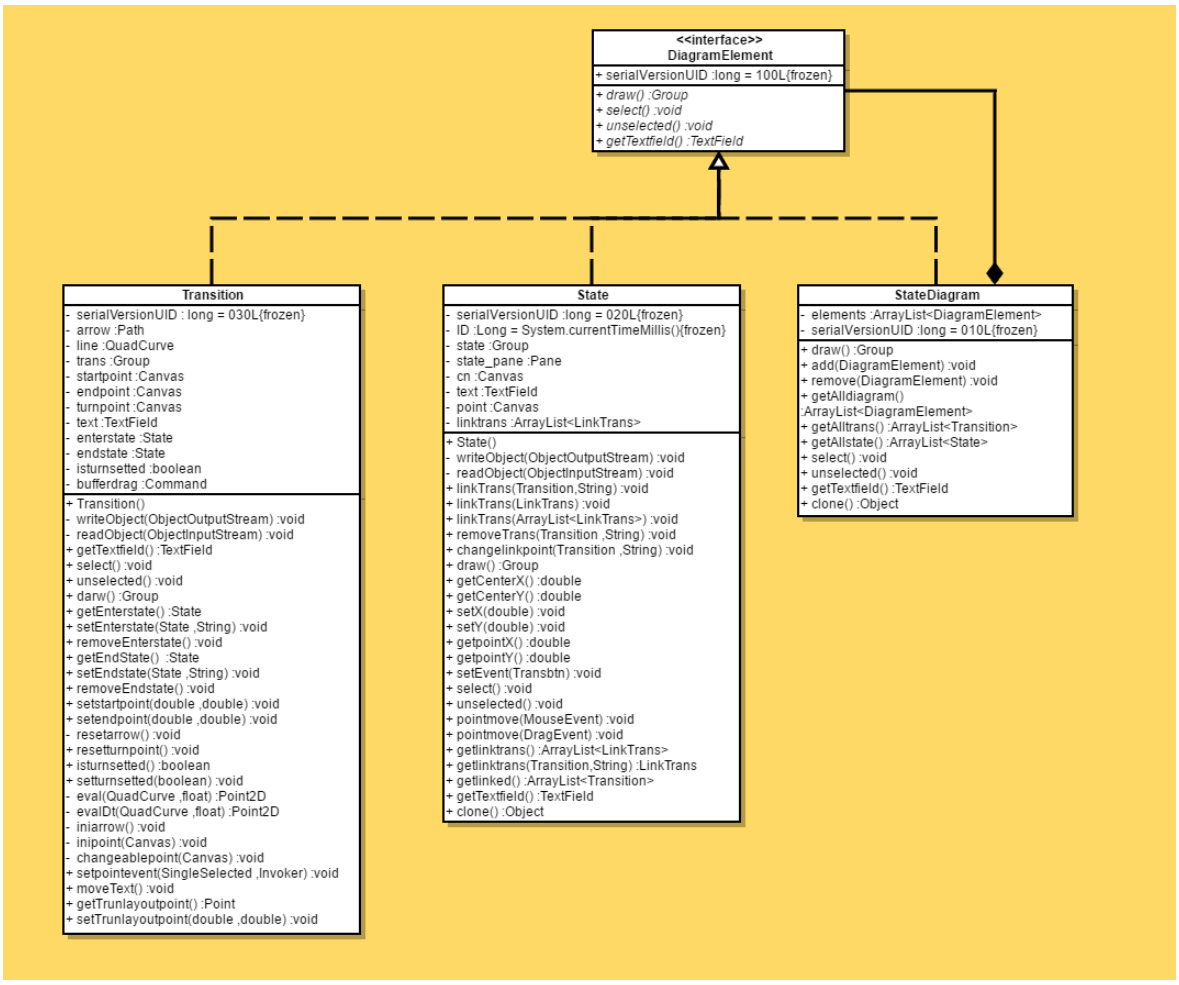
```
/
public interface DiagramElement extends Serializable{
    static final long serialVersionUID = 100L;
    public Group draw();
    public void select();
    public void unselected();
    public TextField getTextfield();
}
```

```
public class State implements DiagramElement,Phototype{
    private static final long serialVersionUID = 020L;
    private final Long ID=System.currentTimeMillis();
    private transient Group state=new Group();
    private transient Pane state_pane=new Pane();
    private transient Canvas cn=new Canvas(100,100);
    private transient TextField text=new TextField();
```

```
 */
public class Transition implements DiagramElement {
    private static final long serialVersionUID = 030L;
    private transient Path arrow;
    private transient QuadCurve line=new QuadCurve();
    private transient Group trans=new Group();
    private transient Canvas startpoint=new Canvas(17,17);
    private transient Canvas endpoint=new Canvas(17,17);
    private transient Canvas turnpoint=new Canvas(17,17);
    private transient TextField text=new TextField();
    private State enterstate=null;
    private State endstate=null;
    private boolean isturnsetted=false;
    private Command bufferdrag;
    public Transition(){

        line.setStartX(50);
        line.setStartY(50);
```

```
public class StateDiagram implements DiagramElement,Phototype {
    private ArrayList<DiagramElement> elements=new ArrayList<>();
    private static final long serialVersionUID = 010L;
    public Group draw(){
        return null;
    }
    public void add(DiagramElement e){
        elements.add(e);

    }
    public void remove(DiagramElement e){
        elements.remove(e);
    }
    public ArrayList<DiagramElement> getAlldiagram(){
```

Structure :


<<interface>>
DiagramElement
+ serialVersionUID :long = 100L{frozen}
+ draw() :Group
+ select() :void
+ unselected() :void
+ getTextfield() :TextField
Transition
- serialVersionUID : long = 030L{frozen}
- arrow :Path
- line :QuadCurve
- trans :Group
- startpoint :Canvas
- endpoint :Canvas
- turnpoint :Canvas
- text :TextField
- enterstate :State
- endstate :State
- isturnsetted :boolean
- bufferdrag :Command
+ Transition()
- writeObject(ObjectOutputStream) :void
- readObject(ObjectInputStream) :void
+ getTextfield() :TextField
+ select() :void
+ unselected() :void
+ darw() :Group
+ getEnterstate() :State
+ setEnterstate(State ,String) :void
+ removeEnterstate() :void
+ getEndState() :State
+ setEndstate(State ,String) :void
+ removeEndstate() :void
+ setstartpoint(double ,double) :void
+ setendpoint(double ,double) :void
- resetarrow() :void
+ resetturnpoint() :void
+ isturnsetted() :boolean
+ setturnsetted(boolean) :void
- eval(QuadCurve ,float) :Point2D
- evalDt(QuadCurve ,float) :Point2D
- iniarrow() :void
- inipoint(Canvas) :void
- changeablepoint(Canvas) :void
+ setpointevent(SingleSelected ,Invoker) :void
+ moveText() :void
+ getTrunlayoutpoint() :Point
+ setTrunlayoutpoint(double ,double) :void
State
- serialVersionUID :long = 020L{frozen}
- ID :Long = System.currentTimeMillis(){frozen}
- state :Group
- state_pane :Pane
- cn :Canvas
- text :TextField
- point :Canvas
- linktrans :ArrayList<LinkTrans>
+ State()
- writeObject(ObjectOutputStream) :void
- readObject(ObjectInputStream) :void
+ linkTrans(Transition,String) :void
+ linkTrans(LinkTrans) :void
+ linkTrans(ArrayList<LinkTrans>) :void
+ removeTrans(Transition ,String) :void
+ changelinkpoint(Transition ,String) :void
+ draw() :Group
+ getCenterX() :double
+ getCenterY() :double
+ setX(double) :void
+ setY(double) :void
+ getpointX() :double
+ getpointY() :double
+ setEvent(Transbtn) :void
+ select() :void
+ unselected() :void
+ pointmove(MouseEvent) :void
+ pointmove(DragEvent) :void
+ getlinktrans() :ArrayList<LinkTrans>
+ getlinktrans(Transition,String) :LinkTrans
+ getlinked() :ArrayList<Transition>
+ getTextfield() :TextField
+ clone() :Object
StateDiagram
- elements :ArrayList<DiagramElement>
- serialVersionUID :long = 010L{frozen}
+ draw() :Group
+ add(DiagramElement) :void
+ remove(DiagramElement) :void
+ getAlldiagram()
:ArrayList<DiagramElement>
+ getAlltrans() :ArrayList<Transition>
+ getAllstate() :ArrayList<State>
+ select() :void
+ unselected() :void
+ getTextfield() :TextField
+ clone() :Object

- State

  Problem : We want to the behavior of all the elements changing along with state changing and allow an object to alter its behavior.

  Solution : Define a abstract class as a part of the button and button can change by itself.

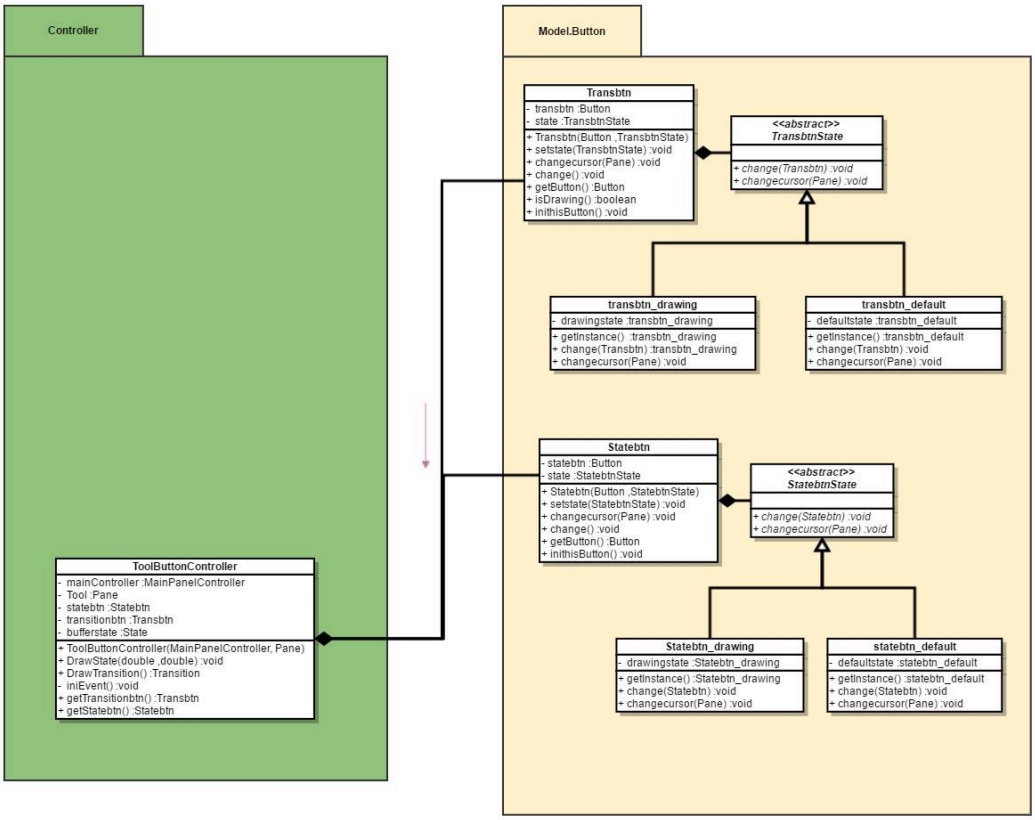
Code :

```java
public class Transbtn{
    private Button transbtn;
    private TransbtnState state;
    public Transbtn(Button btn,TransbtnState defaultstate){
        transbtn=btn;
        state=defaultstate;
    }
    public void setstate(TransbtnState state){
        this.state=state;
    }
    public void changecursor(Pane mainpage){
        state.changecursor(mainpage);
    }
    public void change(){
        state.change(this);
    }

    public Button getButton(){
        return transbtn;
    }
    public boolean isDrawing(){
        return (state == transbtn_drawing.getInstance());
    }
    public void inithisButton(){
        state=transbtn_default.getInstance();
    }
}

```

```java
public abstract class TransbtnState {
    public abstract void change(Transbtn b);
    public abstract void changecursor(Pane mainpage);
}

```

```
 *
 * @author 登陆者
 */
public class transbtn_default extends TransbtnState {

    private static transbtn_default defaultstate=new transbtn_default();
    public static transbtn_default getInstance(){
        return defaultstate;
    }
    @Override
    public void change(Transbtn b) {
        b.setstate(transbtn_drawing.getInstance());
    }

    @Override
    public void changecursor(Pane mainpage) {
        mainpage.setCursor(Cursor.DEFAULT);
    }

}
```

Structure :


Controller
Model.Button
Transbtn
- transbtn :Button
- state :TransbtnState
+ Transbtn(Button ,TransbtnState)
+ setstate(TransbtnState) :void
+ changecursor(Pane) :void
+ change() :void
+ getButton() :Button
+ isDrawing() :boolean
+ initthisButton() :void
<<abstract>>
TransbtnState
+ change(Transbtn) :void
+ changecursor(Pane) :void
transbtn_drawing
- drawingstate :transbtn_drawing
+ getInstance() :transbtn_drawing
+ change(Transbtn) :transbtn_drawing
+ changecursor(Pane) :void
transbtn_default
- defaultstate :transbtn_default
+ getInstance() :transbtn_default
+ change(Transbtn) :void
+ changecursor(Pane) :void
Statebtn
- statebtn :Button
- state :StatebtnState
+ Statebtn(Button ,StatebtnState)
+ setstate(StatebtnState) :void
+ changecursor(Pane) :void
+ change() :void
+ getButton() :Button
+ initthisButton() :void
<<abstract>>
StatebtnState
+ change(Statebtn) :void
+ changecursor(Pane) :void
Statebtn_drawing
- drawingstate :Statebtn_drawing
+ getInstance() :Statebtn_drawing
+ change(Statebtn) :void
+ changecursor(Pane) :void
statebtn_default
- defaultstate :statebtn_default
+ getInstance() :statebtn_default
+ change(Statebtn) :void
+ changecursor(Pane) :void
ToolButtonController
- mainController :MainPanelController
- Tool :Pane
- statebtn :Statebtn
- transitionbtn :Transbtn
- bufferstate :State
+ ToolButtonController(MainPanelController, Pane)
+ DrawState(double ,double) :void
+ DrawTransition() :Transition
- initEvent() :void
+ getTransitionbtn() :Transbtn
+ getStatebtn() :Statebtn

- Proxy

  Problem : We want to load a persistent object into memory when it’s first referenced, and check that the real object is locked before it’s accessed to ensure that no other object can change it.

  Solution : Define a common interface called “Proxy” to let the proxy “FileProxy” and substituted “LocalFile” implement, and use “FileProxy” to controls access to the “LocalFile”.

  In additional, the “FileProxy” can also do the additional function like open file after you open the editor.

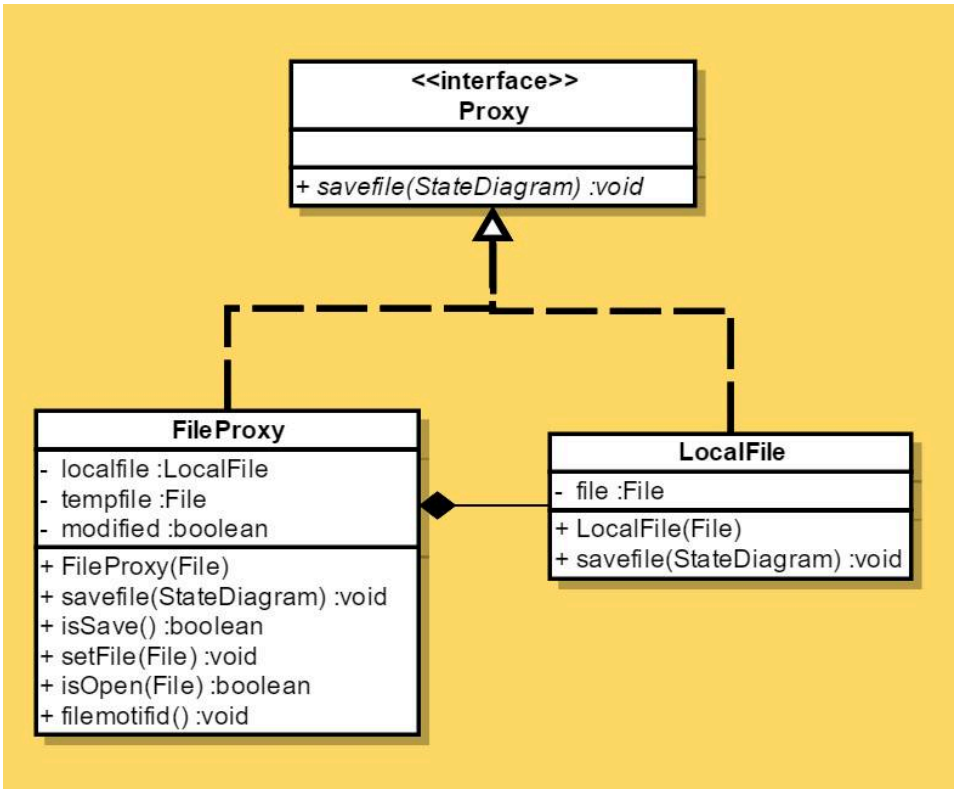
Code :

```
 * @author 甯鍞鍞
 */
public interface Proxy {
    public void openfile(File f);
    public void savefile(StateDiagram sd);
    public void SaveAsNew(StateDiagram sd);
    public boolean isOpen(File f);
    public boolean isSave();
    public void closefile();

```

```
 */
public class FileProxy implements Proxy{
    private MainPanelController maincontroller;
    private ArrayList<LocalFile> localfiles=new ArrayList<>();
    private boolean modified=false;
    private String extension="";
    public FileProxy(MainPanelController controller){
        maincontroller=controller;
    }
    public void openfile(File file){
        if (file!=null){
            String filename=file.getName();
            try{
                FileInputStream fileInputStream = new FileInputStream(file
                ObjectInputStream objInputStream = new ObjectInputStream(f

                maincontroller.getTabPane().getSelectionModel().getSelecte
                StateDiagram opensd=new StateDiagram();
                DiagramElement d;
                while(fileInputStream.available() > 0) {

                    d=(DiagramElement)objInputStream.readObject();
```

```
 *
 * @author 鲁****
 */
public class LocalFile implements Proxy {
    private File file=null;

    public void setfile(File f){
        file=f;
    }
    @Override
    public void savefile(StateDiagram sd) {
        if (file!=null){
            BufferedWriter writer=null;

            try{
                FileOutputStream fs = new FileOutputStream(fi
                ObjectOutputStream os =  new ObjectOutputStre
                //System.out.println(DiagramEdit.getSelectionI
                sd.getAlldiagram().forEach(de->{
                    try{
                        if (de instanceof Transition){
                            if ((((Transition)de).getEnterstat
```

Structure :



- Singleton

Problem : We want to every state of transitions and states created by single, independent.

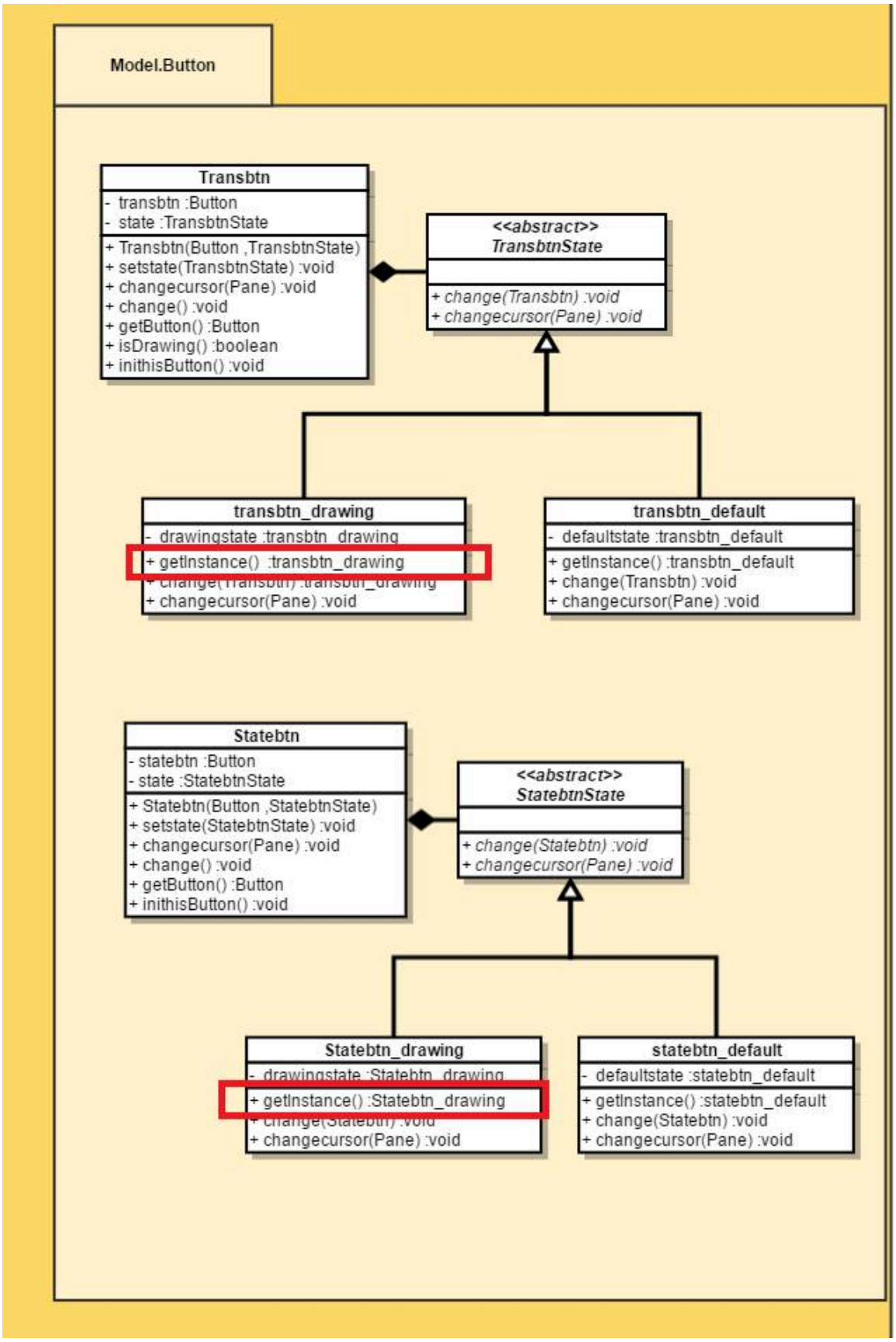
Solution : Because state change often , so we make state create itself when program starting. If button want change state , just get instance already exist.

Code :

```
15 public class transbtn_drawing extends TransbtnState {
16
17     private static transbtn_drawing drawingstate=new transbtn_drawing();
18     public static transbtn_drawing getInstance(){
19         return drawingstate;
20     }
```

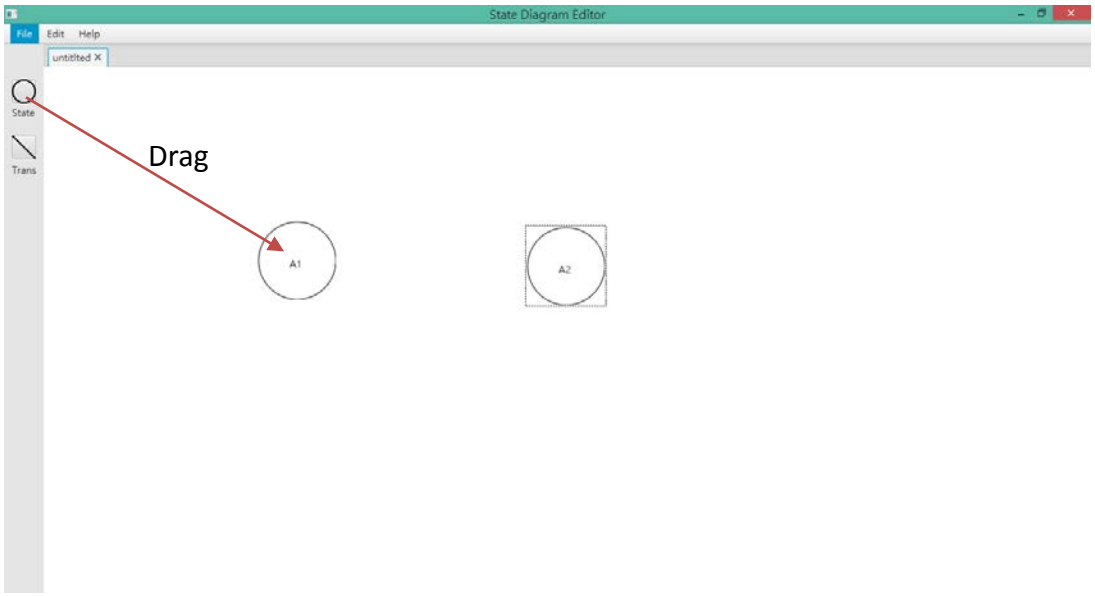
```
16 public class transbtn_default extends TransbtnState {
17
18     private static transbtn_default defaultstate=new transbtn_default();
19     public static transbtn_default getInstance(){
20         return defaultstate;
21     }
```

Structure :

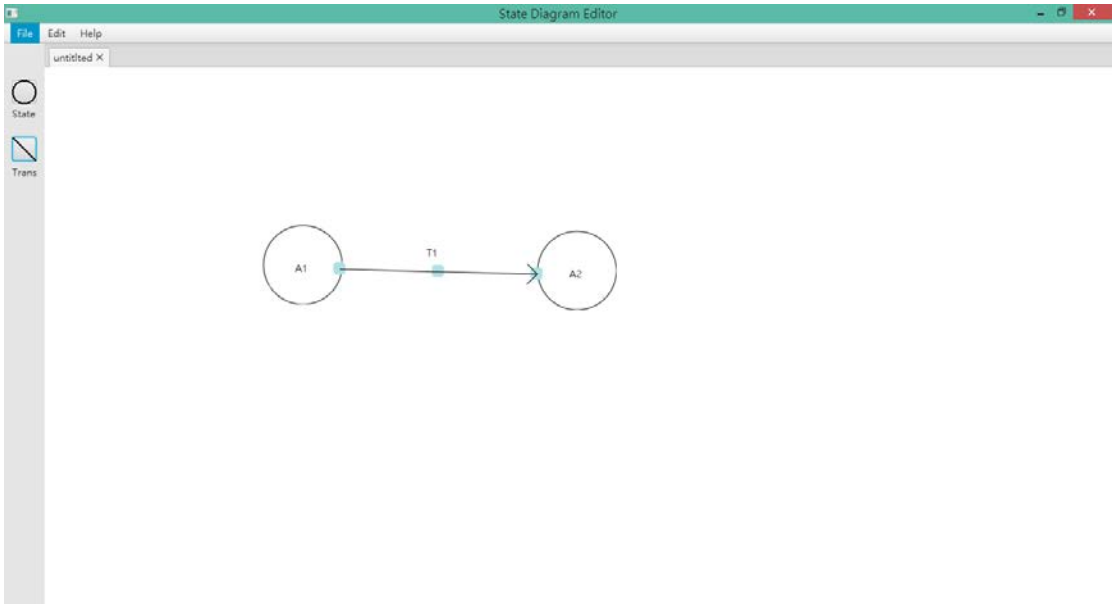

Model.Button
Transbtn
- transbtn :Button
- state :TransbtnState
+ Transbtn(Button ,TransbtnState)
+ setstate(TransbtnState) :void
+ changecursor(Pane) :void
+ change() :void
+ getButton() :Button
+ isDrawing() :boolean
+ inithisButton() :void
<<abstract>>
TransbtnState
+ change(Transbtn) :void
+ changecursor(Pane) :void
transbtn_drawing
- drawingstate :transbtn_drawing
+ getInstance() :transbtn_drawing
+ changecursor(Pane) :void
transbtn_default
- defaultstate :transbtn_default
+ getInstance() :transbtn_default
+ change(Transbtn) :void
+ changecursor(Pane) :void
Statebtn
- statebtn :Button
- state :StatebtnState
+ Statebtn(Button ,StatebtnState)
+ setstate(StatebtnState) :void
+ changecursor(Pane) :void
+ change() :void
+ getButton() :Button
+ inithisButton() :void
<<abstract>>
StatebtnState
+ change(Statebtn) :void
+ changecursor(Pane) :void
Statebtn_drawing
- drawingstate :Statebtn_drawing
+ getInstance() :Statebtn_drawing
+ changecursor(Pane) :void
statebtn_default
- defaultstate :statebtn_default
+ getInstance() :statebtn_default
+ change(Statebtn) :void
+ changecursor(Pane) :void

Operation

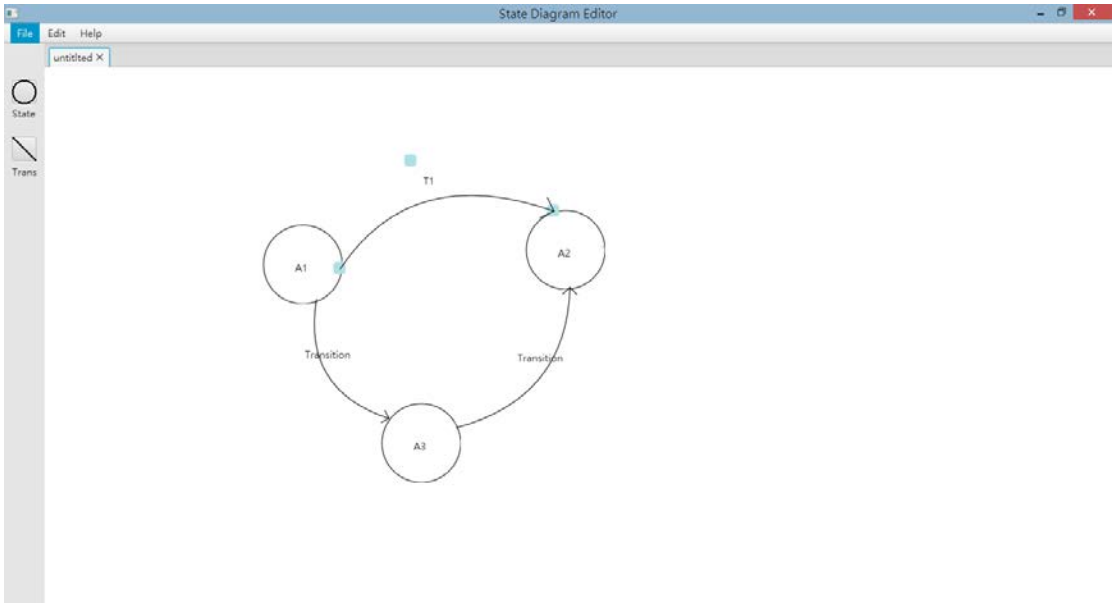
- ✓ **Add State and rename**



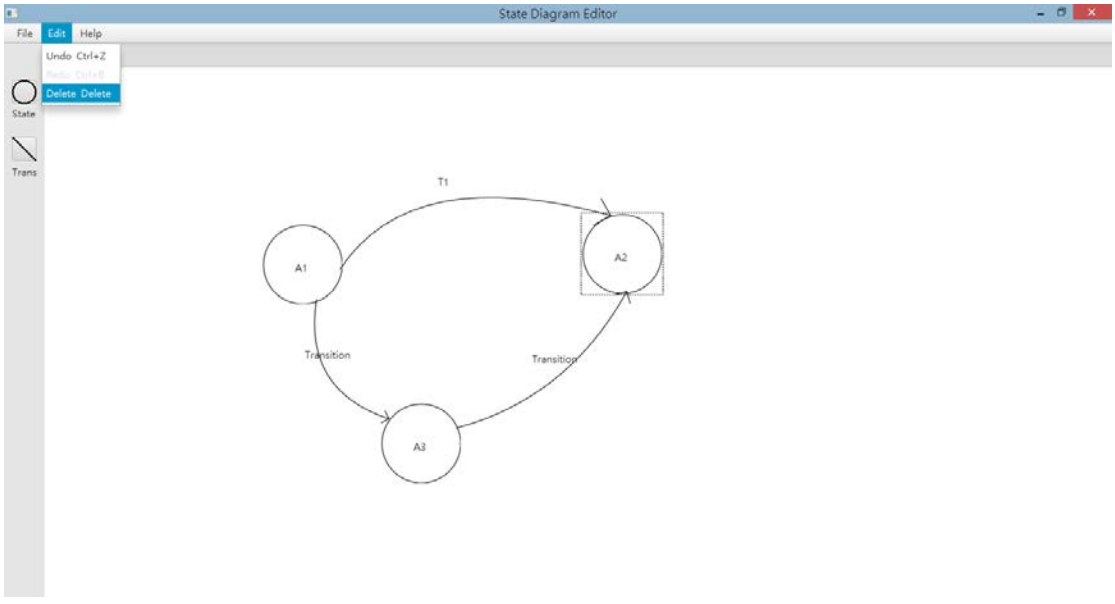
- ✓ **Click Trans button , Mouse will change to drawing state , and you can draw Transition and rename**

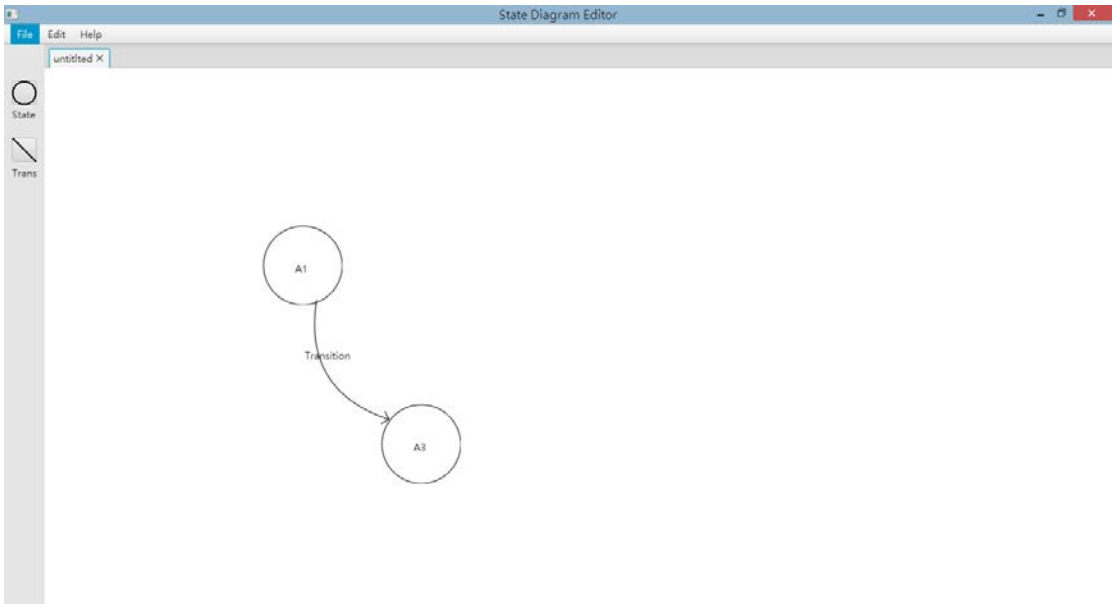


✓ **Also can make transition's point that link the state moving , or make transition bending.**

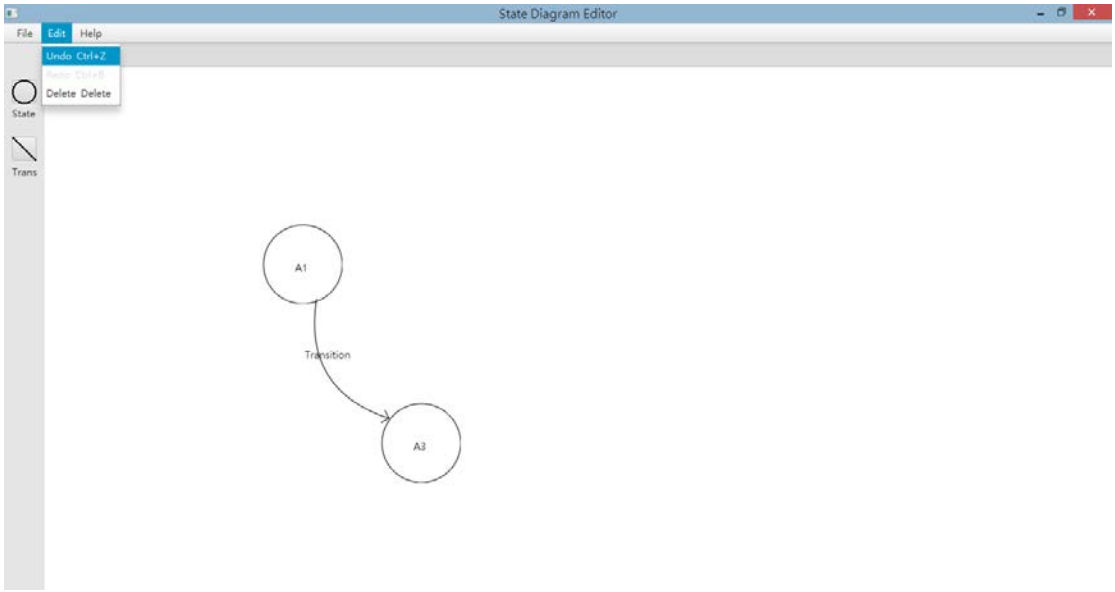


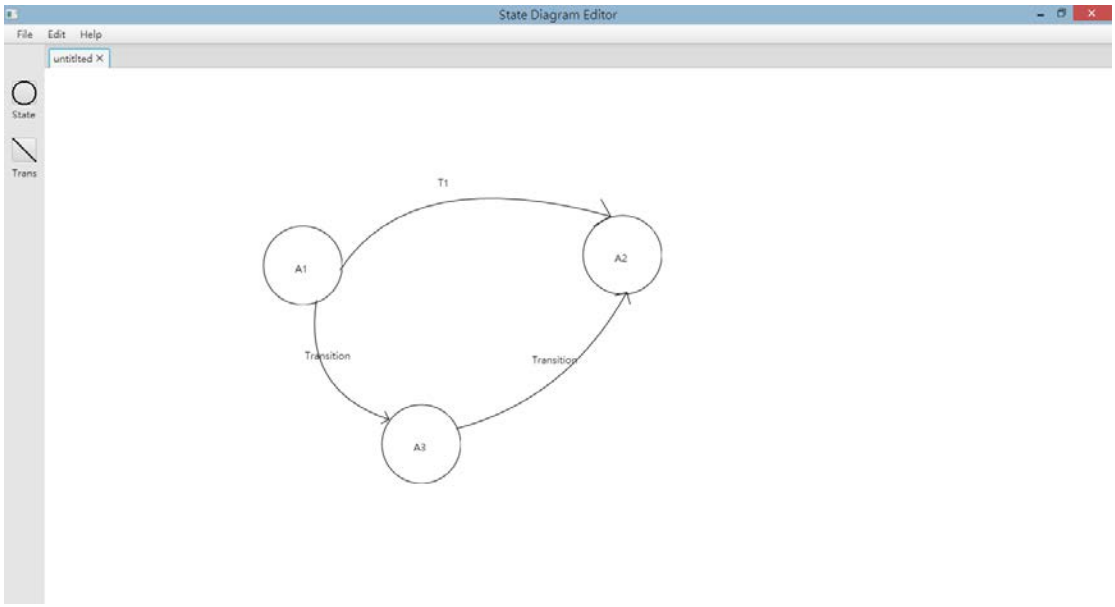
✓ **Delete (or press delete key)**



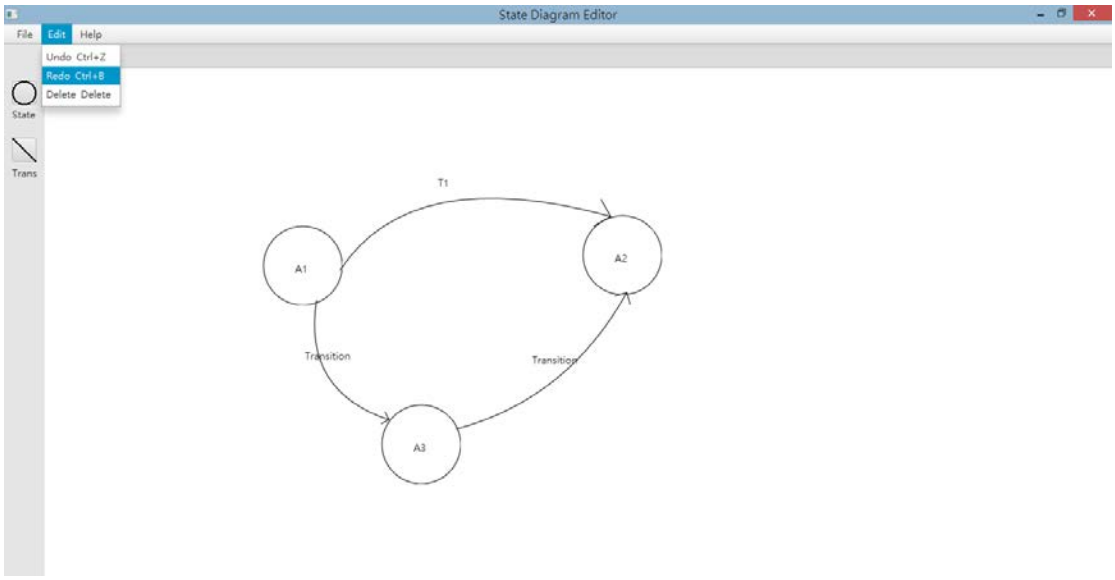

State Diagram Editor
File
Edit
Help
untitled
State
Trans
A1
Transition
A3

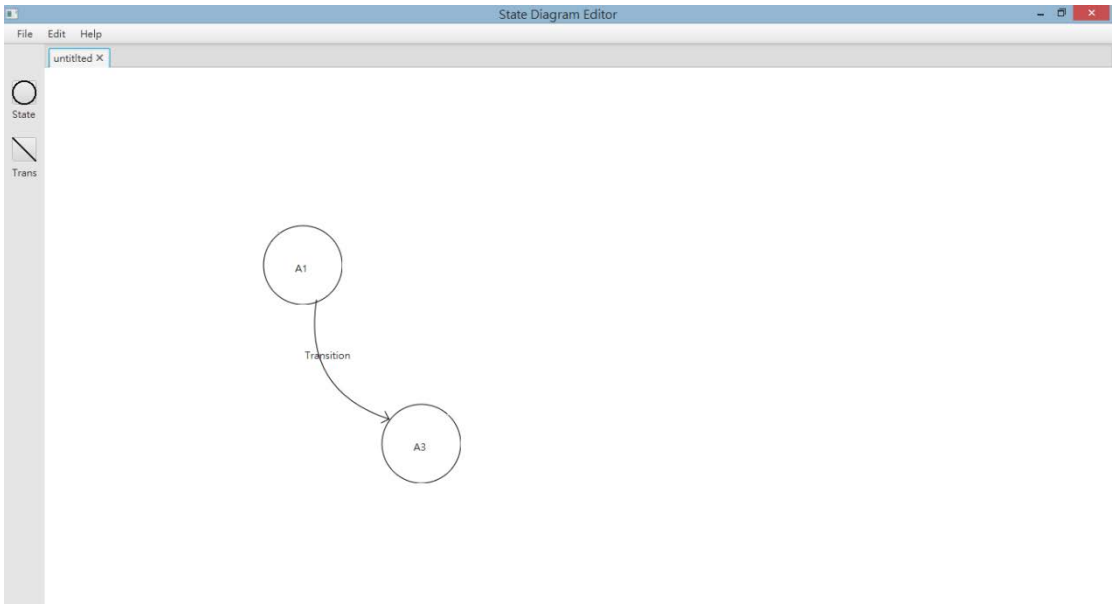
✓ Undo (Ctrl+Z)


State Diagram Editor
File
Edit
Help
Undo Ctrl+Z
Delete Delete
State
Trans
A1
Transition
A3

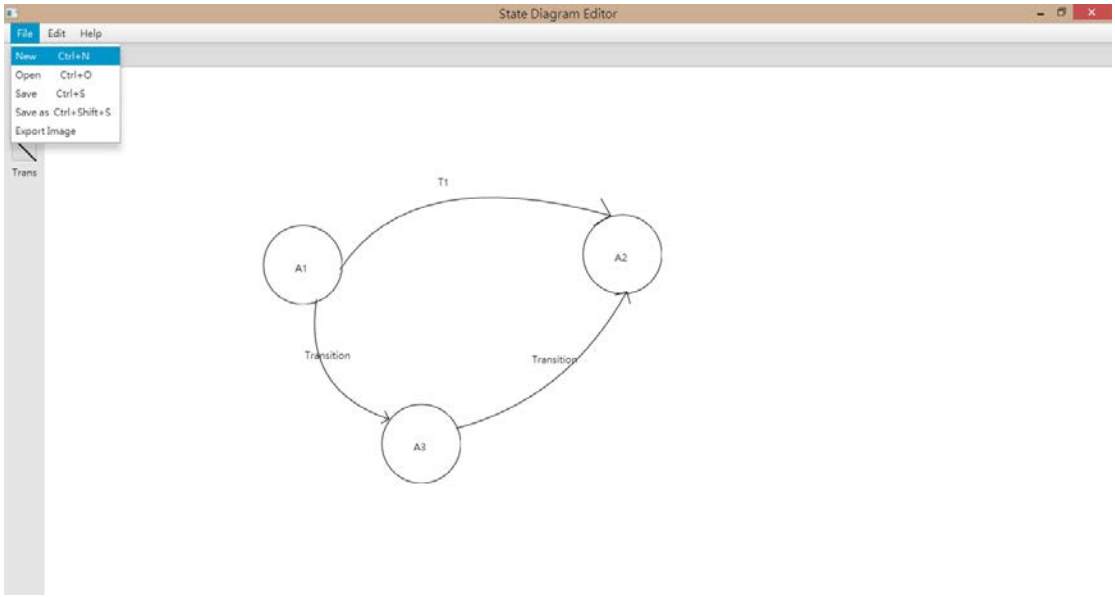

State Diagram Editor
File
Edit
Help
untitled
State
Trans
T1
A1
A2
Transition
Transition
A3

Redo (Ctrl + B)

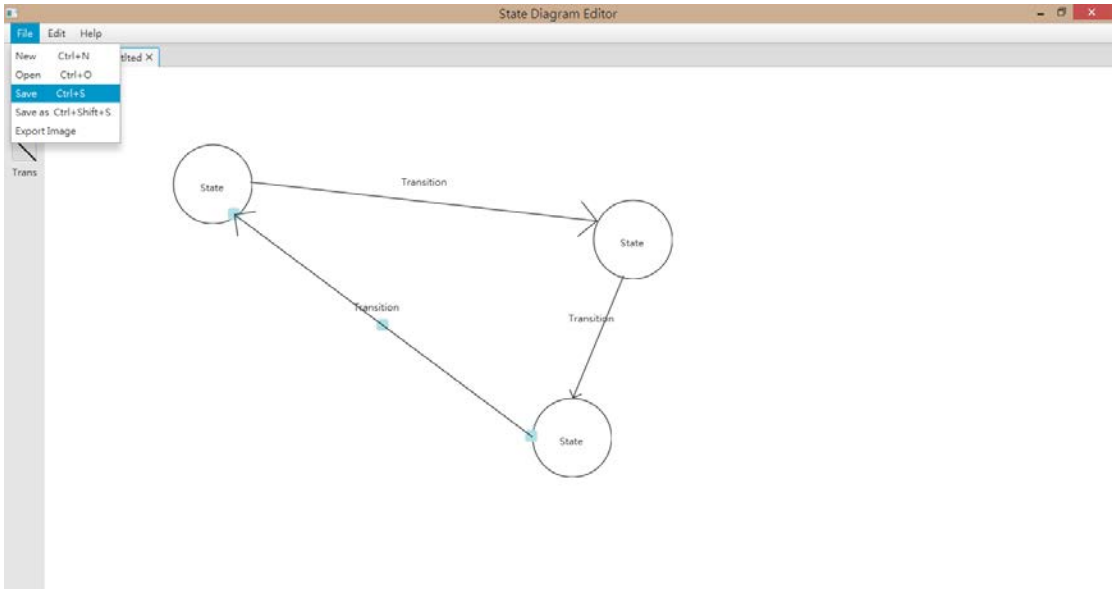

State Diagram Editor
File
Edit
Help
Undo Ctrl+Z
Redo Ctrl+B
Delete Delete
State
Trans
T1
A1
A2
Transition
Transition
A3

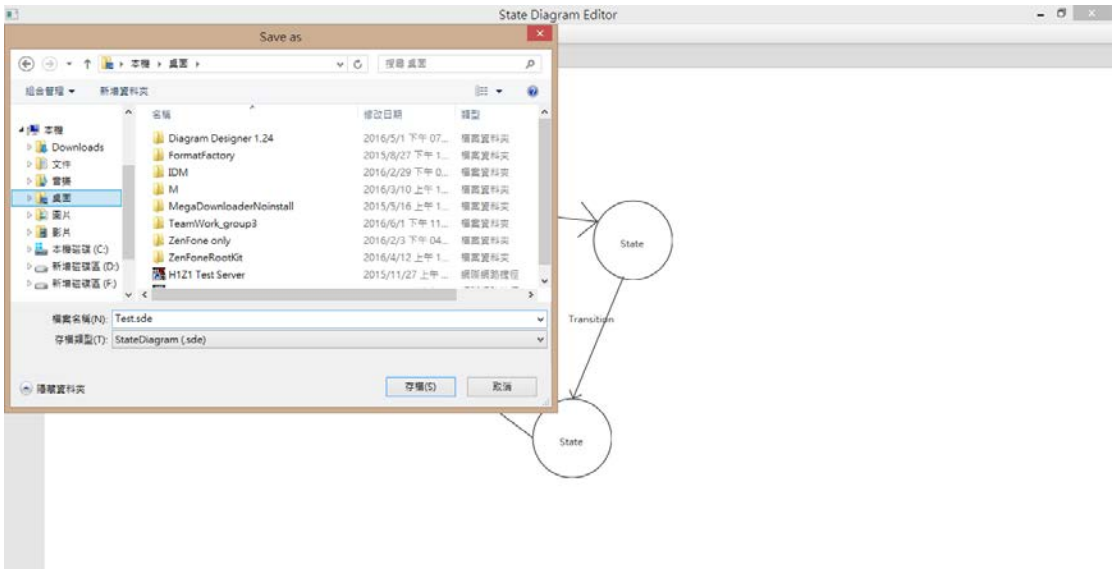


✓ Open new page (Ctrl + N)

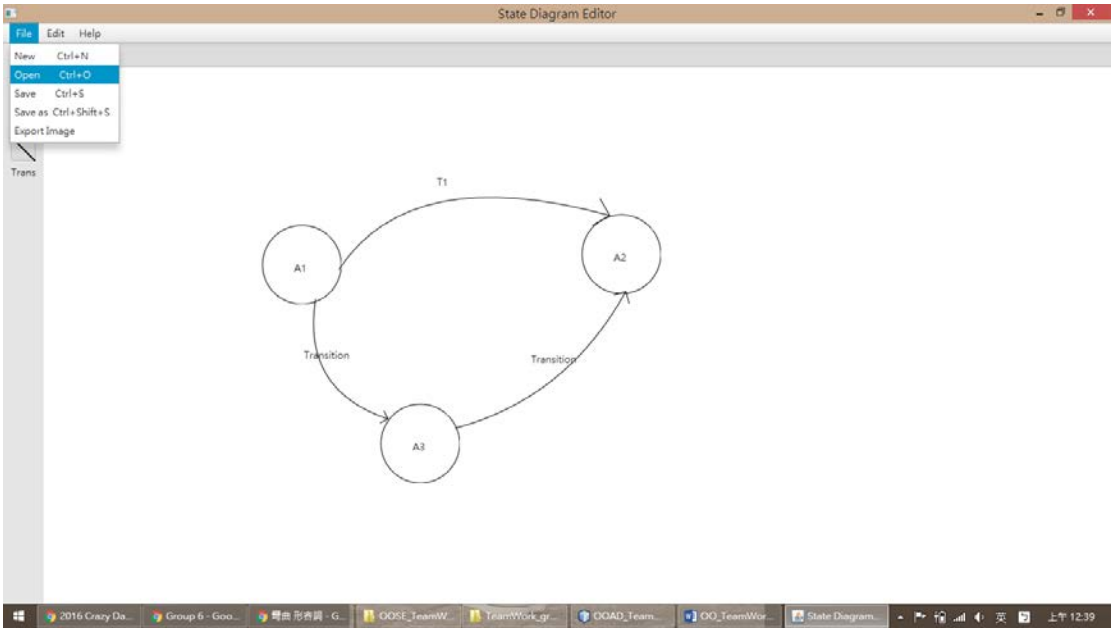


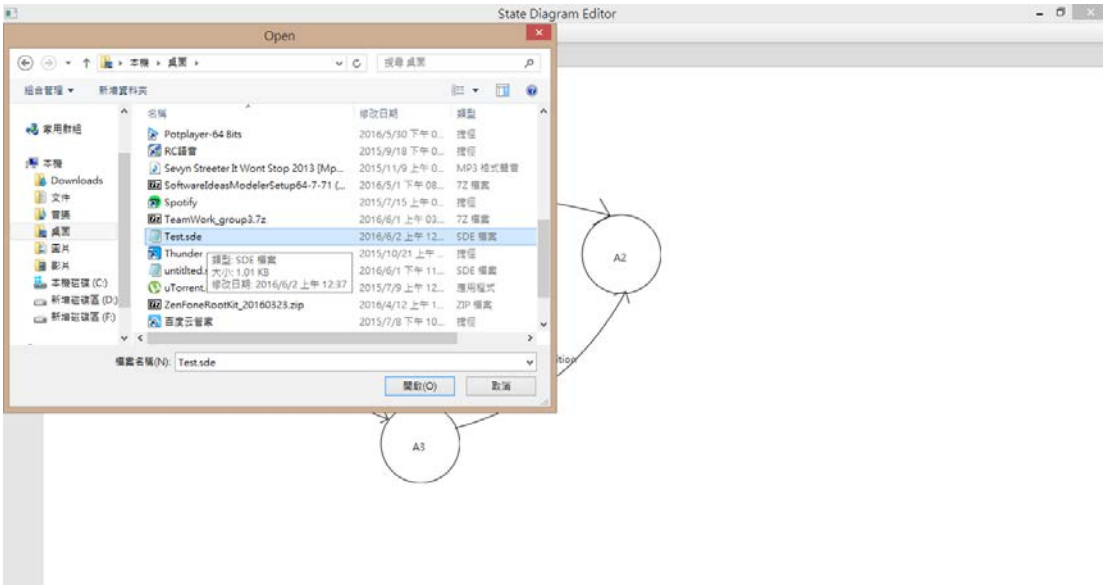
✓ **Save this file(Ctrl + S) or Save as new file(Ctrl + Shift + S)**

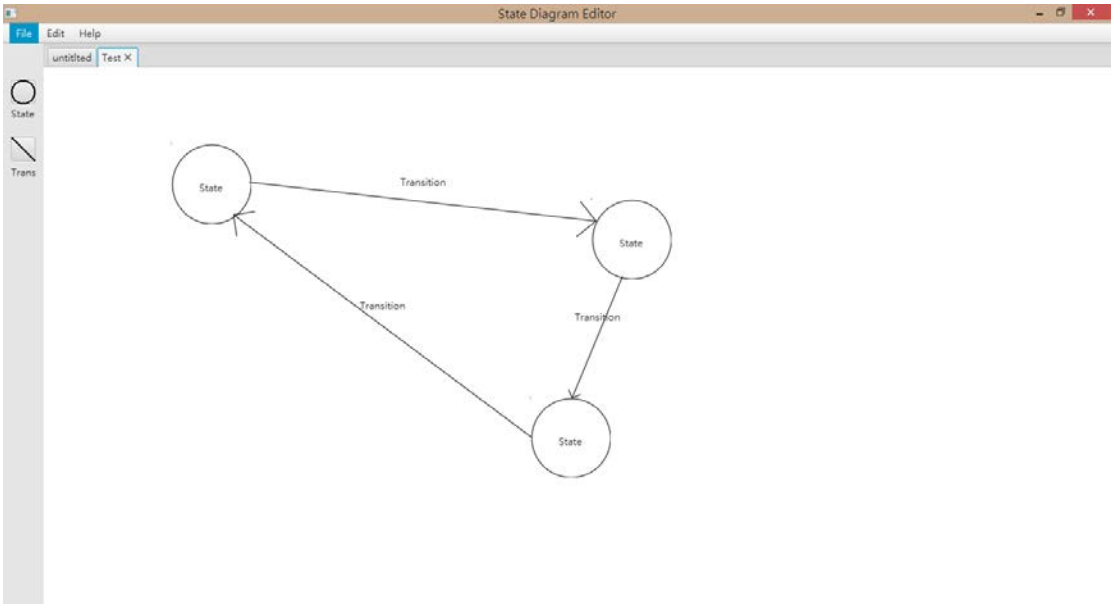




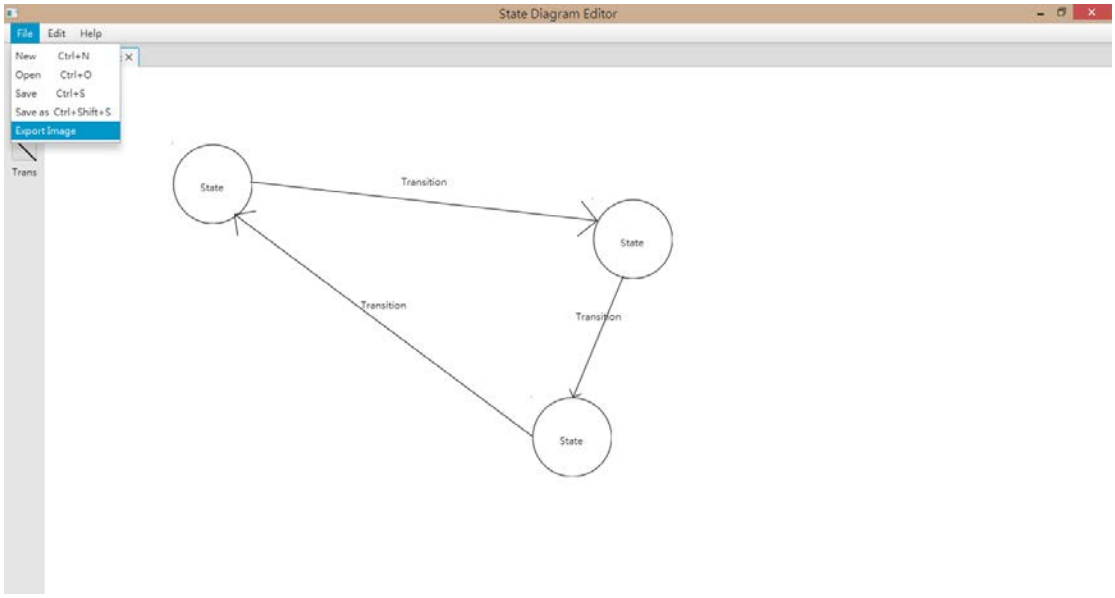
✓ **Open saved file**

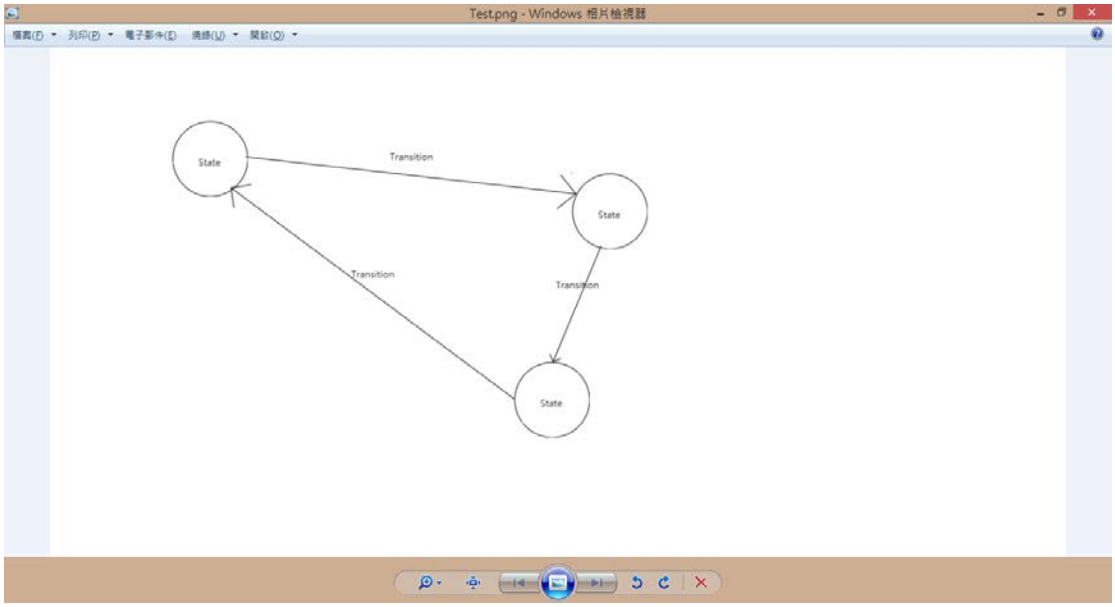

State Diagram Editor
File
Edit
Help
New Ctrl+N
Open Ctrl+O
Save Ctrl+S
Save as Ctrl+Shift+S
Export Image
Trans
T1
A1
A2
A3
Transition
Transition


State Diagram Editor
Open
Potplayer-64 Bits
RC語音
Sevyn Streeter It Wont Stop 2013 [Mp...
SoftwareIdeasModelerSetup64-7-71 (...
Spotify
TeamWork_group3.7z
Test.sde
Thunder
untitled.s
uTorrent
ZenFoneRootKit_20160323.zip
檔案名稱(N): Test.sde
開啟(O)
取消
A2
A3


State Diagram Editor
File
Edit
Help
untitled
Test
State
Trans
State
Transition
State
Transition
Transition
State

✓ Export a Image





Participation :

- Collaboration

(Coding & annotation, Document, PPT, Class Diagram, snapshots & operation, Description)

	B10223005	B10223031	B10223041	B10223047	B10223059	B10223062
Coding & annotation		S	R	S	S	S
Document	R		S	S		
PPT				R	S	
Class Diagram		R	S			
snapshots & operation	S		S			R
Description	S		S		R	S
R = Responsible S = Support						

- Participation summary of team members

B10223005	100%
B10223031	100%
B10223041	100%
B10223047	100%
B10223059	100%
B10223062	100%