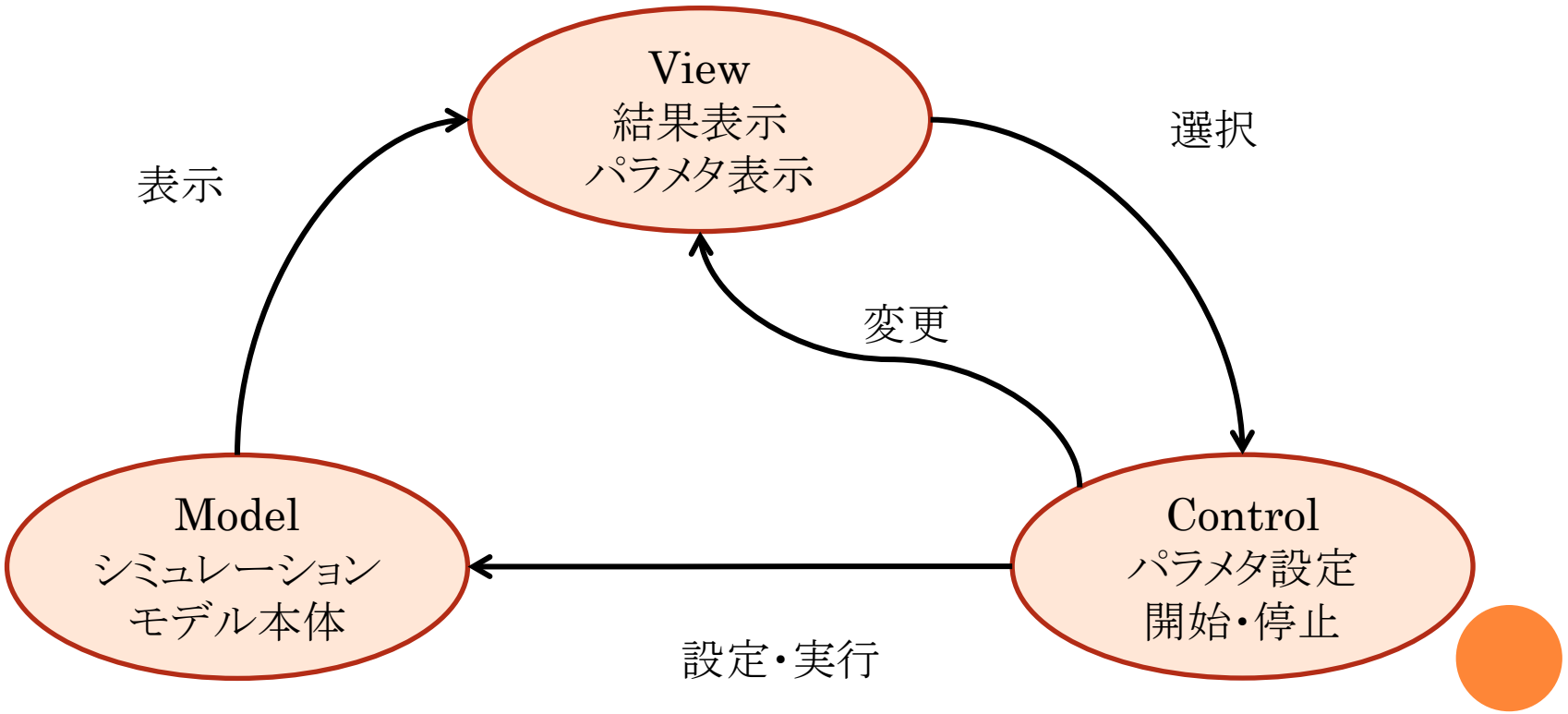
シミュレーションの例
結合写像最適速度模型

オブジェクト指向プログラミング特論

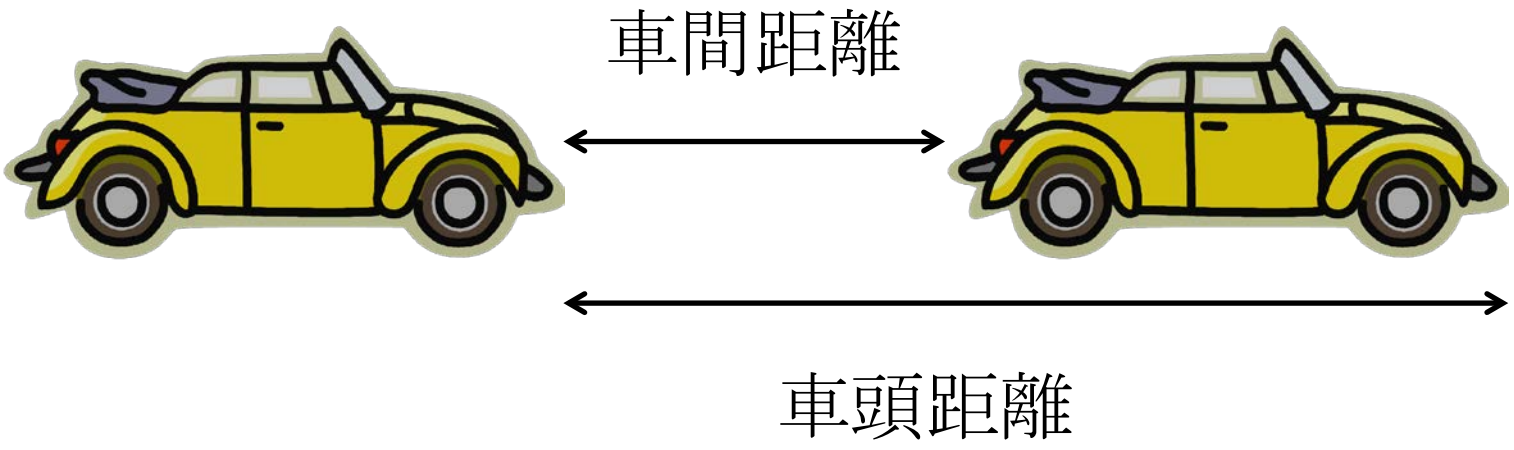
只木進一：総合情報基盤センター

シミュレーションの場合のMVC



結合写像型最適速度模型

- 交通流のモデル
- 車頭距離に応じた最適速度がある
- 最適速度に調整するように加減速する



モデルの定義

時刻 t での、ある車両 i とその先行車両 $i-1$ との車頭距離を $\Delta x(t)$ とする。このとき、車両 i の加速度 $a(t)$ は

$$a(t)=\alpha\left[V_{\text{optimal}}\left(\Delta x(t)\right)-v(t)\right] \tag{1.1}$$

であるとする。ここで $v(t)$ は、時刻 t における車両 i の速度である。つまり、各車両は、車頭距離によって定まる最適速度と現在の速度の差に比例した加速度で、最適速度となるように速度を調整するとする。

時刻が Δt 経過すると、速度と位置は以下のように変化する。

$$v(t+\Delta t)=v(t)+a(t)\Delta t \tag{1.2}$$

$$x(t+\Delta t)=x(t)+v(t)\Delta t \tag{1.3}$$

最適速度

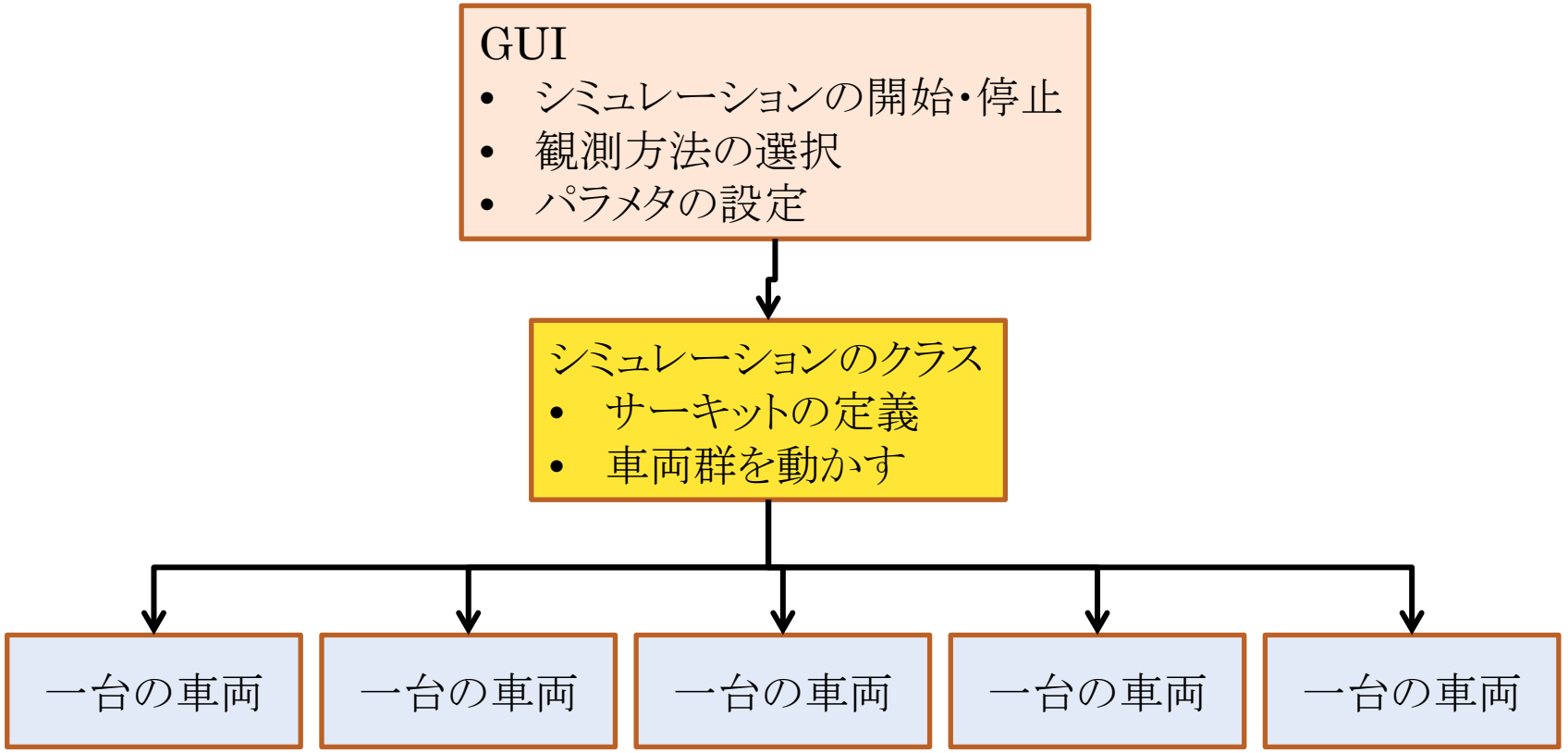
- 一般的にはシグモイド型
- 例：階段関数

$$V_{\text{optimal}}(\Delta x) = v_{\max}\theta(\Delta x - d)$$

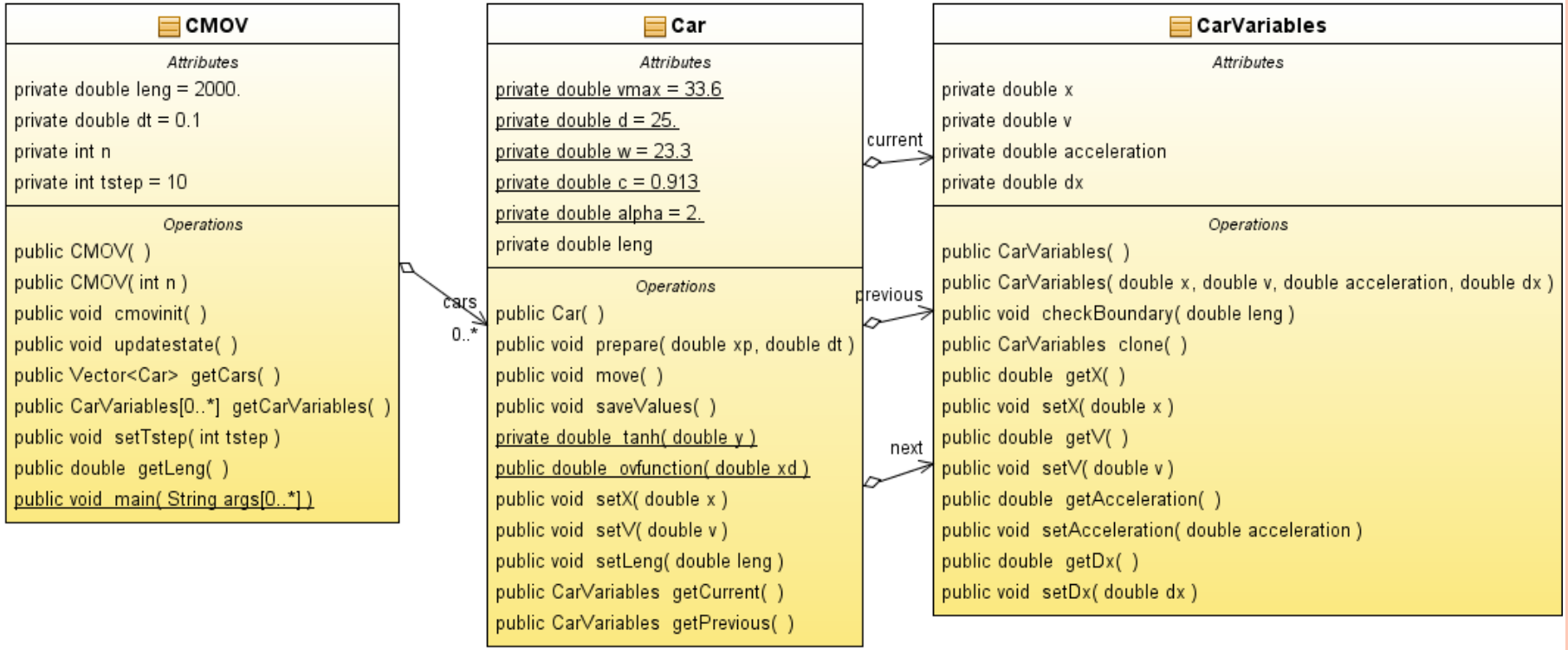
- 例：双曲線正接

$$V_{\text{optimal}}(\Delta x) = \frac{v_{\max}}{2}\left[\tanh\left(2\frac{\Delta x - d}{w}\right) + c\right]$$

全体構成


GUI
• シミュレーションの開始・停止
• 観測方法の選択
• パラメタの設定
シミュレーションのクラス
• サーキットの定義
• 車両群を動かす
一台の車両
一台の車両
一台の車両
一台の車両
一台の車両

CMOVモデル


CMOV
Attributes
private double leng = 2000.
private double dt = 0.1
private int n
private int tstep = 10
Operations
public CMOV( )
public CMOV(int n)
public void cmovinit( )
public void updatestate( )
public Vector<Car> getCars( )
public CarVariables[0..*] getCarVariables( )
public void setTstep(int tstep)
public double getLeng( )
public void main(String args[0..*])
cars
0..*
Car
Attributes
private double vmax = 33.6
private double d = 25.
private double w = 23.3
private double c = 0.913
private double alpha = 2.
private double leng
Operations
public Car( )
public void prepare(double xp, double dt)
public void move( )
public void saveValues( )
private double tanh(double y)
public double ovfunction(double xd)
public void setX(double x)
public void setV(double v)
public void setLeng(double leng)
public CarVariables getCurrent( )
public CarVariables getPrevious( )
current
previous
next
CarVariables
Attributes
private double x
private double v
private double acceleration
private double dx
Operations
public CarVariables( )
public CarVariables(double x, double v, double acceleration, double dx)
public void checkBoundary(double leng)
public CarVariables clone( )
public double getX( )
public void setX(double x)
public double getV( )
public void setV(double v)
public double getAcceleration( )
public void setAcceleration(double acceleration)
public double getDx( )
public void setDx(double dx)

クラスCar

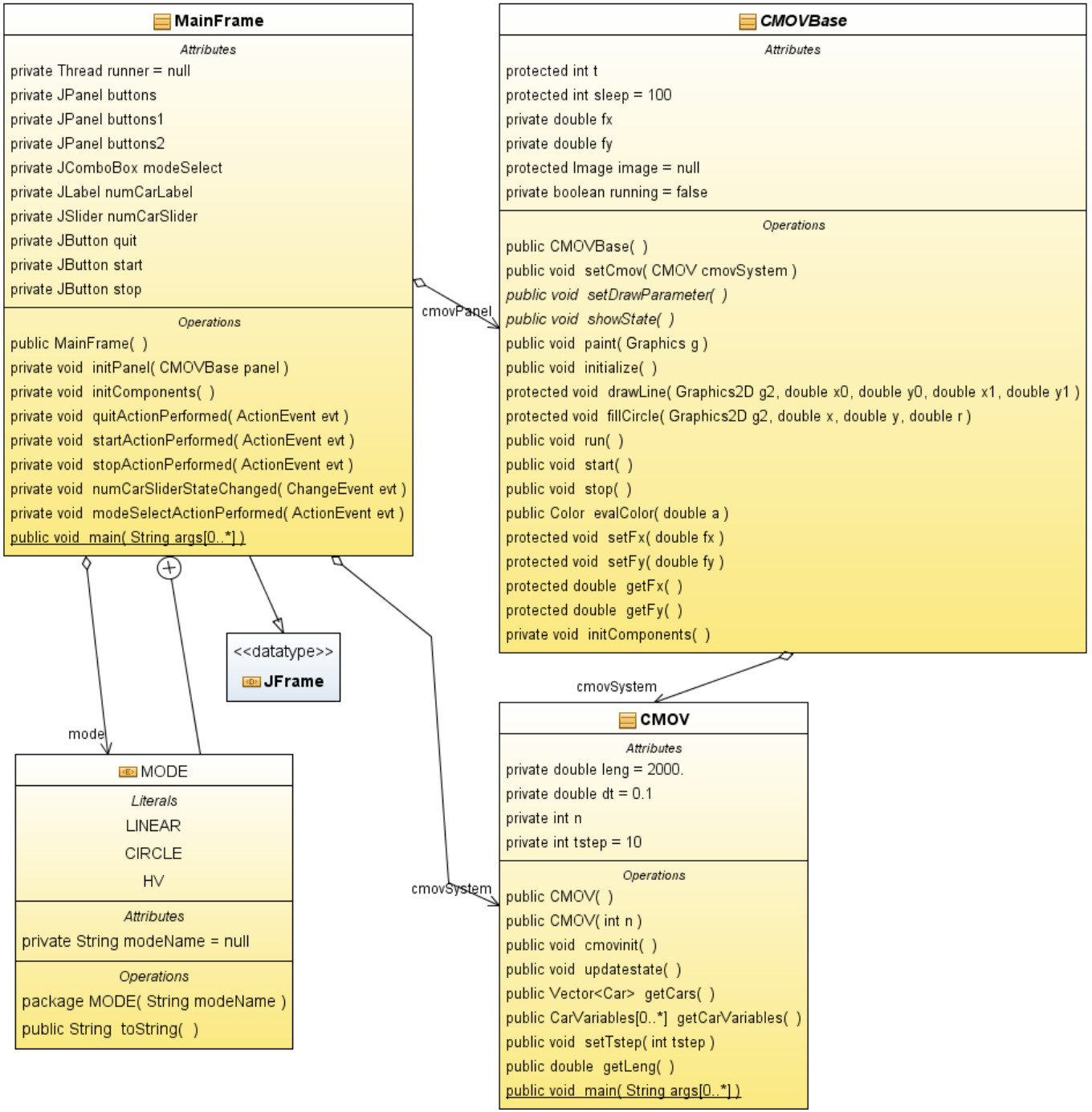
- 一台の車輌のクラス
  - 基本となる変数は位置xと速度v
  - 一つのクラスCarVariablesにまとめる
  - 現在、過去、次の時刻の三種類を準備
  - 次の時刻の量を計算：prepare()
  - 現在の量へ更新：move()
  - 現在の値を過去の値として保存：saveValues();
    - 車輌の軌跡を描くために必要

クラスCMOV

- シミュレーションを実行する
- クラスCarのインスタンスのリスト
  - private java.util.Vector<Car> cars;
- 状態更新手順
  - 現在の値を過去の値として保存
  - 次の時刻の量を計算
  - 現在の量へ更新

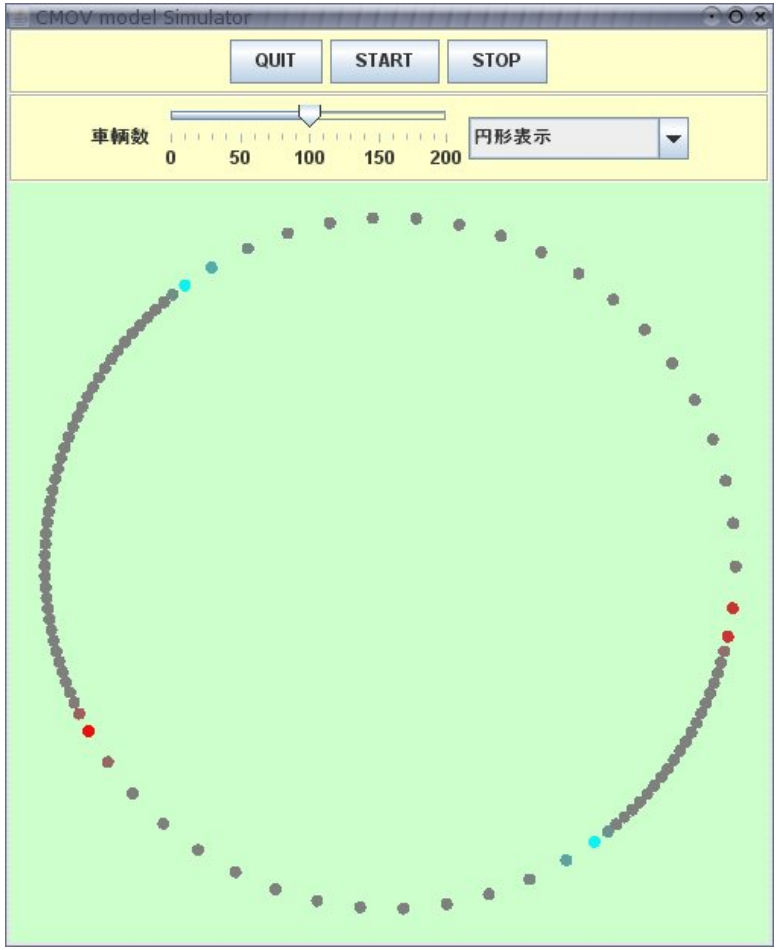
```
public void updatestate() {
    //位置及び速度の保存
    for (int i = 0; i < n; i++) {
        getCars().get(i).saveValues();
    }

    for (int tt = 0; tt < tstep; tt++) {
        //移動準備
        for (int i = 0; i < n - 1; i++) {
            double xp = getCars().get(i + 1).getCurrent().getX();
            getCars().get(i).prepare(xp, dt);
        }
        double xp = getCars().get(0).getCurrent().getX();
        getCars().get(n - 1).prepare(xp, dt);
        //移動実行
        for (Car c : getCars()) {
            c.move();
        }
    }
}
```


MainFrame
Attributes
private Thread runner = null
private JPanel buttons
private JPanel buttons1
private JPanel buttons2
private JComboBox modeSelect
private JLabel numCarLabel
private JSlider numCarSlider
private JButton quit
private JButton start
private JButton stop
Operations
public MainFrame( )
private void initPanel(CMOVBase panel)
private void initComponents( )
private void quitActionPerformed(ActionEvent evt)
private void startActionPerformed(ActionEvent evt)
private void stopActionPerformed(ActionEvent evt)
private void numCarSliderStateChanged(ChangeEvent evt)
private void modeSelectActionPerformed(ActionEvent evt)
public void main(String args[0..*])
CMOVBase
Attributes
protected int t
protected int sleep = 100
private double fx
private double fy
protected Image image = null
private boolean running = false
Operations
public CMOVBase( )
public void setCmov(CMOV cmovSystem)
public void setDrawParameter( )
public void showState( )
public void paint(Graphics g)
public void initialize( )
protected void drawLine(Graphics2D g2, double x0, double y0, double x1, double y1)
protected void fillCircle(Graphics2D g2, double x, double y, double r)
public void run( )
public void start( )
public void stop( )
public Color evalColor(double a)
protected void setFx(double fx)
protected void setFy(double fy)
protected double getFx( )
protected double getFy( )
private void initComponents( )
cmovPanel
<<datatype>>
JFrame
mode
MODE
Literals
LINEAR
CIRCLE
HV
Attributes
private String modeName = null
Operations
package MODE(String modeName)
public String toString( )
cmovSystem
CMOV
Attributes
private double leng = 2000.
private double dt = 0.1
private int n
private int tstep = 10
Operations
public CMOV( )
public CMOV(int n)
public void cmovinit( )
public void updatestate( )
public Vector<Car> getCars( )
public CarVariables[0..*] getCarVariables( )
public void setTstep(int tstep)
public double getLeng( )
public void main(String args[0..*])
cmovSystem

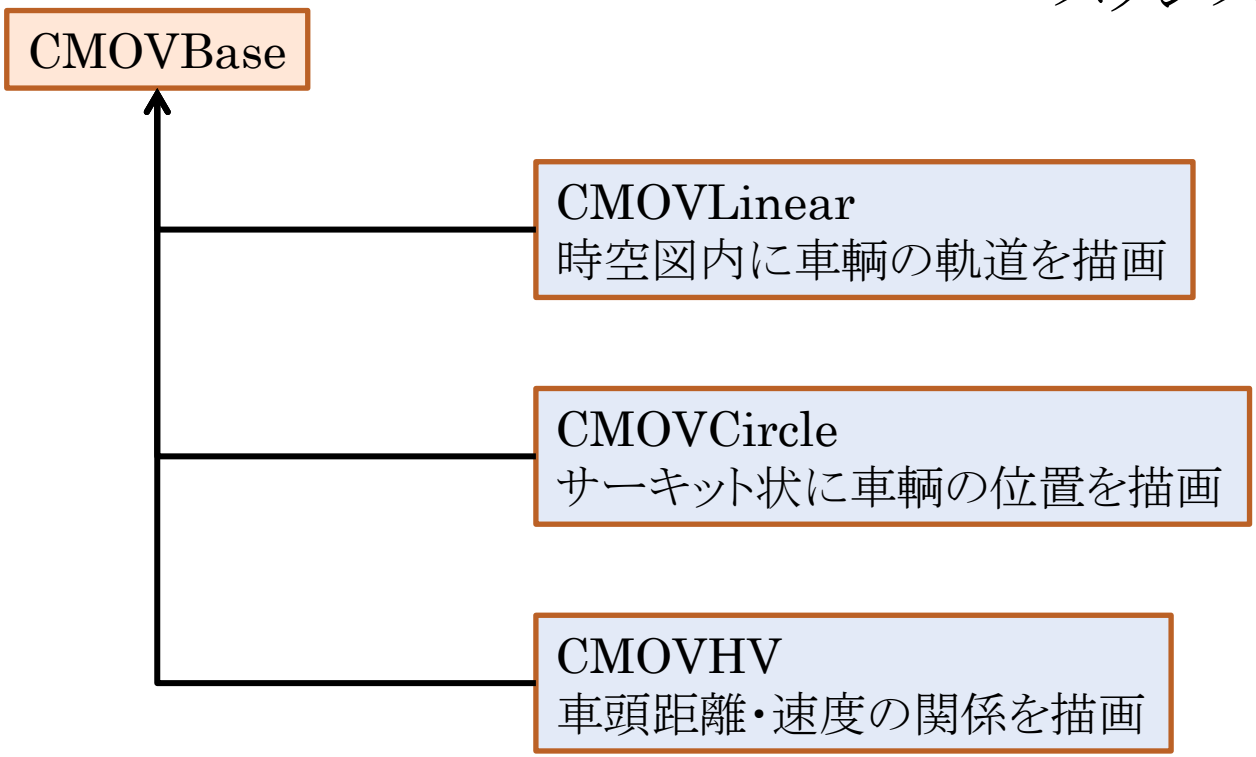
GUIの概要

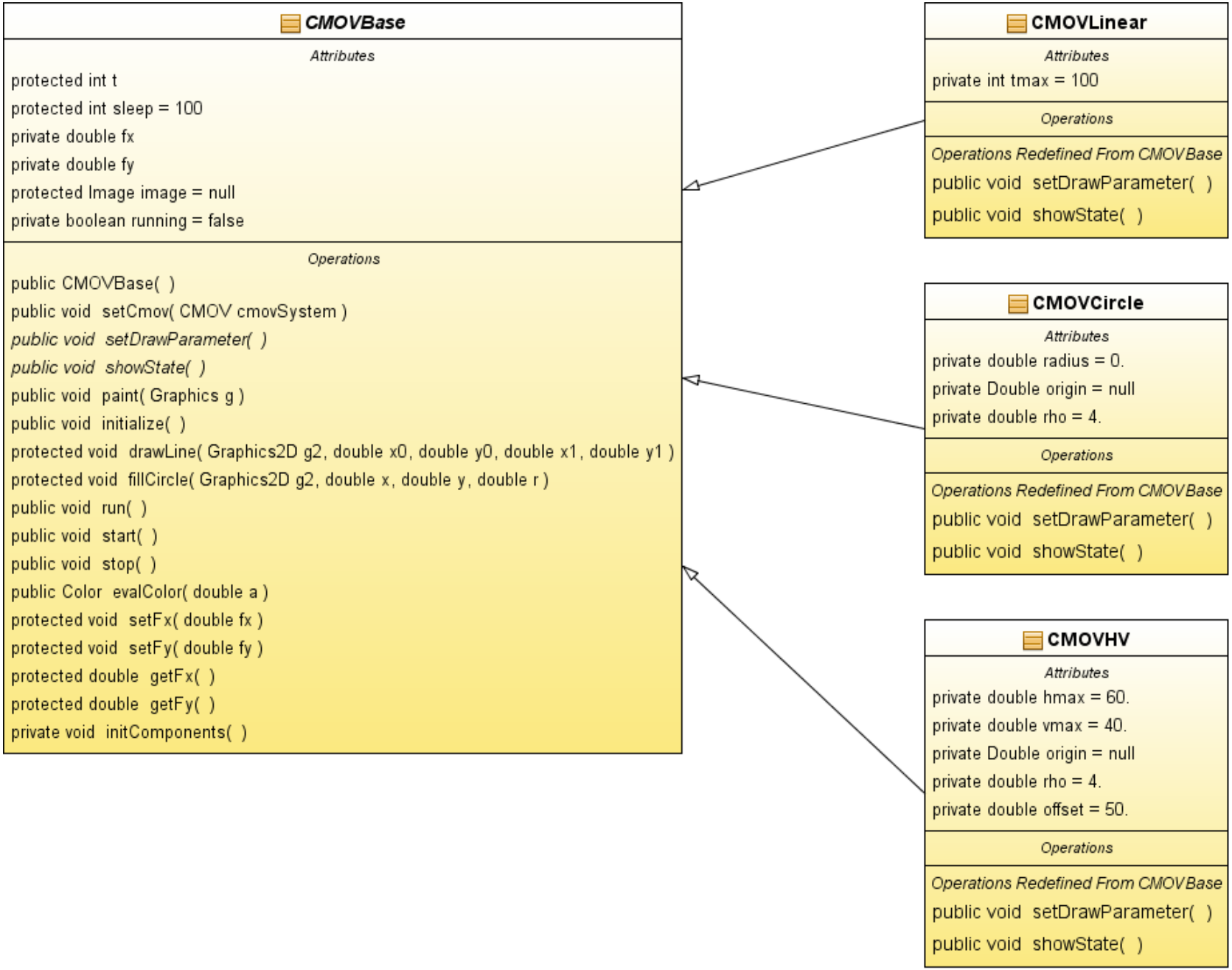
- 全体のフレーム
  - ボタンパネル
    - ボタン
    - ラベル
    - スライダー
    - コンボボックス
- 描画用パネル



描画パネルの階層

MainFrameクラスからすべてCMOVBaseのインスタンスに見せる




CMOVBase
Attributes
protected int t
protected int sleep = 100
private double fx
private double fy
protected Image image = null
private boolean running = false
Operations
public CMOVBase( )
public void setCmov(CMOV cmovSystem)
public void setDrawParameter( )
public void showState( )
public void paint(Graphics g)
public void initialize( )
protected void drawLine(Graphics2D g2, double x0, double y0, double x1, double y1)
protected void fillCircle(Graphics2D g2, double x, double y, double r)
public void run( )
public void start( )
public void stop( )
public Color evalColor(double a)
protected void setFx(double fx)
protected void setFy(double fy)
protected double getFx( )
protected double getFy( )
private void initComponents( )
CMOVLinear
Attributes
private int tmax = 100
Operations
Operations Redefined From CMOVBase
public void setDrawParameter( )
public void showState( )
CMOVCircle
Attributes
private double radius = 0.
private Double origin = null
private double rho = 4.
Operations
Operations Redefined From CMOVBase
public void setDrawParameter( )
public void showState( )
CMOVHV
Attributes
private double hmax = 60.
private double vmax = 40.
private Double origin = null
private double rho = 4.
private double offset = 50.
Operations
Operations Redefined From CMOVBase
public void setDrawParameter( )
public void showState( )

Car.java

```
/**
 * 最適速度模型に従う車両のクラス
 * @author tadaki
 */
package cmov;

public class Car {
    //車輌の変数
    private CarVariables current,  previous,  next;
    private double leng;//コースの長さ

    public Car() {
        current = new CarVariables();
        previous = new CarVariables();
    }

    /**
     * 移動準備
     * @param xp 先行車輌の位置
     * @param dt 時間ステップ
     */
    public void prepare(double xp, double dt) {
        if (xp < getCurrent().getX()) {
            xp += leng;
        }
        getCurrent().setDx(xp - getCurrent().getX());
        getCurrent().setAcceleration(
                CarParameters.alpha * (ovfunction(xp -
getCurrent().getX())
                - getCurrent().getV()));
        double xnew = getCurrent().getX() +
getCurrent().getV() * dt;
        double vnew = getCurrent().getV()
                + getCurrent().getAcceleration() * dt;
        next = new CarVariables(xnew, vnew,
                getCurrent().getAcceleration(),
```

```
 getCurrent().getDx());
    }

    /**
     * 移動の実行
     */
    public void move() {
        current = next.clone();
        getCurrent().checkBoundary(leng);
    }

    public void saveValues() {
        previous = getCurrent().clone();
    }

    static private double tanh(double y) {
        if (y > 0) {
            return (1. - Math.exp(-2 * y)) / (1. +
 Math.exp(-2 * y));
        } else {
            return (Math.exp(2 * y) - 1.) / (1. +
 Math.exp(2 * y));
        }
    }

    static public double ovfunction(double xd) {
        return (1./2.) * CarParameters.vmax
                * (tanh(2. * (xd - CarParameters.d) /
 CarParameters.w)
                + CarParameters.c);
    }

    /********** setters and getters
 ********************************************/
    public void setX(double x) {
        getCurrent().setX(x);
```

```
            getPrevious().setX(x);
        }

        public void setV(double v) {
            getCurrent().setV(v);
            getPrevious().setV(v);
        }

        public void setLeng(double leng) {
            this.leng = leng;
        }

        public CarVariables getCurrent() {
            return current;
        }

        public CarVariables getPrevious() {
            return previous;
        }
    }
```

CMOV.java

```
/**
 * 最適速度モデル
 * @author tadaki
 */
package cmov;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;


public class CMOV {

    private final double leng = 2000.;        //系のサイズ
    private final double dt = 0.1;            //時間ステップ
    private List<Car> cars = null;//車輌リスト
    private int n;                             //車輌数
    private int tstep = 10;                   //一回のupdateで
進める時間回数

    public CMOV() throws IllegalArgumentException {
        String m = "Number of cars must be larger than 10";
        throw new IllegalArgumentException(m);
    }

    public CMOV(int n) throws IllegalArgumentException {
        if (n <= 0) {
            String m = "Number of cars must be larger than
10";
            throw new IllegalArgumentException(m);
        }
        this.n = n;
        cmovinit();
    }
```

```
    /**
     * 初期状態生成
     */
    public void cmovinit() {
        cars = Collections.synchronizedList(new
ArrayList<Car>());
        double dr = getLeng() / n;
        for (int i = 0; i < n; i++) {
            Car car = new Car();
            car.setX(i * dr);
            car.setV(0.);
            car.setLeng(getLeng());
            getCars().add(car);
        }
        int ii = (int) (0.4 * n);
        if (ii >= 0 && ii < getCars().size()) {
            getCars().get(ii).setX(ii * dr - 0.2 * dr);//摂
動
        }
    }

    /**
     * 状態更新
     */
    public void updatestate() {
        //位置及び速度の保存
        for (int i = 0; i < n; i++) {
            getCars().get(i).saveValues();
        }

        for (int tt = 0; tt < tstep; tt++) {
            //移動準備
            for (int i = 0; i < n - 1; i++) {
                double xp = getCars().get(i +
1).getCurrent().getX();
                getCars().get(i).prepare(xp, dt);
```

```
                }
                double xp =
 getCars().get(0).getCurrent().getX();
                getCars().get(n - 1).prepare(xp, dt);
                //移動実行
                for (Car c : getCars()) {
                    c.move();
                }
            }
        }
        /********** setters and getters
 ****************************************/

        public List<Car> getCars() {
            return cars;
        }

        public List<CarVariables> getCarVariables() {
            List<CarVariables> vList =
 Collections.synchronizedList(
                    new ArrayList<CarVariables>());
            for (Car c : cars) {
                vList.add(c.getCurrent().clone());
            }
            return vList;
        }

        public void setTstep(int tstep) {
            this.tstep = tstep;
        }

        public double getLeng() {
            return leng;
        }

        /**
```

```
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int n = 100;
        try {
            if (args.length > 0) {
                n = Integer.valueOf(args[0]);
            }
        } catch (NumberFormatException ex) {

Logger.getLogger(CMOV.class.getName()).log(Level.SEVERE,
null, ex);

Logger.getLogger(CMOV.class.getName()).log(Level.INFO,
                    "Number of car is set to {0}",
String.valueOf(n));
        }
        try {
            CMOV cmov = new CMOV(n);
            if (cmov != null) {
                for (int t = 0; t < 100; t++) {
                    cmov.updatestate();
                }
            }
        } catch (IllegalArgumentException ex) {

Logger.getLogger(CMOV.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
}
```

CarParameters.java

```java
/**
 * CMOVモデルの定数群
 * @author tadaki
 */
package cmov;

public class CarParameters {
    //OV関数の定数
    static public final double vmax = 33.6;
    static public final double d = 25.;
    static public final double w = 23.3;
    static public final double c = 0.913;
    static public final double alpha = 2.;

    /**
     * インスタンス作成を許さない
     */
    private CarParameters(){}
}
```

CarVariables.java

```java
/**
 * 車輌変数のクラス
 * @author tadaki
 */
package cmov;

public class CarVariables {

    private double x;
    private double v;
    private double acceleration;
    private double dx;

    public CarVariables() {
        this(0., 0., 0., 0.);
    }

    public CarVariables(double x, double v, double
acceleration, double dx) {
        this.x = x;
        this.v = v;
        this.acceleration = acceleration;
        this.dx = dx;
    }

    public void checkBoundary(double leng) {
        if (x > leng) {//周期境界条件
            x -= leng;
        }
    }

    @Override
    public CarVariables clone() {
        return new CarVariables(getX(), getV(),
getAcceleration(), getDx());
    }
```

```
    /********** setters and getters
**************************************/
    public double getX() {
        return x;
    }

    public void setX(double x) {
        this.x = x;
    }

    public double getV() {
        return v;
    }

    public void setV(double v) {
        this.v = v;
    }

    public double getAcceleration() {
        return acceleration;
    }

    public void setAcceleration(double acceleration) {
        this.acceleration = acceleration;
    }

    public double getDx() {
        return dx;
    }

    public void setDx(double dx) {
        this.dx = dx;
    }
}
```

CMOVBase.java

```
/*
 * CMOVBase.java
 * シミュレーション表示の共通クラス
 * Created on 2008/08/26, 9:20
 *
 * @author  tadaki
 */
package gui;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Shape;
import java.awt.geom.Ellipse2D;
import java.awt.geom.Line2D;

public abstract class CMOVBase extends javax.swing.JPanel
        implements Runnable {

    protected int t;
    protected int sleep = 100;
    private double fx;
    private double fy;
    protected java.awt.Image image = null;
    protected cmov.CMOV cmovSystem = null;
    private volatile boolean running = false;

    /** Creates new form CMOVBase */
    public CMOVBase() {
        initComponents();
    }

    public void setCmov(cmov.CMOV cmovSystem) {
        this.cmovSystem = cmovSystem;
        setDrawParameter();
```

```
        initialize();
    }

    abstract public void setDrawParameter();

    abstract public void showState();

    @Override
    public void paint(Graphics g) {//状態表示
        if (image == null) {
            return;
        }
        g.drawImage(image, 0, 0, this);
    }

    public void initialize() {
        t = 0;
        Dimension dimension = getPreferredSize();
        image = createImage(dimension.width,
dimension.height);
        if (image != null) {
            Graphics g = image.getGraphics();
            g.setColor(getBackground());
            g.fillRect(0, 0, dimension.width,
dimension.height);
        }
    }

    protected void drawLine(Graphics2D g2,
            double x0, double y0, double x1, double y1) {
        Shape line = new Line2D.Double(x0, y0, x1, y1);
        g2.draw(line);
    }

    protected void fillCircle(Graphics2D g2, double x,
double y, double r) {
```

```
        Ellipse2D.Double q = new Ellipse2D.Double(x, y, r,
 r);
        g2.fill(q);
    }

    @Override
    public void run() {
        while (running) {
            showState();
            repaint();
            try {
                Thread.sleep(sleep);
            } catch (InterruptedException ex) {
            }
        }
    }

    public void start() {
        running = true;
    }

    public void stop() {
        running = false;
    }

    public Color evalColor(double a) {
        int cc = (int) (128. * (1. + a / 20.)) + 256;
        int red = 256 - cc % 256;
        int blue = cc % 256;
        return new Color(red, blue, blue);
    }

    protected void setFx(double fx) {
        this.fx = fx;
    }
```

```
    protected void setFy(double fy) {
        this.fy = fy;
    }

    protected double getFx() {
        return fx;
    }

    protected double getFy() {
        return fy;
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of
this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed"
desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        setBackground(new java.awt.Color(204, 255, 204));
        setForeground(new java.awt.Color(0, 153, 153));
        setMinimumSize(new java.awt.Dimension(500, 500));
        setPreferredSize(new java.awt.Dimension(500, 500));

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
            .addGap(0, 600, Short.MAX_VALUE)
```

```
        );
        layout.setVerticalGroup(

 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
 LEADING)
             .addGap(0, 600, Short.MAX_VALUE)
        );
    }// </editor-fold>//GEN-END:initComponents
    // Variables declaration - do not
 modify//GEN-BEGIN:variables
    // End of variables declaration//GEN-END:variables
 }
```

CMOVLinear.java

```java
/**
 * 車輌位置を直線上に表示
 * @author tadaki
 */
package gui;

import java.awt.Dimension;
import java.awt.Graphics2D;
import java.util.List;

public class CMOVLinear extends CMOVBase {

    private final int tmax = 100;    //時間の上限

    @Override
    public void setDrawParameter() {
        Dimension dimension = getPreferredSize();
        double leng = cmovSystem.getLeng();
        setFx(1. * dimension.width / leng);
        setFy((double) getPreferredSize().height / tmax);
    }

    @Override
    public void showState() {
        t++;
        if (t == tmax) {
            initialize();
            t = 0;
        }
        cmovSystem.updatestate();
        if (image == null) {
            return;
        }
        Graphics2D g2 = (Graphics2D) image.getGraphics();
        List<cmov.Car> cars = cmovSystem.getCars();
        for (cmov.Car c : cars) {
```

```
                double xx = c.getPrevious().getX();
                double x = c.getCurrent().getX();
                if (x >= xx) {
                    g2.setColor(getForeground());
                    drawLine(g2, getFx() * xx, getFy() * (t -
 1),
                            getFx() * x, getFy() * t);
                }
            }
        }
    }
```

CMOVCircle.java

```
/**
 * 車輌の位置を円形に表示する
 * @author tadaki
 */
package gui;

import java.awt.Dimension;
import java.awt.Graphics2D;
import java.awt.geom.Point2D;
import java.util.List;

public class CMOVCircle extends CMOVBase {

    private double radius = 0.;
    private Point2D.Double origin = null;
    private double rho = 4.;

    @Override
    public void setDrawParameter() {
        Dimension dimension = getPreferredSize();
        double leng = cmovSystem.getLeng();
        radius = leng / 2. / Math.PI;
        setFx(dimension.width / 2 / 1.1 / radius);
        setFy(dimension.height / 2 / 1.1 / radius);
        origin = new Point2D.Double(1.1 * radius, 1.1 *
radius);
        cmovSystem.setTstep(1);
        sleep = 10;
    }

    @Override
    public void showState() {
        initialize();
        cmovSystem.updatestate();
        if (image == null) {
            return;
```

```
            }
            Graphics2D g2 = (Graphics2D) image.getGraphics();
            double leng = cmovSystem.getLeng();
            List<cmov.CarVariables> cars =
                    cmovSystem.getCarVariables();
            for (cmov.CarVariables c : cars) {
                double p = c.getX();
                double l = (p - leng) / leng;
                double x = radius * Math.cos(2 * Math.PI * l);
                double y = radius * Math.sin(2 * Math.PI * l);
                g2.setPaint(evalColor(c.getAcceleration()));
                fillCircle(g2, getFx() * (x + origin.x) - rho,
                        getFy() * (y + origin.y) - rho, 2 *
rho);
            }
        }
}
```

CMOVHV.java

```
/**
 * 車頭距離・速度空間のひょうじ
 * @author tadaki
 */
package gui;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics2D;
import java.awt.geom.Point2D;
import java.util.List;

public class CMOVHV extends CMOVBase {

    private double hmax = 60.;
    private double vmax = 40.;
    private Point2D.Double origin = null;
    private double rho = 4.;
    private double offset = 50.;

    @Override
    public void setDrawParameter() {
        Dimension dimension = getPreferredSize();
        setFx((dimension.width - offset) / hmax);
        setFy(-(dimension.height - offset) / vmax);
        origin = new Point2D.Double(offset,
dimension.height - offset);
        cmovSystem.setTstep(1);
        sleep = 10;
    }

    @Override
    public void showState() {
        initialize();
        cmovSystem.updatestate();
        if (image == null) {
```

```
            return;
        }
        Graphics2D g2 = (Graphics2D) image.getGraphics();
        Dimension dimension = getPreferredSize();
        List<cmov.CarVariables> cars =
cmovSystem.getCarVariables();
        for (cmov.CarVariables c : cars) {
            double dx = c.getDx();
            double v = c.getV();
            g2.setPaint(evalColor(c.getAcceleration()));
            fillCircle(g2, getFx() * dx + origin.x - rho,
                    getFy() * v + origin.y - rho, 2 * rho);
        }

        drawLine(g2, origin.x, origin.y, (double)
dimension.width, origin.y);
        drawLine(g2, origin.x, origin.y, origin.x, 0.);

        g2.setColor(Color.black);
        double dx = 0.1;
        int n = (int) (dimension.width / dx);
        for (int i = 0; i < n - 1; i++) {
            double x = i * dx;
            double y = cmov.Car.ovfunction(x);
            double xx = (i + 1) * dx;
            double yy = cmov.Car.ovfunction(xx);
            drawLine(g2, getFx() * x + origin.x, getFy() *
y + origin.y,
                    getFx() * xx + origin.x, getFy() * yy +
origin.y);
        }

    }
}
```

MainFrame.java

```
/*
 * MainFrame.java
 *
 * Created on 2008/08/26, 9:47
 *
 * @author  tadaki
 */
package gui;

public class MainFrame extends javax.swing.JFrame {
    //表示方法一覧

    public enum MODE {

        LINEAR("線形表示"),
        CIRCLE("円形表示"),
        HV("車頭距離・速度表示");
        private String modeName = null;

        MODE(String modeName) {
            this.modeName = modeName;
        }

        @Override
        public String toString() {
            return modeName;
        }
    }
    private CMOVBase cmovPanel = null;
    private cmov.CMOV cmovSystem = null;
    private Thread runner = null;
    private MODE mode = MODE.LINEAR;

    /** Creates new form MainFrame */
    public MainFrame() {
        initComponents();
```

```
        for (MODE m : MODE.values()) {
            modeSelect.addItem(m);
        }
        initPanel(new CMOVLinear());
    }

    /**
     * 表示パネル初期化
     * @param panel 表示対象パネル
     */
    private void initPanel(CMOVBase panel) {
        cmovPanel = panel;
        cmovPanel.setVisible(true);
        getContentPane().add(cmovPanel,
java.awt.BorderLayout.CENTER);
        pack();
        int n = numCarSlider.getValue();
        try {
            cmovSystem = new cmov.CMOV(n);
        } catch (IllegalArgumentException e) {
            n = 10;
            numCarSlider.setValue(n);
            cmovSystem = new cmov.CMOV(n);
        }
        cmovPanel.setCmov(cmovSystem);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of
this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed"
desc="Generated Code">//GEN-BEGIN:initComponents
```

MainFrame.java

```
    private void initComponents() {

        buttons = new javax.swing.JPanel();
        buttons1 = new javax.swing.JPanel();
        quit = new javax.swing.JButton();
        start = new javax.swing.JButton();
        stop = new javax.swing.JButton();
        buttons2 = new javax.swing.JPanel();
        numCarLabel = new javax.swing.JLabel();
        numCarSlider = new javax.swing.JSlider();
        modeSelect = new javax.swing.JComboBox();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON
_CLOSE);
        setTitle("CMOV model Simulator");

        buttons.setBackground(new java.awt.Color(255, 255,
204));
        buttons.setLayout(new
javax.swing.BoxLayout(buttons,
javax.swing.BoxLayout.Y_AXIS));

        buttons1.setBackground(new java.awt.Color(255, 255,
204));

buttons1.setBorder(javax.swing.BorderFactory.createEtchedBor
der());

        quit.setText("QUIT");
        quit.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                quitActionPerformed(evt);
            }
```

```
        });
        buttons1.add(quit);

        start.setText("START");
        start.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                startActionPerformed(evt);
            }
        });
        buttons1.add(start);

        stop.setText("STOP");
        stop.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                stopActionPerformed(evt);
            }
        });
        buttons1.add(stop);

        buttons.add(buttons1);

        buttons2.setBackground(new java.awt.Color(255, 255,
204));

buttons2.setBorder(javax.swing.BorderFactory.createEtchedBor
der());

        numCarLabel.setText("車輌数");
        buttons2.add(numCarLabel);

        numCarSlider.setBackground(new java.awt.Color(255,
255, 204));
```

```
            numCarSlider.setMajorTickSpacing(50);
            numCarSlider.setMaximum(200);
            numCarSlider.setMinorTickSpacing(10);
            numCarSlider.setPaintLabels(true);
            numCarSlider.setPaintTicks(true);
            numCarSlider.setValue(100);
            numCarSlider.setBorder(new
 javax.swing.border.MatteBorder(null));
            numCarSlider.addChangeListener(new
 javax.swing.event.ChangeListener() {
                public void
 stateChanged(javax.swing.event.ChangeEvent evt) {
                    numCarSliderStateChanged(evt);
                }
            });
            buttons2.add(numCarSlider);

            modeSelect.addActionListener(new
 java.awt.event.ActionListener() {
                public void
 actionPerformed(java.awt.event.ActionEvent evt) {
                    modeSelectActionPerformed(evt);
                }
            });
            buttons2.add(modeSelect);

            buttons.add(buttons2);

            getContentPane().add(buttons,
 java.awt.BorderLayout.NORTH);
        }// </editor-fold>//GEN-END:initComponents

 private void quitActionPerformed(java.awt.event.ActionEvent
 evt) {//GEN-FIRST:event_quitActionPerformed
        this.dispose();
 }//GEN-LAST:event_quitActionPerformed
```

```
 private void
 startActionPerformed(java.awt.event.ActionEvent evt)
 {//GEN-FIRST:event_startActionPerformed
     cmovPanel.start();
     runner = new Thread(cmovPanel);
     runner.start();
 }//GEN-LAST:event_startActionPerformed

 private void stopActionPerformed(java.awt.event.ActionEvent
 evt) {//GEN-FIRST:event_stopActionPerformed
     cmovPanel.stop();
 }//GEN-LAST:event_stopActionPerformed

 private void
 numCarSliderStateChanged(javax.swing.event.ChangeEvent evt)
 {//GEN-FIRST:event_numCarSliderStateChanged
     cmovPanel.stop();
     int n = numCarSlider.getValue();
     cmovSystem = new cmov.CMOV(n);
     cmovPanel.setCmov(cmovSystem);
 }//GEN-LAST:event_numCarSliderStateChanged

 private void
 modeSelectActionPerformed(java.awt.event.ActionEvent evt)
 {//GEN-FIRST:event_modeSelectActionPerformed
     if (cmovPanel == null) {
         return;
     }
     cmovPanel.stop();
     getContentPane().remove(cmovPanel);
     mode = (MODE) modeSelect.getSelectedItem();
     switch (mode) {
         case CIRCLE:
             initPanel(new CMOVCircle());
             break;
```

```
            case HV:
                initPanel(new CMOVHV());
                break;
            default:
                initPanel(new CMOVLinear());
                break;
        }
    }//GEN-LAST:event_modeSelectActionPerformed

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new MainFrame().setVisible(true);
            }
        });
    }
    // Variables declaration - do not
modify//GEN-BEGIN:variables
    private javax.swing.JPanel buttons;
    private javax.swing.JPanel buttons1;
    private javax.swing.JPanel buttons2;
    private javax.swing.JComboBox modeSelect;
    private javax.swing.JLabel numCarLabel;
    private javax.swing.JSlider numCarSlider;
    private javax.swing.JButton quit;
    private javax.swing.JButton start;
    private javax.swing.JButton stop;
    // End of variables declaration//GEN-END:variables
}
```