```java
////////////////////////////////////////////////////
// 課題１　統計クラス設計
////////////////////////////////////////////////////

interface Statistics{
  void setDataArray(int[] a); // 対象のデータ配列の設定
  double getStat1();          // 現在の処理対象のデータの１つ目の統計値の受取
  double getStat2();          // 現在の処理対象のデータの２つ目の統計値の受取
  double getStat3();          // 現在の処理対象のデータの３つ目の統計値の受取
}

// 取得する統計値は順に平均，分散，標準偏差として実装
class MyStatistics implements Statistics{
  private int[] data;
  // 対象のデータ配列の受取
  public void setDataArray(int[] data){
    this.data = data;
  }
  //  １つ目の統計値は平均として実現
  public double getStat1(){
    double ave = 0;;
    try{
      for(int d:data)ave += d;
      ave /= data.length;
    }catch(NullPointerException e){
      ave = -1;
    }
    return ave;
  }
  //  ２つ目の統計値は分散として実現
  public double getStat2(){
    double var = 0;
    double ave = this.getStat1();
    if(ave >= 0){
      for(int d:data)var += Math.pow((d - ave), 2);
      var /= data.length;
    }else{
      var = -1;
    }
    return var;
  }
  //  ３つ目の統計値は標準偏差として実現
  public double getStat3(){
    double std;
    double var = this.getStat2();
    if(var >= 0){
      std = Math.sqrt(var);
    }else{
      std = -1;
    }
    return std;
  }
```

```
53 }
54
55 class Final01{
56   public static void main(String[] args){
57     int[] scores = {78,68,56,93,41,66,82,95,36,89};
58     Statistics mystat = new MyStatistics();
59     mystat.setDataArray(scores);
60     System.out.println("平均は"+mystat.getStat1());
61     System.out.println("分散は"+mystat.getStat2());
62     System.out.println("標準偏差は"+mystat.getStat3());
63   }
64 }
```

```java
////////////////////////////////////////////////////
// 課題２　ロジスティック写像
////////////////////////////////////////////////////
import java.io.*;

class LogisticMapping extends Thread{
  private double para_a;  // 係数a
  private double init_x;  // 初期値x（繰返し0回目の値）
  private int from;       // データ保存する繰返し回数開始
  private int to;         // データ保存する繰返し回数終了
  private double[] ary_x; // データ保存用配列

  // コンストラクタ（各種パラメータの設定）
  public LogisticMapping(double a, double x, int f, int t){
    this.para_a = a;
    this.init_x = x;
    this.from = f;
    this.to = t;
    this.ary_x = new double[this.to - this.from + 1];
  }
  // 係数aの取得
  public double getA(){
    return para_a;
  }
  // データ保存用配列の取得
  public double[] getXs(){
    double[] cpy_x = new double[this.ary_x.length];
    for(int i=0; i<this.ary_x.length; i++)
      cpy_x[i] = this.ary_x[i];
    return cpy_x;
  }
  // ロジスティクス方程式の繰返し計算
  public void run
    double x_crt, x_nxt;
    x_crt = this.init_x;
    for(int i = 0; i<=this.to; i++){
      if(i >= this.from)
        this.ary_x[i - this.from] = x_crt;

      x_nxt = this.para_a * x_crt * (1.0 - x_crt);
      x_crt = x_nxt;
    }
  }
}

class Final02{
  public static void main(String[] args){
    LogisticMapping[][] logmap = new LogisticMapping[50][10];
    // スレッド準備
    for(int a=0; a<logmap.length; a++){
      for(int c=0; c<logmap[0].length; c++){
```

```
52          logmap[a][c] = new LogisticMapping(4.0*a/logmap.length, Math.random(), 150,
    200);
53        }
54      }
55      // スレッド実行
56      for(int a=0; a<logmap.length; a++){
57        for(int c=0; c<logmap[0].length; c++){
58          logmap[a][c].start();
59        }
60      }
61      // スレッド待ち
62      for(int a=0; a<logmap.length; a++){
63        for(int c=0; c<logmap[0].length; c++){
64          try{
65            logmap[a][c].join();
66          }catch(InterruptedException e){}
67        }
68      }
69      // CSVファイルへ出力
70      // フォーマット（行毎）：係数a, x
71      double para_a;
72      double[] ary_x;
73      PrintWriter pw;
74      try{
75        pw = new PrintWriter(new BufferedWriter(new FileWriter("data2.csv")));
76        for(int a=0; a<logmap.length; a++){
77          for(int c=0; c<logmap[0].length; c++){
78            para_a = logmap[a][c].getA();
79            for(double d:logmap[a][c].getXs()){
80              pw.println(para_a+", "+d);
81            }
82          }
83        }
84        pw.close();
85      }catch(IOException e){
86        System.out.println("ファイルにデータが保存できませんでした");
87      }
88    }
89  }
```

```java
////////////////////////////////////////////////
// 課題３　コロナ感性者数解析
////////////////////////////////////////////////
import java.io.*;

class Final03{
  public static void main(String[] args){
    BufferedReader br;              // 入力ファイル
    PrintWriter pw;                 // 出力ファイル
    int[] dailycount = new int[7]; // 直近7日間データ配列
    double ave7days;                // 7日間移動平均
    String strline;                 // 1行分データ
    String strdate;                 // Ｌ日付の取出
    String strcount;                // Ｌ感染者数の取出
    int sum = 0;                    // 総計

    try{
      // ファイルのオープン
      br = new BufferedReader(new FileReader("pcr_positive_daily.csv"));
      pw = new PrintWriter(new BufferedWriter(new FileWriter("data3.csv")));
     // 読み込みファイルのヘッダー読込
      strline = br.readLine();
      // 書き込みファイルのヘッダー保存
      strdate = strline.substring(0, strline.indexOf(','));
      strcount = strline.substring(strline.indexOf(',') + 1, strline.length());
      pw.println(strdate+", "+strcount+", 7日間移動平均, 累計");
      // 入力ファイルの最終行まで繰返し
      while(true){
        strline = br.readLine();
        if(strline == null)break;
        strdate = strline.substring(0, strline.indexOf(','));
        strcount = strline.substring(strline.indexOf(',') + 1, strline.length());
        // 直近7日間データ配列要素を1日後方へ移動
        for(int i=dailycount.length-1; i>0; i--){
          dailycount[i] = dailycount[i-1];
        }
        // 最新のデータの配列の保存と7日間移動平均の計算
        dailycount[0] = Integer.parseInt(strcount);
        ave7days = 0;
        for(int i:dailycount) ave7days += i;
        ave7days /= 7;
        // 累積の計算
        sum += dailycount[0];
        // ファイル出力
        pw.println(strdate+", "+strcount+", "+ave7days+", "+sum);
      }

      // ファイルのクローズ
      br.close();
      pw.close();
```

```
51      }catch(IOException e){
52        System.out.println("入出力エラーが発生しました");
53      }
54
55    }
56  }
```

```java
////////////////////////////////////////////////////
// 課題４　ＢＭＩ計算
////////////////////////////////////////////////////
import java.awt.*;
import java.awt.event.*;

class Final04 extends Frame{
  private TextField tf_height;
  private TextField tf_weight;
  private Label lb_bmi;

  public Final04(){
    // タイトル設定
    super("BMI計算");
    // GUI部品設定
    Label lb_height = new Label("身長(m):");
    Label lb_weight = new Label("体重(kg):");
    tf_height = new TextField(7);
    tf_weight = new TextField(7);
    Button bt = new Button("BMI計算");
    lb_bmi = new Label("<<結果の表示>>");
    // レイアウト設定
    Panel p = new Panel();
    p.add(lb_height);
    p.add(tf_height);
    p.add(lb_weight);
    p.add(tf_weight);
    p.add(bt);
    p.add(lb_bmi);
    add(p);
    // リスナー登録
    bt.addActionListener(new MyActionHandler());
    addWindowListener(new MyWindowHandler());
    // 画面表示
    setSize(600,75);
    setVisible(true);
  }
  // ボタンイベント処理
  private class MyActionHandler implements ActionListener{
    public void actionPerformed(ActionEvent e){
      // ボタンが押されたら身長と体重を読取，BMIを計算
      double height = Double.parseDouble(tf_height.getText());
      double weight = Double.parseDouble(tf_weight.getText());
      double bmi = weight/height/height;
      lb_bmi.setText("BMI は "+String.format("%.2f",bmi)+" です");
    }
  }
  // ウィンドウイベント処理
  private class MyWindowHandler extends WindowAdapter{
    public void windowClosing(WindowEvent e){
      System.exit(0);
```

```
52      }
53    }
54    public static void main(String[] args){
55      Final04 app = new Final04();
56    }
57  }
```

```java
////////////////////////////////////////////////////
// 課題５　ピンボール
////////////////////////////////////////////////////
import java.awt.*;
import java.awt.event.*;

class Final05 extends Frame{
  private MyCanvas cv;    //  ピンボール用キャンバス
  // コンストラクタ
  public Final05(){
    // タイトル設定
    super("ピンボール");
    // 初期設定
    cv = new MyCanvas(600,400);
    // キャンバス貼付け
    add(cv);
    // ウィンドウリスナー登録
    addWindowListener(new MyWindowHandler());
    // 画面表示
    // - キャンバスサイズに調整
    pack();
    Insets is = getInsets();
    setSize(cv.getWidth() + is.left + is.right, cv.getHeight() + is.top +
is.bottom);
    // - 画面の表示
    setVisible(true);
  }
  // キャンバス（ピンボール）クラス
  private class MyCanvas extends Canvas implements Runnable{
    private int width, height;    // キャンバスサイズ
    private int mx, my;           // マウスの位置
    private int bar_x, bar_y;     // ラケット座標（サイズ：横50, 縦10）
    private int ball_x, ball_y;   // ボール位置（直径10）
    private int ball_x_delta;     // ボールのx軸速度ベクトル
    private int ball_y_delta;     // ボールのy軸速度ベクトル
    // コンストラクタ
    public MyCanvas(int w, int h){
      width = w;
      height = h;
      mx = my = 0;
      bar_x = bar_y = 0;
      ball_x = ball_y= 20;
      ball_x_delta = 1;
      ball_y_delta = 1;
      // キャンバスサイズの設定
      setSize(width, height);
      // ボールのアニメーション開始
      Thread th = new Thread(this);
      th.start();
      // マウスイベントリスナー登録
```

```
50      addMouseMotionListener(new MyMotionHandler());
51    }
52    // ボールアニメーションスレッド
53    public void run(){
54      while(true){
55        // キャンバスが表示された状態になったら処理開始
56        if(isShowing()==true){
57          // ラケットの移動
58          bar_x = mx;
59          bar_y = (int)(height*0.8);
60          // ボールの移動
61          ball_x += ball_x_delta;
62          ball_y += ball_y_delta;
63          // 上，左，右の枠に到達したら跳ね返り
64          if(ball_x < 0 || ball_x > width) ball_x_delta *= -1;
65          if(ball_y < 0) ball_y_delta *= -1;
66          // ラケットのラインに到達したら，跳ね返り or 終了
67          if(ball_y >= bar_y){
68            if(ball_x > bar_x - 25 && ball_x < bar_x + 25) ball_y_delta *= -1;
69            else System.exit(0);
70          }
71          // 指定ミリ秒のスリープ
72          try{
73            Thread.sleep(7);
74          }catch(InterruptedException e){}
75          // キャンバスの再描画
76          repaint();
77        }
78      }
79    }
80    // キャンバスの描画
81    public void paint(Graphics g){
82      g.setColor(Color.CYAN);
83      g.fillRect(0, bar_y, width, height-bar_y);
84      g.setColor(Color.DARK_GRAY);
85      g.fillRect(bar_x-25, bar_y-5, 50, 10);
86      g.fillOval(ball_x - 5, ball_y - 5, 10, 10);
87    }
88    // マウスイベント処理
89    private class MyMotionHandler extends MouseAdapter{
90      public void mouseMoved(MouseEvent e){
91        // マウス座標を更新
92        mx = e.getX();
93        my = e.getY();
94      }
95    }
96  }
97  // ウィンドウイベント処理
98  private class MyWindowHandler extends WindowAdapter{
99    public void windowClosing(WindowEvent e){
100     System.exit(0);
101   }
```

```
102   }
103   public static void main(String[] args){
104     Final05 app = new Final05();
105   }
106 }
```