

САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)

Отчёт по лабораторной работе №3  
по курсу «Моделирование информационно-вычислительных систем»

Выполнил:  
Власов Г.В.  
гр.6303-0930301D

Проверила:  
Симонова Е .В.

# ЗАДАНИЕ НА МОДЕЛИРОВАНИЕ

## **5. Императивное управление без приоритетов уведомлений при наличии в системе нескольких таймеров**

В системе имеется один таймер с нулевым смещением показаний, называемый системным. Все остальные таймеры задаются смещением их показаний относительно текущих показаний системного таймера.

1. Включить в календарь уведомление о событии  $EV_i$  под таймер  $T_j$  в абсолютном времени этого таймера.
2. Определить значение текущего времени в шкале таймера  $T_j$ .
3. Оцифровать таймер  $T_j$  абсолютным значением времени  $TT$ , т.е. определить смещение показаний таймера  $T_j$  относительно показаний системного таймера и изменить положение меток соответствующих событий в календаре.
4. Отменить все события, связанные с таймером  $T_j$ .
5. В течение заданного промежутка времени выполнить действия, соответствующие событиям, запланированным в календаре.

# ЛИСТИНГ

## Основной класс

```
import javafx.application.Application;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.stage.Stage;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.*;

public class Main extends Application {
    private Stage primaryStage;
    private Scene scene;
    private Label label, label1, label2, label3, label4, label5, label6;
    private ListView<String>
    listView1, listView2, listView3, eventsInfo1, eventsInfo2, eventsInfo3;
    private int maxtime = 0;
    private Random random = new Random();
    private int STIME = 0;
    private int DELAY = 1500;
    private ArrayList<Timer_> timers = new ArrayList<>(3);
    private ArrayList<Event_> eventsList1, eventsList2, eventsList3;
    private Timer_ timerList1, timerList2, timerList3;
    private String namesOfEvents[] = {"Увеличить на 50 процентов", "Уменьшить
    в 2 раза", "Не увеличивать",
        "Прокрутить на 90 градусов", "Прокрутить на 180 градусов",
        "Прокрутить на -90 градусов"},};
    private int boundForEvents = namesOfEvents.length;
    private Button buttonRepeat;
    private Button btnClearFirstTimerEvents = new Button("Отменить события");
    private Button btnClearSecondTimerEvents = new Button("Отменить
    события");
    private Button btnClearThirdTimerEvents = new Button("Отменить события");
    private Button btnMainPage = new Button("На главную");
    private Button btnStart = new Button("Поехали!");

    FileInputStream inputCat;
    FileInputStream inputDog;
    FileInputStream inputToupe;
    {
        try {
            inputCat = new
            FileInputStream("/Users/george/Desktop/00П/MIVS/cat.png");
            inputDog = new
            FileInputStream("/Users/george/Desktop/00П/MIVS/dog.png");
            inputToupe = new
            FileInputStream("/Users/george/Desktop/00П/MIVS/toupe.png");
        } catch (FileNotFoundException e) {
```

```

        e.printStackTrace();
    }
}
Image imageCat = new Image(inputCat);
Image imageDog = new Image(inputDog);
Image imageToupe = new Image(inputToupe);
ImageView imageViewDog = new ImageView(imageDog);
ImageView imageViewToupe = new ImageView(imageToupe);
ImageView imageViewCat = new ImageView(imageCat);

private double currentScacleCat = 0.25;
private double currentScacleToupe = 0.25;
private double currentScacleDog = 0.25;
private int currentRorateCat = 0;
private int currentRorateDog = 0;
private int currentRorateToupe = 0;
private ArrayList<Integer> randomArray;

public ArrayList<Event_> getEventsList1() {
    return eventsList1;
}

public ArrayList<Event_> getEventsList2() {
    return eventsList2;
}

public ArrayList<Event_> getEventsList3() {
    return eventsList3;
}

public ArrayList<Event_> createEventList(int numberOfEvents) {
    ArrayList<Event_> arrayList = new ArrayList<>(numberOfEvents);
    int index;
    int time;
    for (int i = 0; i < numberOfEvents; i++) {
        index = random.nextInt(boundForEvents);
        time = random.nextInt(15);
        Event_ event_ = new Event_(namesOfEvents[index], time);
        arrayList.add(event_);
    }
    return arrayList;
}

//Calculate max time in timer
public int CountTime(ArrayList<Event_> arrayList, int Offset) {
    int maxtime = 0;
    for (int i = 0; i < arrayList.size(); i++) {
        if(maxtime < arrayList.get(i).getTime())
            maxtime = arrayList.get(i).getTime();
    }
    return maxtime + Math.abs(Offset);
}

public static void main(String[] args) {
    launch(args);
}

public void __init__() {
    randomArray = new ArrayList<>();
    label = new Label("Системное время = " + STIME);
    buttonRepeat = new Button("Смоделировать ещё раз!");
    buttonRepeat.setVisible(false);
}

```

```

eventsList1 = new ArrayList<>();
eventsList2 = new ArrayList<>();
eventsList3 = new ArrayList<>();
listView1 = new ListView<>();
listView2 = new ListView<>();
listView3 = new ListView<>();
eventsInfo1 = new ListView<>();
eventsInfo2 = new ListView<>();
eventsInfo3 = new ListView<>();
eventsList1 = createEventList(random.nextInt(9)+1);
eventsList2 = createEventList(random.nextInt(9)+1);
eventsList3 = createEventList(random.nextInt(9)+1);
timerList1 = new Timer_"1", 0, getEventsList1();
timerList2 = new Timer_"2", random.nextInt(11) - 6,
getEventsList2();
timerList3 = new Timer_"3", random.nextInt(11) - 6,
getEventsList3();
timers.add(timerList1);
timers.add(timerList2);
timers.add(timerList3);
label1 = new Label("Таймер со смещением = " +
timers.get(0).getOffset());
label2 = new Label("Таймер со смещением = " +
timers.get(1).getOffset());
label3 = new Label("Таймер со смещением = " +
timers.get(2).getOffset());
label4 = new Label("Текущее время = " + timers.get(0).getOffset());
label5 = new Label("Текущее время = " + timers.get(1).getOffset());
label6 = new Label("Текущее время = " + timers.get(2).getOffset());
label1.setTextFill(Color.BLUE);
label2.setTextFill(Color.BLUE);
label3.setTextFill(Color.BLUE);
label4.setTextFill(Color.GREEN);
label5.setTextFill(Color.GREEN);
label6.setTextFill(Color.GREEN);
label.setTextFill(Color.RED);
for (int i = 0; i < timers.size(); i++) {
    for (int j = 0; j < timers.get(i).getEvents().size(); j++) {
        if(i == 0)
eventsInfo1.getItems().add(timers.get(i).getEvents().get(j).getName() + " "
+timers.get(i).getEvents().get(j).getTime());
        else if(i == 1)
eventsInfo2.getItems().add(timers.get(i).getEvents().get(j).getName() +" "+
timers.get(i).getEvents().get(j).getTime());
        else if(i == 2)
eventsInfo3.getItems().add(timers.get(i).getEvents().get(j).getName() +" "+
timers.get(i).getEvents().get(j).getTime());
    }
}
checkMaxTime();
fillMas();
currentScacleCat = 0.25;
currentScacleToupe = 0.25;
currentScacleDog = 0.25;
currentRorateDog = 0;
imageViewDog.setRotate(currentRorateDog);
currentRorateCat = 0;
imageViewCat.setRotate(currentRorateCat);
currentRorateToupe = 0;
imageViewToupe.setRotate(currentRorateToupe);
INT = 0;
label.setVisible(false);

```

```

    }

    public void draw(int timerID, int eventID){
        if(timerID == 0)
        {
            if(timers.get(timerID).getEvents().get(eventID).getName().equals("Увеличить
на 50 процентов")) {
                currentScacleCat*=1.5;
                imageViewCat.setScaleY(currentScacleCat);
                imageViewCat.setScaleX(currentScacleCat);
                timers.get(timerID).getEvents().remove(eventID);
            }
            else if
            (timers.get(timerID).getEvents().get(eventID).getName().equals("Уменьшить в 2
паза")) {
                currentScacleCat*=0.5;
                imageViewCat.setScaleY(currentScacleCat);
                imageViewCat.setScaleX(currentScacleCat);
                timers.get(timerID).getEvents().remove(eventID);
            }
            else if
            (timers.get(timerID).getEvents().get(eventID).getName().equals("Прокрутить на
90 градусов")) {
                currentRorateCat +=90;
                imageViewCat.setRotate(currentRorateCat);
                timers.get(timerID).getEvents().remove(eventID);
            }
            else if
            (timers.get(timerID).getEvents().get(eventID).getName().equals("Прокрутить на
180 градусов")) {
                currentRorateCat +=180;
                imageViewCat.setRotate(currentRorateCat);
                timers.get(timerID).getEvents().remove(eventID);
            }
            else
            if(timers.get(timerID).getEvents().get(eventID).getName().equals("Прокрутить
на -90 градусов")){
                currentRorateCat -=90;
                imageViewCat.setRotate(currentRorateCat);
                timers.get(timerID).getEvents().remove(eventID);
            }
        }
        else if(timerID == 1) {

            if(timers.get(timerID).getEvents().get(eventID).getName().equals("Увеличить
на 50 процентов")) {
                currentScacleToupe*=1.5;
                imageViewToupe.setScaleY(currentScacleToupe);
                imageViewToupe.setScaleX(currentScacleToupe);
                timers.get(timerID).getEvents().remove(eventID);
            }
            else if
            (timers.get(timerID).getEvents().get(eventID).getName().equals("Уменьшить в 2
паза")) {
                currentScacleToupe*=0.5;
                imageViewToupe.setScaleY(currentScacleToupe);
                imageViewToupe.setScaleX(currentScacleToupe);
                timers.get(timerID).getEvents().remove(eventID);
            }
            else if
            (timers.get(timerID).getEvents().get(eventID).getName().equals("Прокрутить на

```

```

90 градусов")) {
    currentRorateToupe +=90;
    imageViewToupe.setRotate(currentRorateToupe);
    timers.get(timerID).getEvents().remove(eventID);
}
else if
(timers.get(timerID).getEvents().get(eventID).getName().equals("Прокрутить на
180 градусов")) {
    currentRorateToupe +=180;
    imageViewToupe.setRotate(currentRorateToupe);
    timers.get(timerID).getEvents().remove(eventID);
}
else if
(timers.get(timerID).getEvents().get(eventID).getName().equals("Прокрутить на
-90 градусов")) {
    currentRorateToupe -=90;
    imageViewToupe.setRotate(currentRorateToupe);
    timers.get(timerID).getEvents().remove(eventID);
}
}
else if(timerID == 2) {

if(timers.get(timerID).getEvents().get(eventID).getName().equals("Увеличить
на 50 процентов")) {
    currentScacleDog*=1.5;
    imageViewDog.setScaleY(currentScacleDog);
    imageViewDog.setScaleX(currentScacleDog);
    timers.get(timerID).getEvents().remove(eventID);
}
else if
(timers.get(timerID).getEvents().get(eventID).getName().equals("Уменьшить в 2
раза")) {
    currentScacleDog*=0.5;
    imageViewDog.setScaleY(currentScacleDog);
    imageViewDog.setScaleX(currentScacleDog);
    timers.get(timerID).getEvents().remove(eventID);
}
else if
(timers.get(timerID).getEvents().get(eventID).getName().equals("Прокрутить на
90 градусов")) {
    currentRorateDog +=90;
    imageViewDog.setRotate(currentRorateDog);
    timers.get(timerID).getEvents().remove(eventID);
}
else if
(timers.get(timerID).getEvents().get(eventID).getName().equals("Прокрутить на
180 градусов")) {
    currentRorateDog +=180;
    imageViewDog.setRotate(currentRorateDog);
    timers.get(timerID).getEvents().remove(eventID);
}
else if
(timers.get(timerID).getEvents().get(eventID).getName().equals("Прокрутить на
-90 градусов")) {
    currentRorateDog -=90;
    imageViewDog.setRotate(currentRorateDog);
    timers.get(timerID).getEvents().remove(eventID);
}
}
}
}

```

*//Choose what and when we should draw*

```

    public void ProcesseEvent(int stime) {
        for (int i = 0; i < 3; i++) {
            if (timers.get(i).getEvents().size() != 0) {
                for (int j = 0; j < timers.get(i).getEvents().size(); j++) {
                    if (timers.get(i).getEvents().get(j).getTime() <=
(timers.get(i).getOffset()+stime)) {
                        draw(i,j);
                    }
                }
            }
        }
    }

    //Choose the biggest amount of time among all timers
    public void checkMaxTime() {
        maxtime = 0;
        STIME = 0;
        if (CountTime(getEventsList1(), timers.get(0).getOffset()) > maxtime)
        {
            maxtime = CountTime(getEventsList1(),timers.get(0).getOffset());
        }
        if (CountTime(getEventsList2(),timers.get(1).getOffset()) > maxtime)
        {
            maxtime = CountTime(getEventsList2(),timers.get(1).getOffset());
        }
        if (CountTime(getEventsList3(),timers.get(2).getOffset()) > maxtime)
        {
            maxtime = CountTime(getEventsList3(),timers.get(2).getOffset());
        }
    }

    public void LabelsVisualisation(){
        LabelStimeUpdate labelStimeUpdate = new
LabelStimeUpdate(maxtime,DELAY,randomArray);
        label.textProperty().bind(labelStimeUpdate.messageProperty());
        Thread th = new Thread(labelStimeUpdate);
        th.setDaemon(true);
        th.start();

        Label1Thread label1Thread = new
Label1Thread(maxtime,timers.get(0).getOffset(),DELAY,randomArray);
        label4.textProperty().bind(label1Thread.messageProperty());
        Thread thlabel1 = new Thread(label1Thread);
        thlabel1.setDaemon(true);
        thlabel1.start();

        Label1Thread label2Thread = new
Label1Thread(maxtime,timers.get(1).getOffset(),DELAY,randomArray);
        label5.textProperty().bind(label2Thread.messageProperty());
        Thread thlabel2 = new Thread(label2Thread);
        thlabel2.setDaemon(true);
        thlabel2.start();

        Label1Thread label3Thread = new
Label1Thread(maxtime,timers.get(2).getOffset(),DELAY,randomArray);
        label6.textProperty().bind(label3Thread.messageProperty());
        Thread thlabel3 = new Thread(label3Thread);
        thlabel3.setDaemon(true);
        thlabel3.start();
    }

    //mass of future values of stime

```



```

//each elemnt in array is the future
//increment for system time
public void fillMas(){
    int sum = 0;
    int i = 0;
    while (sum <= maxtime){
        randomArray.add(random.nextInt(3)+1);
        sum+=randomArray.get(i);
        i++;
    }
}

private Parent positing() {
    btnClearFirstTimerEvents.setVisible(true);
    btnClearSecondTimerEvents.setVisible(true);
    btnClearThirdTimerEvents.setVisible(true);
    buttonRepeat.setVisible(true);
    btnMainPage.setVisible(true);
    btnStart.setVisible(true);
    Pane topLine = new Pane();
    btnClearFirstTimerEvents.setLayoutX(51);
    btnClearFirstTimerEvents.setLayoutY(10);
    btnClearSecondTimerEvents.setLayoutX(296);
    btnClearSecondTimerEvents.setLayoutY(10);
    btnClearThirdTimerEvents.setLayoutX(551);
    btnClearThirdTimerEvents.setLayoutY(10);
    label1.setLayoutX(31);
    label1.setLayoutY(35);
    label2.setLayoutX(281);
    label2.setLayoutY(35);
    label3.setLayoutX(531);
    label3.setLayoutY(35);
    eventsInfo1.setLayoutX(1);
    eventsInfo1.setLayoutY(50);
    eventsInfo1.setPrefSize(250, 300);
    eventsInfo2.setLayoutX(251);
    eventsInfo2.setLayoutY(50);
    eventsInfo2.setPrefSize(250, 300);
    eventsInfo3.setLayoutX(501);
    eventsInfo3.setLayoutY(50);
    eventsInfo3.setPrefSize(250, 300);
    label4.setLayoutX(41);
    label4.setLayoutY(355);
    label5.setLayoutX(301);
    label5.setLayoutY(355);
    label6.setLayoutX(571);
    label6.setLayoutY(355);
    imageViewCat.setScaleX(currentScacleCat);
    imageViewCat.setScaleY(currentScacleCat);
    imageViewCat.setLayoutX(-180);
    imageViewCat.setLayoutY(200);
    imageViewToupe.setScaleX(currentScacleToupe);
    imageViewToupe.setScaleY(currentScacleToupe);
    imageViewToupe.setLayoutX(60);
    imageViewToupe.setLayoutY(200);
    imageViewDog.setScaleX(currentScacleDog);
    imageViewDog.setScaleY(currentScacleDog);
    imageViewDog.setLayoutX(330);
    imageViewDog.setLayoutY(200);
    btnMainPage.setLayoutX(51);
    btnMainPage.setLayoutY(650);
    btnStart.setLayoutX(296);

```

```

        btnStart.setLayoutY(650);
        buttonRepeat.setLayoutX(551);
        buttonRepeat.setLayoutY(650);
        label.setLayoutX(300);
        label.setLayoutY(5);
        label.setFont(Font.font(20));

        topLine.getChildren().addAll(btnClearFirstTimerEvents,
        btnClearSecondTimerEvents, btnClearThirdTimerEvents, label1, label2, label3,
        eventsInfo1, eventsInfo2, eventsInfo3, label4,
        label5, label6, imageViewCat, imageViewToupe, imageViewDog,
        btnStart, btnMainPage, buttonRepeat, label);
        return topLine;
    }

    private int INT = 0; //variable for lambda expression when we run the
simulation
    private Parent startSimulation(){
        __init__();
        BorderPane mainRoot = new BorderPane();
        mainRoot.setTop(positing());
        // clear first timer events
        btnClearFirstTimerEvents.setOnAction(event -> {
            eventsInfo1.getItems().clear();
            listView1.getItems().clear();
            eventsList1.clear();
            checkMaxTime();
            btnClearFirstTimerEvents.setVisible(false);
        });
        // clear second timer events
        btnClearSecondTimerEvents.setOnAction(event -> {
            eventsInfo2.getItems().clear();
            listView2.getItems().clear();
            eventsList2.clear();
            checkMaxTime();
            btnClearSecondTimerEvents.setVisible(false);
        });
        // clear third timer events
        btnClearThirdTimerEvents.setOnAction(event -> {
            eventsInfo3.getItems().clear();
            listView3.getItems().clear();
            eventsList3.clear();
            checkMaxTime();
            btnClearThirdTimerEvents.setVisible(false);
        });
        // run button
        btnStart.setOnAction(event -> {
            btnClearFirstTimerEvents.setVisible(false);
            btnClearSecondTimerEvents.setVisible(false);
            btnClearThirdTimerEvents.setVisible(false);
            btnMainPage.setVisible(false);
            btnStart.setVisible(false);
            buttonRepeat.setVisible(false);
            if(eventsInfo1.getItems().size() == 0 &&
eventsInfo2.getItems().size() == 0 && eventsInfo3.getItems().size() == 0)
            {
                btnMainPage.setVisible(true);
                btnStart.setVisible(true);
                buttonRepeat.setVisible(true);
            }
            else {
                label.setVisible(true);
            }
        });
    }

```

```

        LabelsVisualisation();
        Thread run = new Thread(() -> {
            while (STIME <= maxtime) {
                try {
                    ProccesEvent(STIME);
                    STIME+=randomArray.get(INT);
                    INT++;
                    Thread.sleep(DELAY);
                } catch (InterruptedException ex) {
                }
            }
            buttonRepeat.setVisible(true);
        });
        run.start();
    }
});
// main page return button
btnMainPage.setOnAction(event -> {
    listView1.getItems().clear();
    listView2.getItems().clear();
    listView3.getItems().clear();
    eventsInfo1.getItems().clear();
    eventsInfo2.getItems().clear();
    eventsInfo3.getItems().clear();
    eventsList1.clear();
    eventsList2.clear();
    eventsList3.clear();
    timers.clear();
    buttonRepeat.setVisible(false);
    maxtime = 0;
    STIME = 0;
    try {
        start(primaryStage);
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
});
// repeat button
buttonRepeat.setOnAction(event -> {
    listView1.getItems().clear();
    listView2.getItems().clear();
    listView3.getItems().clear();
    eventsInfo1.getItems().clear();
    eventsInfo2.getItems().clear();
    eventsInfo3.getItems().clear();
    eventsList1.clear();
    eventsList2.clear();
    eventsList3.clear();
    timers.clear();
    buttonRepeat.setVisible(false);
    maxtime = 0;
    STIME = 0;
    primaryStage.setResizable(true);
    scene = new Scene(startSimulation(), 750, 680);
    primaryStage.setScene(scene);
    primaryStage.setResizable(false);
    primaryStage.show();
});
return mainRoot;
}

```

@Override

```

    public void start(Stage stage) throws FileNotFoundException {
        primaryStage = stage;
        primaryStage.setTitle("5 Вариант 3 Лаба ММВС Власов Георгий");
        primaryStage.setResizable(true);

        FileInputStream input = new
FileInputStream("/Users/george/Desktop/00П/MIVS/1.png");
        Image image = new Image(input);
        ImageView imageView = new ImageView(image);

        BorderPane root = new BorderPane();
        root.setBottom(imageView);
        Pane rootPane = new Pane();
        Button startButton = new Button("Смоделировать");
        startButton.setLayoutX(180.0);
        startButton.setLayoutY(1.0);
        rootPane.getChildren().add(startButton);

        Pane imgPane = new Pane();
        imageView.setLayoutX(-235.0);
        imageView.setLayoutY(-155.0);
        imageView.setScaleX(0.5);
        imageView.setScaleY(0.5);
        imgPane.getChildren().add(imageView);

        root.setTop(rootPane);
        root.setBottom(imgPane);
        scene = new Scene(root, 465, 350);
        scene.setFill(Color.WHITE);
        primaryStage.setScene(scene);
        primaryStage.show();
        primaryStage.setResizable(false);
        startButton.setOnAction(event -> {
            primaryStage.setResizable(true);
            scene = new Scene(startSimulation(), 750, 680);
            primaryStage.setScene(scene);
            primaryStage.setResizable(false);
            primaryStage.show();
        });
    }
}

```

## Класс событие

```
public class Event_ {
    private String name;
    private int time;

    public Event_() {
        this.name = "";
        this.time = 0;
    }

    public Event_(String Name, int Time){
        name = Name;
        time = Time;
    }

    public String getName() {
        return name;
    }

    public int getTime() {
        return time;
    }
}
```

## Класс таймер

```
import java.util.ArrayList;
import java.util.List;

public class Timer_ {
    private String id;
    private int offset;
    public ArrayList<Event_> events;

    public Timer_(String id, int offset, ArrayList<Event_> events) {
        this.id = id;
        this.offset = offset;
        this.events = events;
    }

    public int getOffset() {
        return offset;
    }

    public List<Event_> getEvents() {
        return events;
    }
}
```

## Вспомогательные классы-потоки для обновления графического интерфейса

```
import javafx.concurrent.Task;
import java.util.ArrayList;

public class Label1Thread extends Task<Void> {
    private int MAXTIME;
    private int OFFSET;
    private int DELAY;
    private ArrayList<Integer> RANDOMARRAY;
    private int STIME = 0;
    public Label1Thread(int MaxTime, int Offset, int delay, ArrayList<Integer>
random) {
        MAXTIME = MaxTime;
        OFFSET = Offset;
        DELAY = delay;
        RANDOMARRAY = random;
        STIME=OFFSET;
        MAXTIME+=OFFSET;
    }
    @Override
    protected Void call() throws Exception {
        int i = 0;
        while(STIME <= MAXTIME)
        {
            updateMessage("Текущее время: " + (STIME));
            STIME+=RANDOMARRAY.get(i);
            i++;
            Thread.sleep(DELAY);
        }
        return null;
    }
}

import javafx.concurrent.Task;
import java.util.ArrayList;
public class LabelStimeUpdate extends Task<Void> {
    private int MAXTIME;
    private int DELAY;
    private ArrayList<Integer> RANDOMARRAY;
    private int STIME = 0;
    public LabelStimeUpdate(int maxtime, int delay, ArrayList<Integer> array)
{
        MAXTIME = maxtime;
        DELAY = delay;
        RANDOMARRAY = array;
    }
    @Override
    protected Void call() throws Exception {
        int i = 0;
        while (STIME <= MAXTIME){
            updateMessage("Системное время: " + STIME);
            STIME+=RANDOMARRAY.get(i);
            i++;
            Thread.sleep(DELAY);
        }
        return null;
    }
}
```

# РЕЗУЛЬТАТ МОДЕЛИРОВАНИЯ

5 Вариант 3 Лаба МИВС Власов Георгий

Отменить события

Таймер со смещением = 0

Прокрутить на 90 градусов 12

Прокрутить на 90 градусов 3

Не увеличивать 9

Не увеличивать 3

Не увеличивать 10

Прокрутить на 180 градусов 12

Прокрутить на 180 градусов 6

Отменить события

Таймер со смещением = -3

Прокрутить на -90 градусов 13

Уменьшить в 2 раза 6

Не увеличивать 7

Не увеличивать 8

Прокрутить на -90 градусов 1

Прокрутить на 180 градусов 11

Не увеличивать 1

Отменить события

Таймер со смещением = -4

Увеличить на 50 процентов 2

Уменьшить в 2 раза 4

Не увеличивать 0

Уменьшить в 2 раза 8




Прокрутить на 90 градусов 2

Уменьшить в 2 раза 7

Текущее время = 0

Текущее время = -3

Текущее время = -4

На главную

Поехали!

Смоделировать ещё раз!

5 Вариант 3 Лаба МИВС Власов Георгий

Системное время: 15

Таймер со смещением = 0

Таймер со смещением = -3

Таймер со смещением = -4

Прокрутить на 90 градусов 12

Прокрутить на 90 градусов 3

Не увеличивать 9

Не увеличивать 3

Не увеличивать 10

Прокрутить на 180 градусов 12

Прокрутить на 180 градусов 6

Прокрутить на -90 градусов 13

Уменьшить в 2 раза 6

Не увеличивать 7

Не увеличивать 8

Прокрутить на -90 градусов 1

Прокрутить на 180 градусов 11

Не увеличивать 1

Увеличить на 50 процентов 2

Уменьшить в 2 раза 4

Не увеличивать 0

Уменьшить в 2 раза 8




Прокрутить на 90 градусов 2

Уменьшить в 2 раза 7

Текущее время: 15

Текущее время: 12

Текущее время: 11

Смоделировать ещё раз!