МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Магнитогорский государственный технический университет

им. Г. И. Носова»

(ФГБОУ ВО «МГТУ им. Г.И. Носова»)

Кафедра вычислительной техники и программирования

**РЕФЕРАТ**

по дисциплине «Теория автоматов»

на тему: «Реализация электронных часов с будильником»

Исполнитель: Варламов М.Н. студент 2 курса, группа АВб-19-1

Руководитель: Ильина Е.А., доцент каф. ВТиП, к.п.н.

Магнитогорск,2021

**Оглавление**

[ФОРМЫ ПРЕДСТАВЛЕНИЯ АВТОМАТА 3](#_Toc67124526)

[Спецификация разработанного будильника 3](#_Toc67124527)

[Представление в графовой форме 4](#_Toc67124528)

[Представление в табличной форме 5](#_Toc67124529)

[ОБЪЕКТНЫЙ ПОДХОД 6](#_Toc67124530)

[ПРОЦЕДУРНЫЙ ПОДХОД 7](#_Toc67124531)

[ЗАКЛЮЧЕНИЕ 8](#_Toc67124532)

[ПРИЛОЖЕНИE 9](#_Toc67124533)

# ФОРМЫ ПРЕДСТАВЛЕНИЯ АВТОМАТА

## Спецификация разработанного будильника

Существует 3 состояния автомата: OFF, ON, ALARM\_SET. Переход между состояниями осуществляется с помощью кнопки ‘A’.

Будильник начинает свою работу с состояния OFF, который подсвечивается красным цветом. В данном соcстоянии с ним ничего нельзя сделать, только перевести его в состояние ALARM\_SET.

ALARM\_SET устанавливает время срабатывания будильника, поэтому кнопки ‘H’ и ‘M’ становятся активны и увеличивают количество часов и минут соответственно. Для понимания пользователя данное состояние характеризуется синим цветом. После того, как время было установлено осуществляется переход в следующее состояние.

ON – состояние, в котором происходит работа часов. В данном состоянии можно выполнять установку времени. Кнопки ‘H’ и ‘M’ увеличивают показанное время на 1 час и 1 минуту соответственно.

## Представление в графовой форме

На рисунке 1 представлен разработанный автомат в графовой форме.

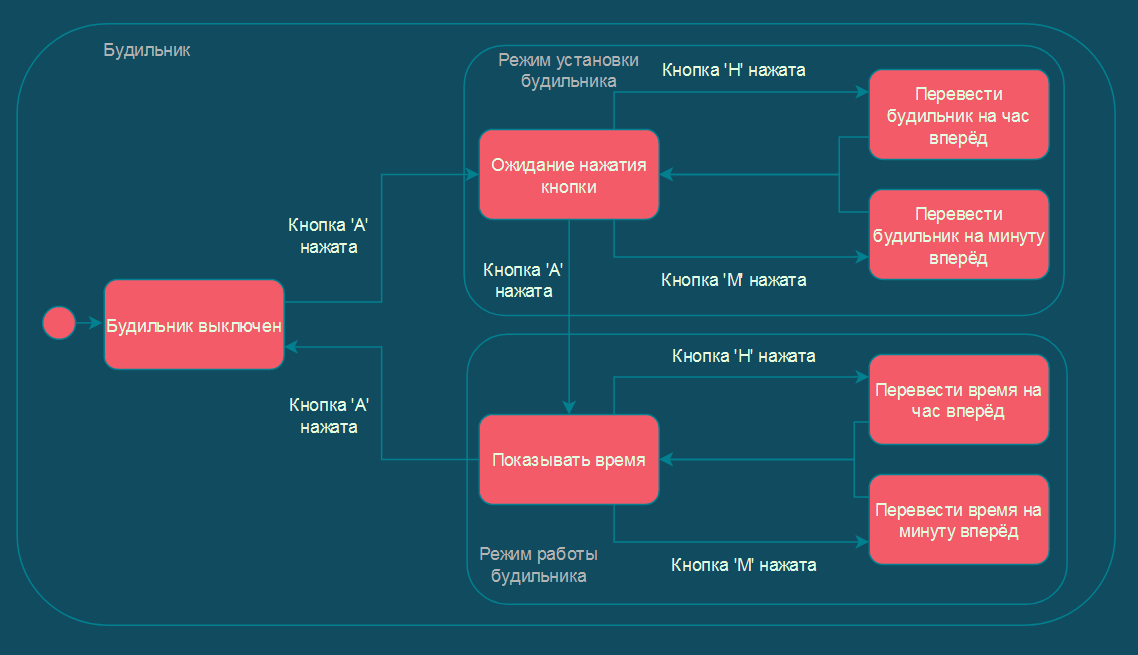


Рисунок 1 – Автомат в графовой форме

## Представление в табличной форме

На рисунке 1 представлен разработанный автомат в табличной форме.

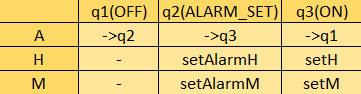


Рисунок 1 – Автомат в графовой форме

Здесь стрелками указан переход к следующему состоянию. setAlarmH и setAlarmM устанавливают время для будильника, во время состояния установки будильника, а setH и setM устанавливают показываемое будильником время, будучи в состоянии ON.

# ОБЪЕКТНЫЙ ПОДХОД

Код программы разделен на 2 пакета – Model и GUI.

Пакет GUI содержит 2 класса – Window и ClockView. Класс Window отвечает за всю графику приложения, а класс ClockView за отображение. ClockView имплементируется от интерфейса ClockListener, чтобы можно было изменять его состояние, в соответствии с классом Clock.

Пакет Model содержит 4 класса – Clock, State, ClockThread и ClockListener. ClockThread – это поток, который считает время и записывает значение в класс Clock. Класс State – это enum, который содержит в себе состояния будильника (OFF, ALARM\_SET, ON).

# ПРОЦЕДУРНЫЙ ПОДХОД

Содержит те же пакеты, что и ОО подход. Однако в пакете Model отсутствует интерфейс ClockListener. Процедурный подход представлен в классе Window.

# ЗАКЛЮЧЕНИЕ

В процессе выполнения лабораторной работы был разработан конечный автомат, выполняющий функции цифрового будильника, а также графический интерфейс, табличное и графовое представление для него. Программа была написана в объектно-ориентированном и процедурном стиле на языке Java.

# ПРИЛОЖЕНИE

**ОО подход**

package Task1.OOP.GUI;

import Task1.OOP.Model.Clock;

import Task1.OOP.Model.ClockListener;

import Task1.OOP.Model.ClockState;

import javax.swing.\*;

import java.awt.\*;

public class ClockView extends JLabel implements ClockListener {

public ClockView(){

super(String.format("%02d",Clock.getH()) + ":" + String.format("%02d",Clock.getM()));

setFont(new Font(Font.MONOSPACED, Font.PLAIN, 25));

}

@Override

public void clockStateChanged(ClockState state) {

switch (state) {

case ON -> setForeground(Color.GREEN);

case OFF -> setForeground(Color.RED);

case ALARM\_SET -> setForeground(Color.BLUE);

}

}

@Override

public void setClockValues() {

setText(String.format("%02d",Clock.getH()) + ":" + String.format("%02d",Clock.getM()));

}

@Override

public void setAlarmValues() {

setText(String.format("%02d",Clock.getAlarmH()) + ":" + String.format("%02d",Clock.getAlarmM()));

}

}

package Task1.OOP.GUI;

import Task1.OOP.Model.Clock;

import javax.swing.\*;

import java.awt.\*;

public class Window extends Container {

private Clock clock = new Clock();

public Window() {

setLayout(new BoxLayout(this, BoxLayout.Y\_AXIS));

JPanel buttonPanel = new JPanel();

buttonPanel.setLayout(new BoxLayout(buttonPanel, BoxLayout.X\_AXIS)); //ЗАТЕСТИТЬ

JButton aButton = new JButton("A");

JButton hButton = new JButton("H");

JButton mButton = new JButton("M");

buttonPanel.add(aButton);

buttonPanel.add(hButton);

buttonPanel.add(mButton);

ClockView clockView = new ClockView();

clockView.setAlignmentX(Component.CENTER\_ALIGNMENT);

clockView.setForeground(Color.RED);

Clock clock = new Clock();

clock.setClockListener(clockView);

add(clockView);

add(buttonPanel);

aButton.addActionListener(clock);

hButton.addActionListener(clock);

mButton.addActionListener(clock);

}

}

package Task1.OOP.Model;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class Clock implements ActionListener {

static ClockListener clockListener;

public static ClockState state = ClockState.OFF;

private static int H = 0, M = 0;

private static int alarmH = 0, alarmM = 0;

public static void turnOn() {

reset();

Clock.state = ClockState.ON;

new ClockThread().start();

}

public static void turnOff() {

Clock.state = ClockState.OFF;

reset();

}

public static void reset() {

setH(0);

setM(0);

}

public static int getH() {

return H;

}

public static int getM() {

return M;

}

public static void setH(int h) {

H = h;

clockListener.setClockValues();

}

public static void setM(int m) {

M = m;

clockListener.setClockValues();

}

public void setClockListener(ClockListener listener) {

clockListener = listener;

}

public static int getAlarmH() {

return alarmH;

}

public static int getAlarmM() {

return alarmM;

}

@Override

public void actionPerformed(ActionEvent e) {

switch (e.getActionCommand()) {

case "A" -> aButtonAction();

case "H" -> alarmButtonsAction("H");

case "M" -> alarmButtonsAction("M");

}

}

private void aButtonAction() {

switch (state) {

case OFF -> Clock.state = ClockState.ALARM\_SET;

case ALARM\_SET -> Clock.turnOn();

case ON -> Clock.turnOff();

}

clockListener.clockStateChanged(state);

}

private void alarmButtonsAction(String type) {

if (Clock.state == ClockState.ALARM\_SET) {

if (type.equals("H"))

alarmH = (alarmH + 1) % 24;

else

alarmM = (alarmM + 1) % 60;

clockListener.setAlarmValues();

}

if (Clock.state == ClockState.ON) {

if (type.equals("H"))

H = (H + 1) % 24;

else

M = (M + 1) % 60;

clockListener.setClockValues();

}

}

}

package Task1.OOP.Model;

public interface ClockListener {

void clockStateChanged(ClockState state);

void setClockValues();

void setAlarmValues();

}

package Task1.OOP.Model;

public enum ClockState {

OFF, ON, ALARM\_SET

}

package Task1.OOP.Model;

import javax.swing.\*;

public class ClockThread extends Thread {

@Override

public void run() {

while (Clock.state == ClockState.ON) {

try {

sleep(600);

Clock.setM((Clock.getM() + 1) % 60);

if (Clock.getM() == 0)

Clock.setH((Clock.getH() + 1) % 24);

if (Clock.getM() == Clock.getAlarmM() && Clock.getH() == Clock.getAlarmH())

JOptionPane.showMessageDialog(null, "Wake Up!","Alarm", JOptionPane.INFORMATION\_MESSAGE);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

**Процедурный подход**

package Task1.Procedure.GUI;

import javax.swing.\*;

import java.awt.\*;

public class ClockView extends JLabel {

public ClockView(){

super(String.format("%02d",0) + ":" + String.format("%02d",0));

setFont(new Font(Font.MONOSPACED, Font.PLAIN, 25));

}

public void setValues(int H, int M) {

setText(String.format("%02d",H) + ":" + String.format("%02d",M));

}

}

package Task1.Procedure.GUI;

import Task1.Procedure.Model.ClockThread;

import Task1.Procedure.Model.Clock;

import Task1.Procedure.Model.ClockState;

import javax.swing.\*;

import java.awt.\*;

public class Window extends Container {

public Window() {

setLayout(new BoxLayout(this, BoxLayout.Y\_AXIS));

JPanel buttonPanel = new JPanel();

buttonPanel.setLayout(new BoxLayout(buttonPanel, BoxLayout.X\_AXIS)); //ЗАТЕСТИТЬ

JButton aButton = new JButton("A");

JButton hButton = new JButton("H");

JButton mButton = new JButton("M");

buttonPanel.add(aButton);

buttonPanel.add(hButton);

buttonPanel.add(mButton);

ClockView clockView = new ClockView();

clockView.setAlignmentX(Component.CENTER\_ALIGNMENT);

clockView.setForeground(Color.RED);

add(clockView);

add(buttonPanel);

aButton.addActionListener(e -> {

switch (Clock.state) {

case ON -> {

Clock.state = ClockState.OFF;

clockView.setForeground(Color.RED);

Clock.M = 0;

Clock.H = 0;

}

case OFF -> {

Clock.state = ClockState.ALARM\_SET;

clockView.setForeground(Color.BLUE);

}

case ALARM\_SET -> {

Clock.state = ClockState.ON;

clockView.setForeground(Color.GREEN);

new ClockThread(clockView).start();

}

}

});

hButton.addActionListener(e -> {

if (Clock.state == ClockState.ALARM\_SET) {

Clock.alarmH = (Clock.alarmH + 1) % 24;

clockView.setValues(Clock.alarmH, Clock.alarmM);

}

if (Clock.state == ClockState.ON) {

Clock.H = (Clock.H + 1) % 24;

clockView.setValues(Clock.H, Clock.M);

}

});

mButton.addActionListener(e -> {

if (Clock.state == ClockState.ALARM\_SET) {

Clock.alarmM = (Clock.alarmM + 1) % 60;

clockView.setValues(Clock.alarmH, Clock.alarmM);

}

if (Clock.state == ClockState.ON) {

Clock.M = (Clock.M + 1) % 60;

clockView.setValues(Clock.H, Clock.M);

}

});

}

}

package Task1.Procedure.Model;

public class Clock {

public static ClockState state = ClockState.OFF;

public static int H = 0, M = 0;

public static int alarmH = 0, alarmM = 0;

}

package Task1.Procedure.Model;

public enum ClockState {

OFF, ON, ALARM\_SET

}

package Task1.Procedure.Model;

import Task1.Procedure.GUI.ClockView;

import javax.swing.\*;

public class ClockThread extends Thread {

ClockView clockView;

public ClockThread(ClockView clockView){

this.clockView = clockView;

}

@Override

public void run() {

while (Clock.state == ClockState.ON) {

try {

sleep(600);

Clock.M = (Clock.M + 1) % 60;

if (Clock.M == 0)

Clock.H = (Clock.H + 1) % 24;

clockView.setValues(Clock.H, Clock.M);

if (Clock.M == Clock.alarmM && Clock.H == Clock.alarmH)

JOptionPane.showMessageDialog(null, "Wake Up!");

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}