```
//import com.sun.xml.internal.bind.v2.runtime.output.StAXExStreamWriterOutput;
package com.company;

import java.awt.Color;
import java.awt.Font;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Calendar;
import java.util.Arrays;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JButton;

import javax.swing.border.LineBorder;

public class Main {
	/*class DynamicDisplay{
		JPanel DynamicDisplayPanel = new JPanel();
		JPanel[] smallPanel= new JPanel[4];
		GridLayout layout= new GridLayout(2,2);
		int settingunit=0;
		int[] eachCurrentmode=new int[]{0,1,2,5};
		int status=0;
		public DynamicDisplay(){
			DynamicDisplayPanel.setSize(300,200);
			DynamicDisplayPanel.setLayout(layout);
			smallPanel[0]=time.ShowTimeKeeping();
			smallPanel[1]=alr.ShowAlarm();
			smallPanel[2]=tmr.ShowTimer();
			smallPanel[3]=wt.ShowWorldtime();
			DynamicDisplayPanel.add(smallPanel[0]);
			DynamicDisplayPanel.add(smallPanel[1]);
			DynamicDisplayPanel.add(smallPanel[2]);
			DynamicDisplayPanel.add(smallPanel[3]);
			smallPanel[0].setBorder(new LineBorder(Color.black,2));
			smallPanel[1].setBorder(new LineBorder(Color.black,2));
			smallPanel[2].setBorder(new LineBorder(Color.black,2));
			smallPanel[3].setBorder(new LineBorder(Color.black,2));
		}
		public void changefoucus(){
			status=1;
				if(settingunit<4)
				{
					settingunit++;
				}
				else if(settingunit>=4){
					settingunit=1;
				}
				switch (settingunit) {
					case 0:
						smallPanel[eachCurrentmode[0]].setBackground(Color.red);
						smallPanel[eachCurrentmode[1]].setBackground(Color.BLACK);
						smallPanel[eachCurrentmode[2]].setBackground(Color.BLACK);
						smallPanel[eachCurrentmode[3]].setBackground(Color.BLACK);
						break;
```

```
                case 1:
                        smallPanel[eachCurrentmode[0]].setBackground(Color.BLACK);
                        smallPanel[eachCurrentmode[1]].setBackground(Color.red);
                        smallPanel[eachCurrentmode[2]].setBackground(Color.BLACK);
                        smallPanel[eachCurrentmode[3]].setBackground(Color.BLACK);
                        break;
                case 2:
                        smallPanel[eachCurrentmode[0]].setBackground(Color.BLACK);
                        smallPanel[eachCurrentmode[1]].setBackground(Color.BLACK);
                        smallPanel[eachCurrentmode[2]].setBackground(Color.red);
                        smallPanel[eachCurrentmode[3]].setBackground(Color.BLACK);
                        break;
                case 3:
                        smallPanel[eachCurrentmode[0]].setBackground(Color.BLACK);
                        smallPanel[eachCurrentmode[1]].setBackground(Color.BLACK);
                        smallPanel[eachCurrentmode[2]].setBackground(Color.BLACK);
                        smallPanel[eachCurrentmode[3]].setBackground(Color.red);
                        break;
        }

}
public void Changemode(){
        switch(settingunit)
        {
                case 0:
                        if(eachCurrentmode[0]<6)
                        {
                                eachCurrentmode[0]++;
                        }
                        else if(eachCurrentmode[0]==6)
                        {
                                eachCurrentmode[0]=0;
                        }
                        break;
                case 1:
                        if(eachCurrentmode[1]<6)
                        {
                                eachCurrentmode[1]++;
                        }
                        else if(eachCurrentmode[1]==6)
                        {
                                eachCurrentmode[1]=0;
                        }
                        break;
                case 2:
                        if(eachCurrentmode[2]<6)
                        {
                                eachCurrentmode[2]++;
                        }
                        else if(eachCurrentmode[2]==6)
                        {
                                eachCurrentmode[2]=0;
                        }
                        break;
                case 3:
                        if(eachCurrentmode[3]<6)
                        {
```

```
                        eachCurrentmode[3]++;
                    }
                    else if(eachCurrentmode[3]==6)
                    {
                        eachCurrentmode[3]=0;
                    }
                    break;
            }
        }
        public void endsetting(){
            smallPanel[0].setBackground(Color.WHITE);
            smallPanel[1].setBackground(Color.WHITE);
            smallPanel[2].setBackground(Color.WHITE);
            smallPanel[3].setBackground(Color.WHITE);
            settingunit=1;
        }
        public JPanel ShowDynamicDisplay(){
            DynamicDisplayPanel.revalidate();
            DynamicDisplayPanel.repaint();
            switch(eachCurrentmode[0])
            {
                case 0:
                    smallPanel[0]=time.ShowTimeKeeping();
                    break;
                case 1:
                    smallPanel[0]=alr.ShowAlarm();
                    break;
                case 2:
                    smallPanel[0]=tmr.ShowTimer();
                    break;
                case 3:
                    smallPanel[0]=stp.ShowStopwatch();
                    break;
                case 4:
                    smallPanel[0]=ctm.ShowCycleTimer();
                    break;
                case 5:
                    smallPanel[0]=wt.ShowWorldtime();
                    break;
            }
            switch(eachCurrentmode[1])
            {
                case 0:
                    smallPanel[1]=time.ShowTimeKeeping();
                    break;
                case 1:
                    smallPanel[1]=alr.ShowAlarm();
                    break;
                case 2:
                    smallPanel[1]=tmr.ShowTimer();
                    break;
                case 3:
                    smallPanel[1]=stp.ShowStopwatch();
                    break;
                case 4:
                    smallPanel[1]=ctm.ShowCycleTimer();
                    break;
```

```
                    case 5:
                        smallPanel[1]=wt.ShowWorldtime();
                        break;
                }
                switch(eachCurrentmode[2])
                {
                    case 0:
                        smallPanel[2]=time.ShowTimeKeeping();
                        break;
                    case 1:
                        smallPanel[2]=alr.ShowAlarm();
                        break;
                    case 2:
                        smallPanel[2]=tmr.ShowTimer();
                        break;
                    case 3:
                        smallPanel[2]=stp.ShowStopwatch();
                        break;
                    case 4:
                        smallPanel[2]=ctm.ShowCycleTimer();
                        break;
                    case 5:
                        smallPanel[2]=wt.ShowWorldtime();
                        break;
                }
                switch(eachCurrentmode[3])
                {
                    case 0:
                        smallPanel[3]=time.ShowTimeKeeping();
                        break;
                    case 1:
                        smallPanel[3]=alr.ShowAlarm();
                        break;
                    case 2:
                        smallPanel[3]=tmr.ShowTimer();
                        break;
                    case 3:
                        smallPanel[3]=stp.ShowStopwatch();
                        break;
                    case 4:
                        smallPanel[3]=ctm.ShowCycleTimer();
                        break;
                    case 5:
                        smallPanel[3]=wt.ShowWorldtime();
                        break;
                }
                DynamicDisplayPanel.add(smallPanel[0]);
                DynamicDisplayPanel.add(smallPanel[1]);
                DynamicDisplayPanel.add(smallPanel[2]);
                DynamicDisplayPanel.add(smallPanel[3]);
                return DynamicDisplayPanel;
            }
        }*/
        private int CurrentMode =1;
        private Font buttonFont =new Font("궁서",Font.BOLD,10);
        //Font ScreenFont1 =new Font("궁서",Font.BOLD,15);
```

```
    private Font ScreenFont2 =new Font("궁서",Font.BOLD,30);
    private JFrame frame =new JFrame("Digital Watch");
    private JPanel panel= new JPanel();
    private JPanel screenPanel = new JPanel();
    private JButton buttonA = new JButton("MODE");
    private JButton buttonB = new JButton("ADJUST");
    private JButton buttonC = new JButton("START/STOP");
    private JButton buttonD = new JButton("RESET");
    public TimeKeeping time= new TimeKeeping();
    public Alarm alr= new Alarm();
    public Timer tmr= new Timer();
    public Stopwatch stp= new Stopwatch();
    public CycleTimer ctm= new CycleTimer();
    public Worldtime wt= new Worldtime();
    //DynamicDisplay dd= new DynamicDisplay();
    public Main(){
        panel.setLayout(null);
        panel.setBackground(Color.lightGray);
        screenPanel.setBackground(Color.darkGray);
        buttonA.setFont(buttonFont); //Applying font
        buttonB.setFont(buttonFont);
        buttonC.setFont(buttonFont);
        buttonD.setFont(buttonFont);
        screenPanel.setFont(ScreenFont2);
        buttonA.setBounds(300,100,100,100); //set label
        buttonB.setBounds(300,0,100,100);
        buttonC.setBounds(400,100,100,100);
        buttonD.setBounds(400,0,100,100);
        panel.add(buttonA);
        panel.add(buttonB);
        panel.add(buttonC);
        panel.add(buttonD);
        panel.add(screenPanel);
        //frame.add(panel);
        frame.add(panel);
        frame.add(screenPanel);
        frame.setLocation(400,400); //
        frame.setSize(520,240);
        frame.setContentPane(panel);
        panel.setOpaque(true);
        //frame.add(panel);
        frame.setVisible(true);
        tmr.ShowTimer();
        stp.ShowStopwatch();
        alr.ShowAlarm();
        time.ShowTimeKeeping();
        ButtonEvent();
        start();
    }
    public void SwitchMode(){
        switch(CurrentMode)
        {
            case 1:
                CurrentMode =2;
                System.out.println("mode2");
                break;
            case 2:
```

```
                    CurrentMode =3;
                    System.out.println("mode3");
                    break;
                case 3:
                    CurrentMode =4;
                    System.out.println("mode4");
                    break;
                case 4:
                    CurrentMode =5;
                    System.out.println("mode5");
                    break;
                case 5:
                    CurrentMode=6;
                    System.out.println("mode6");
                    break;
                case 6:
                    CurrentMode=1;
                    System.out.println("mode1");
                    break;
                /*case 7:
                    CurrentMode=1;
                    System.out.println("mode1");
                    break;*/
                default:
                    break;
            }
    }
    public void ButtonEvent(){
        buttonA.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                switch(CurrentMode)
                {
                    case 1:
                        if(time.getstatus()==0){
                            SwitchMode();
                        }
                        else if(time.getstatus()==1){
                            time.changeElement();
                        }
                        break;
                    case 2:
                        if(alr.getstatus()==0){
                            SwitchMode();
                        }
                        else if(alr.getstatus()==1){
                            alr.changeElement();
                        }
                        break;
                    case 3:
                        if(tmr.getStatus()==0){
                            SwitchMode();
                        }
                        else if(tmr.getStatus()==1){
                            tmr.changeElement();
                        }
                        break;
```

```
                        case 4:
                                if(stp.getStatus()==0){
                                        SwitchMode();
                                }
                                break;
                        case 5:
                                if(ctm.getStatus()==0){
                                        SwitchMode();
                                }
                                else if(ctm.getStatus()==1){
                                        ctm.changeElement();
                                }
                                break;
                        case 6:
                                if(wt.getStatus()==0){
                                        SwitchMode();
                                } //no reaction, if wt is in setting display
                                break;
                        /*case 7:
                                if(dd.status==0){
                                        SwitchMode();
                                }
                                else if(dd.status==1){
                                        dd.changefoucus();
                                }
                                break;*/
                        default:
                                break;
                }
                display();
        }
});
buttonB.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
                switch(CurrentMode)
                {
                        case 1:
                                if(time.getstatus()==0)
                                {
                                        time.startsetting();
                                        display();
                                }
                                else if(time.getstatus()==1){
                                        time.endsetting();
                                        display();
                                }
                                break; //mode:Timekeeping
                        case 2:
                                if(alr.getstatus()==0 && alr.getIsAlarmOn()==0)
                                {
                                        alr.startsetting();
                                        display();
                                }
                                else if(alr.getstatus()==1 && alr.getIsAlarmOn()==0){
                                        alr.endsetting();
                                        display();
```

```
                        }
                        break; //mode:Alarm
                    case 3:
                        if(tmr.getStatus()==0)
                        {
                            tmr.startSetting();
                            display();
                        }
                        else if(tmr.getStatus()==1){
                            tmr.endSetting();
                            display();
                        }
                        break; //mode:Timer
                    case 4:
                        break; //mode Stopwatch, ButtonB is not used in Stopwatch mode
                    case 5:
                        if(ctm.getStatus()==0)
                        {
                            ctm.startSetting();
                            display();
                        }
                        else if(ctm.getStatus()==1){
                            ctm.endSetting();
                            display();
                        }
                        break; //mode:CycleTimer
                    case 6:
                        if(wt.getStatus()==0){
                            wt.startSetting();
                            display();
                        }
                        else if(wt.getStatus()==1){
                            wt.endSetting();
                            display();
                        }
                        break;
                    /*case 7:
                        if(dd.status==0)
                        {
                            dd.status=1;
                            dd.smallPanel[0].setBackground(Color.red);
                            display();
                        }
                        else if(dd.status==1)
                        {
                            dd.status=0;
                            dd.endsetting();
                            display();
                        }
                        break;*/
                    default:
                        break;
                }
            }
        });
        buttonC.addActionListener(new ActionListener() {
            @Override
```

```
        public void actionPerformed(ActionEvent e) {
            switch(CurrentMode) {
                case 1:
                    if (time.getstatus() == 0) {
                        break; // ButtonC is not used in default display of Timekeeping mode
                    } else if (time.getstatus() == 1) {
                        time.IncreaseElement();

                    }
                    break; //modeTimekeeping
                case 2:
                    if (alr.getstatus() == 0) {
                        alr.activateAlarm();
                        break;
                    } else if (alr.getstatus() == 1) {
                        alr.IncreaseElement();
                    }
                    break; //modeAlarm
                case 3:
                    if (tmr.getStatus() == 0) {

                        tmr.StartandStop();
                    } else if (tmr.getStatus() == 1) {
                        tmr.IncreaseElement();
                    }
                    break;
                case 4:
                    stp.StartandStop();
                    break;
                case 5:
                    if (ctm.getStatus() == 0) {
                        ctm.activateCycleTimer();
                    } else if (ctm.getStatus() == 1) {
                        ctm.IncreaseElement();
                    }
                    break;
                case 6:
                    if(wt.getStatus()==1){
                        wt.ChangeElement();
                    }
                    break;
                /*case 7:
                    if(dd.status==1){
                        dd.Changemode();
                    }
                    break;*/
                default:
                    break;
            }
            display();
        }
    });
    buttonD.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            switch(CurrentMode)
            {
```

```
                        case 1: //No use in TimeKeeping mode
                                break;
                        case 2: //No use in Alarm mode
                                break;
                        case 3:
                                tmr.resetTimer();
                                break;
                        case 4:
                                if(stp.getIsStopwatchActivated()==0)
                                {
                                        stp.resetStopwatch();;
                                }
                                else if(stp.getIsStopwatchActivated()==1)
                                {
                                        stp.StartandStop();
                                        stp.resetStopwatch();;
                                }
                                break;
                        case 5:
                                ctm.resetCycleTimer();
                                break;
                        case 6: //no reaction
                                break;
                        default:
                                break;
                        }
                        display();
                }
        });
}
public void start(){
        while(true){
                time.counttime();
                if(alr.getIsAlarmOn()==1) {
                        alr.checkAlarm(time);
                }
                if(tmr.getIsTimerActivated()==1){
                        tmr.counttimer();
                        //tmr.checkTimer();
                }
                if(stp.getIsStopwatchActivated()==1){
                        stp.countStopwatch();
                }
                ctm.checkCycleTimer();
                wt.getdatefromTimekeeping(time);
                display();
                try{
                        Thread.sleep(1000);
                }catch(InterruptedException e){


                }
        }
}
public void display(){
        switch(CurrentMode) {
                case 1:
```

```
                        screenPanel=time.ShowTimeKeeping();
                        System.out.println("Timekeeping");
                        break;
                case 2:
                        screenPanel=alr.ShowAlarm();
                        System.out.println("Alarm");
                        break;
                case 3:
                        screenPanel=tmr.ShowTimer();
                        System.out.println("Timer");
                        break;
                case 4:
                        screenPanel=stp.ShowStopwatch();
                        System.out.println("Stopwatch");
                        break;
                case 5:
                        screenPanel=ctm.ShowCycleTimer();
                        System.out.println("CycleTimer");
                        break;
                case 6:
                        screenPanel=wt.ShowWorldtime();
                        System.out.println("Worldtime");
                        break;
                default:
                        break;
            }
            panel.add(screenPanel);
            screenPanel.revalidate();
            screenPanel.repaint();
            //panel.revalidate();
            //panel.repaint();
        }
        public static void main(String[] args)
        {
            new Main();
        }
}
class TimeKeeping{
        private Font ScreenFont2 =new Font("궁서",Font.BOLD,30);
        private Font ScreenFont3= new Font("궁서",Font.BOLD,20);
        private Calendar c=Calendar.getInstance();
        private int year = c.get(Calendar.YEAR);
        private int mon = c.get(Calendar.MONTH)+1;
        private int day = c.get(Calendar.DAY_OF_MONTH);
        private int hour = c.get(Calendar.HOUR_OF_DAY);
        private int min = c.get(Calendar.MINUTE);
        private int second =c.get(Calendar.SECOND);
        private int settingunit=1;
        private int status=0; // 0 is default display, 1 is setting display
        private int[] dayarr=new int[]{31,29,31,30,31,30,31,31,30,31,30,31};
        private JPanel TimeKeepingPanel= new JPanel();
        private JLabel OnOff = new JLabel("");
        private JLabel Mode= new JLabel("2");
        private JLabel Year= new JLabel("1222");
        private JLabel Month= new JLabel("3");
        private JLabel Day= new JLabel("$");
        private JLabel Hour= new JLabel("6");
```

```
    private JLabel Minute= new JLabel("7");
    private JLabel Second= new JLabel("8");
    public TimeKeeping(){
        OnOff.setFont(ScreenFont2);
        Mode.setFont(ScreenFont2);
        Year.setFont(ScreenFont3);
        Month.setFont(ScreenFont3);
        Day.setFont(ScreenFont3);
        Hour.setFont(ScreenFont2);
        Minute.setFont(ScreenFont2);
        Second.setFont(ScreenFont2);
        TimeKeepingPanel.setLayout(null);
        TimeKeepingPanel.add(OnOff);
        TimeKeepingPanel.add(Mode);
        TimeKeepingPanel.add(Year);
        TimeKeepingPanel.add(Month);
        TimeKeepingPanel.add(Day);
        TimeKeepingPanel.add(Hour);
        TimeKeepingPanel.add(Minute);
        TimeKeepingPanel.add(Second);
        TimeKeepingPanel.setSize(300,200);
        OnOff.setBounds(20,20,50,40);
        Mode.setBounds(60,20,60,40);
        Year.setBounds(140,20,90,40);
        Month.setBounds(210,20,60,40);
        Day.setBounds(250,20,50,40);
        Hour.setBounds(70,130,60,40);
        Minute.setBounds(130,130,60,40);
        Second.setBounds(190,130,60,40);
        OnOff.setText("");
        Mode.setText("TKP");
    }
    public int getstatus(){
        return status;
    }
    public int getyear(){
        return year;
    }
    public int getmonth(){
        return mon;
    }
    public int getdate(){
        return day;
    }
    public int gethour(){
        return hour;
    }
    public int getminute(){
        return min;
    }
    public int getSecond(){
        return second;
    }

    public JPanel ShowTimeKeeping(){
        Year.setText(Integer.toString(year)+" -");
        Month.setText(Integer.toString(mon)+" -");
```

```
        Day.setText(Integer.toString(day));
        Hour.setText(Integer.toString(hour));
        Minute.setText(Integer.toString(min));
        Second.setText(Integer.toString(second));
        System.out.println(Integer.toString(hour)+" "+Integer.toString(min)+" "+Integer.toString(second));
        return TimeKeepingPanel;
    }
    public void changeElement(){
        if(settingunit==5)
        {
            settingunit=1;
        }
        else if(settingunit<5)
        {
            settingunit++;
        }
        switch (settingunit){
            case 1:
                Hour.setForeground(Color.red);
                Minute.setForeground(Color.black);
                Month.setForeground(Color.black);
                Day.setForeground(Color.black);
                Year.setForeground(Color.black);
                break;
            case 2:
                Hour.setForeground(Color.black);
                Minute.setForeground(Color.red);
                Month.setForeground(Color.black);
                Day.setForeground(Color.black);
                Year.setForeground(Color.black);
                break;
            case 3:
                Hour.setForeground(Color.black);
                Minute.setForeground(Color.black);
                Month.setForeground(Color.black);
                Day.setForeground(Color.black);
                Year.setForeground(Color.red);
                break;
            case 4:
                Hour.setForeground(Color.black);
                Minute.setForeground(Color.black);
                Month.setForeground(Color.red);
                Day.setForeground(Color.black);
                Year.setForeground(Color.black);
                break;
            case 5:
                Hour.setForeground(Color.black);
                Minute.setForeground(Color.black);
                Month.setForeground(Color.black);
                Day.setForeground(Color.red);
                Year.setForeground(Color.black);
                break;
            default:
                break;
        }
    }
    public void IncreaseElement(){
```

```
        switch(settingunit)
        {
            case 1:
                hour++;
                if(hour==24)
                {hour=0;}
                break;
            case 2:
                min++;
                if(min==60)
                {min=0;}
                break;
            case 3:
                year++;
                if(year>9999)
                {year=0;}
                break;
            case 4:
                mon++;
                if(mon==13)
                {mon=1;}
                break;
            case 5:
                day++;
                if(day>dayarr[mon-1])
                {day=1;}
                break;
            default:
                break;
        }
    }
    public void startsetting(){
        status=1;
        Hour.setForeground(Color.red);
    }
    public void endsetting(){
        status=0;
        Hour.setForeground(Color.black);
        Minute.setForeground(Color.black);
        Month.setForeground(Color.black);
        Day.setForeground(Color.black);
        Year.setForeground(Color.black);
        settingunit=1;
    }
    public void counttime(){
        second++;
        if(second==60){
            second=0;
            min++;
        }
        if(min==60){
            min=0;
            hour++;
        }
        if(day>dayarr[mon-1])
        {
            day=1;
```

```
                mon++;
            }
            if(mon>12)
            {
                mon=1;
                year++;
            }
            if(year>9999)
            {
                year=1;
            }
        }
}
class Alarm{
        private Calendar c=Calendar.getInstance();
        //private int year = c.get(Calendar.YEAR); //
        private    int[] mon = new int[4];
        private    int[] day = new int[4];
        private    int[] hour = new int[4];
        private    int[] min = new int[4];
        private    int settingunit=1; // 1 is hour, 2 is min, 3 is mon, 4 is day
        private    int status=0; // 0 is default display, 1 is setting display
        private    int isAlarmOn=0; // 0 is deactivated, 1 is activated
        private    int alarmindex= 1;
        private Font ScreenFont2 =new Font("궁서",Font.BOLD,30);
        private JPanel AlarmPanel= new JPanel();
        private JLabel Alarmindex= new JLabel();
        private JLabel OnOff = new JLabel("Off");
        private JLabel Mode= new JLabel("2");
        private JLabel Month= new JLabel("3");
        private JLabel Day= new JLabel("$");
        private JLabel Hour= new JLabel("6");
        private JLabel Minute= new JLabel("7");
        private Toolkit toolkit=Toolkit.getDefaultToolkit();
        public Alarm(){
            Arrays.fill(mon,c.get(Calendar.MONTH));
            Arrays.fill(day,c.get(Calendar.DATE));
            Arrays.fill(hour,c.get(Calendar.HOUR));
            Arrays.fill(min,c.get(Calendar.MINUTE));
            Alarmindex.setFont(ScreenFont2);
            OnOff.setFont(ScreenFont2);
            Mode.setFont(ScreenFont2);
            Month.setFont(ScreenFont2);
            Day.setFont(ScreenFont2);
            Hour.setFont(ScreenFont2);
            Minute.setFont(ScreenFont2);
            AlarmPanel.setLayout(null);
            AlarmPanel.add(OnOff);
            AlarmPanel.add(Mode);
            AlarmPanel.add(Month);
            AlarmPanel.add(Day);
            AlarmPanel.add(Hour);
            AlarmPanel.add(Minute);
            AlarmPanel.add(Alarmindex);
            AlarmPanel.setSize(300,200);
            OnOff.setBounds(10,20,60,40);
            Mode.setBounds(60,20,70,40);
```

```
        Month.setBounds(140,20,60,40);
        Day.setBounds(185,20,50,40);
        Hour.setBounds(70,130,60,40);
        Minute.setBounds(130,130,60,40);
        Alarmindex.setBounds(150,80,80,40);
        OnOff.setText("off");
        Mode.setText("ALR");
    }
    public int getstatus(){
        return status;
    }
    public int getIsAlarmOn(){
        return isAlarmOn;
    }
    public JPanel ShowAlarm(){
        Month.setText(Integer.toString(mon[alarmindex]));
        Day.setText(Integer.toString(day[alarmindex]));
        Hour.setText(Integer.toString(hour[alarmindex]));
        Minute.setText(Integer.toString(min[alarmindex]));
        Alarmindex.setText("alr"+(alarmindex+1));
        System.out.println(Integer.toString(hour[alarmindex])+" "+Integer.toString(min[alarmindex]));
        return AlarmPanel;
    }
    public void changeElement(){
        if(settingunit==5)
        {
            settingunit=1;
        }
        else if(settingunit<5)
        {
            settingunit++;
        }
        switch (settingunit){
            case 1:
                Hour.setForeground(Color.red);
                Minute.setForeground(Color.black);
                Month.setForeground(Color.black);
                Day.setForeground(Color.black);
                Alarmindex.setForeground(Color.black);
                break;
            case 2:
                Hour.setForeground(Color.black);
                Minute.setForeground(Color.red);
                Month.setForeground(Color.black);
                Day.setForeground(Color.black);
                Alarmindex.setForeground(Color.black);
                break;
            case 3:
                Hour.setForeground(Color.black);
                Minute.setForeground(Color.black);
                Month.setForeground(Color.red);
                Day.setForeground(Color.black);
                Alarmindex.setForeground(Color.black);
                break;
            case 4:
                Hour.setForeground(Color.black);
                Minute.setForeground(Color.black);
```

```
                Month.setForeground(Color.black);
                Day.setForeground(Color.red);
                Alarmindex.setForeground(Color.black);
                break;
            case 5:
                Hour.setForeground(Color.black);
                Minute.setForeground(Color.black);
                Month.setForeground(Color.black);
                Day.setForeground(Color.black);
                Alarmindex.setForeground(Color.red);
                break;
            default:
                break;
        }
    }
    public void IncreaseElement(){
        switch(settingunit)
        {
            case 1:
                hour[alarmindex]++;
                if(hour[alarmindex]==24)
                {hour[alarmindex]=0;}
                break;
            case 2:
                min[alarmindex]++;
                if(min[alarmindex]==60)
                {min[alarmindex]=0;}
                break;
            case 3:
                mon[alarmindex]++;
                if(mon[alarmindex]==13)
                {mon[alarmindex]=1;}
                break;
            case 4:
                day[alarmindex]++;
                if(day[alarmindex]==32)
                {day[alarmindex]=1;}
                break;
            case 5:
                alarmindex++;
                if(alarmindex>=4)
                {
                    alarmindex=0;
                }
                break;
            default:
                break;
        }
    }
    public void activateAlarm(){
        if(isAlarmOn==0)
        {
            isAlarmOn=1;
            OnOff.setText("On");
        }
        else if(isAlarmOn==1)
        {
```

```
                    isAlarmOn=0;
                    OnOff.setText("Off");
                }
        }
        public void startsetting(){
                status=1;
                Hour.setForeground(Color.red);
        }
        public void endsetting(){
                status=0;
                Hour.setForeground(Color.black);
                Minute.setForeground(Color.black);
                Month.setForeground(Color.black);
                Day.setForeground(Color.black);
                Alarmindex.setForeground(Color.black);
                settingunit=1;
        }
        public void checkAlarm(TimeKeeping tmk){
                if(tmk.gethour()==hour[alarmindex] && tmk.getminute()==min[alarmindex]){
                        toolkit.beep();
                }
        }
}
class Timer{
        private Font ScreenFont2 =new Font("궁서",Font.BOLD,30);
        private Font ScreenFont3 =new Font("궁서",Font.BOLD,20);
        private int hour = 0;
        private int min = 0;
        private int second =3;
        private int settingunit=1; //1 is hour, 2 is min
        private int isTimerActivated=0; //0 is deactivated, 1 is activated
        private int status=0; // 0 is default display, 1 is setting
        private Toolkit toolkit=Toolkit.getDefaultToolkit();
        private JPanel TimerPanel= new JPanel();
        private JLabel OnOff = new JLabel("stop");
        private JLabel Day= new JLabel("$");
        private JLabel Hour= new JLabel("6");
        private JLabel Minute= new JLabel("7");
        private JLabel Second= new JLabel("8");
        private JLabel mode= new JLabel("2");
        public Timer(){
                OnOff.setFont(ScreenFont3);
                Hour.setFont(ScreenFont2);
                Minute.setFont(ScreenFont2);
                Second.setFont(ScreenFont2);
                mode.setFont(ScreenFont2);
                TimerPanel.setLayout(null);
                TimerPanel.add(OnOff);
                TimerPanel.add(Hour);
                TimerPanel.add(Minute);
                TimerPanel.add(Second);
                TimerPanel.add(mode);
                TimerPanel.setSize(300,200);
                OnOff.setBounds(10,20,60,40);
                mode.setBounds(80,20,70,40);
                OnOff.setBounds(20,20,50,40);
                Hour.setBounds(70,130,60,40);
```

```
        Minute.setBounds(130,130,60,40);
        Second.setBounds(190,130,60,40);
        mode.setText("TMR");
    }
    public int getStatus(){
        return status;
    }
    public int getIsTimerActivated(){
        return    isTimerActivated;
    }
    public JPanel ShowTimer(){
        Hour.setText(Integer.toString(hour));
        Minute.setText(Integer.toString(min));
        Second.setText(Integer.toString(second));
        System.out.println(Integer.toString(hour)+" "+Integer.toString(min)+" "+Integer.toString(second));
        return TimerPanel;
    }
    public void changeElement(){
        if(settingunit==1)
        {
            settingunit=2;
        }
        else if(settingunit==2)
        {
            settingunit=1;
        }
        switch (settingunit){
            case 1:
                Hour.setForeground(Color.red);
                Minute.setForeground(Color.black);
                break;
            case 2:
                Hour.setForeground(Color.black);
                Minute.setForeground(Color.red);
                break;
            default:
                break;
        }
    }
    public void IncreaseElement(){
        switch(settingunit)
        {
            case 1:
                hour++;
                if(hour==24)
                {hour=0;}
                break;
            case 2:
                min++;
                if(min==60)
                {min=0;}
                break;
            default:
                break;
        }
    }
    public void startSetting(){
```

```
            status=1;
            Hour.setForeground(Color.red);
        }
        public void endSetting(){
            status=0;
            Hour.setForeground(Color.black);
            Minute.setForeground(Color.black);
            Day.setForeground(Color.black);
            settingunit=1;
        }
        public void counttimer(){
            second--;
            if(second==-1){
                second=59;
                min--;
            }
            if(min==-1){
                min=59;
                hour--;
            }
            if(hour==-1)
            {
                hour=0;
                min=0;
                second=0;
                isTimerActivated=0;
                checkTimer();
            }
        }
        public void checkTimer(){
            isTimerActivated=0;
            OnOff.setText("stop");
            toolkit.beep();
        }
        public void resetTimer(){
            hour=0;
            min=0;
            second=0;
            isTimerActivated=0;
            OnOff.setText("stop");
        }
        public void StartandStop(){
            if(hour==0 && min==0 &&second==0)
            {
                return;
            }
            if(isTimerActivated==0) {
                isTimerActivated = 1;
                OnOff.setText("start");
            }else if(isTimerActivated==1)
            {
                isTimerActivated=0;
                OnOff.setText("stop");
            }
        }
}
class Stopwatch{
```

```
    private Font ScreenFont2 =new Font("궁서",Font.BOLD,30);
    private Font ScreenFont3 =new Font("궁서",Font.BOLD,20);
    private int hour = 0;
    private int min = 0;
    private int second =0;
    //private Toolkit toolkit=Toolkit.getDefaultToolkit();
    private JPanel TimerPanel= new JPanel();
    private JLabel OnOff = new JLabel("stop");
    private JLabel Hour= new JLabel("6");
    private JLabel Minute= new JLabel("7");
    private JLabel Second= new JLabel("8");
    private JLabel mode= new JLabel("2");
    private int status=0; // 0 is default display, 1 is setting display
    private int isStopwatchActivated=0; //0 is deactivated, 1 is activated
    public Stopwatch(){
        OnOff.setFont(ScreenFont3);
        Hour.setFont(ScreenFont2);
        Minute.setFont(ScreenFont2);
        Second.setFont(ScreenFont2);
        mode.setFont(ScreenFont2);
        TimerPanel.setLayout(null);
        TimerPanel.add(OnOff);
        TimerPanel.add(Hour);
        TimerPanel.add(Minute);
        TimerPanel.add(Second);
        TimerPanel.add(mode);
        TimerPanel.setSize(300,200);
        mode.setBounds(80,20,70,40);
        OnOff.setBounds(10,20,60,40);
        Hour.setBounds(70,130,60,40);
        Minute.setBounds(130,130,60,40);
        Second.setBounds(190,130,60,40);
        mode.setText("STP");
    }
    public int getStatus(){
        return status;
    }
    public int getIsStopwatchActivated(){
        return isStopwatchActivated;
    }
    public JPanel ShowStopwatch(){
        Hour.setText(Integer.toString(hour));
        Minute.setText(Integer.toString(min));
        Second.setText(Integer.toString(second));
        System.out.println(Integer.toString(hour)+" "+Integer.toString(min)+" "+Integer.toString(second));
        return TimerPanel;
    }
    public void resetStopwatch(){
        hour=0; min=0; second=0;
        isStopwatchActivated=0;
        OnOff.setText("stop");
    }
    public void countStopwatch(){
        second++;
        if(second==60){
            second=0;
            min++;
```

```
            }
            if(min==60){
                min=0;
                hour++;
            }
        }
        public void StartandStop(){
            if(isStopwatchActivated==0)
            {
                isStopwatchActivated=1;
                OnOff.setText("start");
            }
            else if(isStopwatchActivated==1)
            {
                isStopwatchActivated=0;
                OnOff.setText("stop");
            }
        }

}
class CycleTimer{
        private Font ScreenFont2 =new Font("궁서",Font.BOLD,30);
        private Toolkit toolkit=Toolkit.getDefaultToolkit();
        private int hour = 0; // this value is increasing while checking cycle
        private int min = 0;
        private int second =0;
        private int checkhour=0; // using for check next cycle
        private int checkmin=0;
        private int checksecond=0;
        private int settingunit=1; // 1 is hour, 2 is min, 3 is second
        private int status=0; // 0 is default display, 1 is setting display
        private int isCycleTimerOn=0; // 0 is Off, 1 is On
        private JPanel CycleTimerPanel= new JPanel();
        private JLabel OnOff = new JLabel("Off");
        private JLabel Mode= new JLabel("CTM");
        private JLabel Hour= new JLabel("");
        private JLabel Minute= new JLabel("");
        private JLabel Second= new JLabel("");
        public CycleTimer(){
            OnOff.setFont(ScreenFont2);
            Mode.setFont(ScreenFont2);
            Hour.setFont(ScreenFont2);
            Minute.setFont(ScreenFont2);
            Second.setFont(ScreenFont2);
            CycleTimerPanel.setLayout(null);
            CycleTimerPanel.add(OnOff);
            CycleTimerPanel.add(Mode);
            CycleTimerPanel.add(Hour);
            CycleTimerPanel.add(Minute);
            CycleTimerPanel.add(Second);
            CycleTimerPanel.setSize(300,200);
            OnOff.setBounds(20,20,50,40);
            Mode.setBounds(80,20,70,40);
            Hour.setBounds(70,130,60,40);
            Minute.setBounds(130,130,60,40);
            Second.setBounds(190,130,60,40);
        }
```

```
    public int getStatus(){
        return status;
    }
    public int getIsCycleTimerOn(){
        return isCycleTimerOn;
    }
    public JPanel ShowCycleTimer(){
        Hour.setText(Integer.toString(hour));
        Minute.setText(Integer.toString(min));
        Second.setText(Integer.toString(second));
        System.out.println(Integer.toString(hour)+" "+Integer.toString(min)+" "+Integer.toString(second));
        return CycleTimerPanel;
    }
    public void changeElement(){
        if(settingunit==3)
        {
            settingunit=1;
        }
        else if(settingunit<3)
        {
            settingunit++;
        }
        switch (settingunit){
            case 1:
                Hour.setForeground(Color.red);
                Minute.setForeground(Color.black);
                Second.setForeground(Color.black);
                break;
            case 2:
                Hour.setForeground(Color.black);
                Minute.setForeground(Color.red);
                Second.setForeground(Color.black);
                break;
            case 3:
                Hour.setForeground(Color.black);
                Minute.setForeground(Color.black);
                Second.setForeground(Color.red);
                break;
            default:
                break;
        }
    }
    public void IncreaseElement(){
        switch(settingunit)
        {
            case 1:
                hour++;
                if(hour==24)
                {hour=0;}
                break;
            case 2:
                min++;
                if(min==60)
                {min=0;}
                break;
            case 3:
                second++;
```

```
                        if(second==60)
                        {second=0;}
                        break;
                default:
                        break;
            }
        }
        public void startSetting(){
            status=1;
            Hour.setForeground(Color.red);
        }
        public void endSetting(){
            status=0;
            Hour.setForeground(Color.black);
            Minute.setForeground(Color.black);
            Second.setForeground(Color.black);
            settingunit=1;
        }
        public void checkCycleTimer(){
            if(isCycleTimerOn==1) {
                checksecond++;
                if(checksecond==60){
                    checksecond=0;
                    checkmin++;
                }
                if(checkmin==60){
                    checkmin=0;
                    checkhour++;
                }
                if(hour==checkhour && min==checkmin && second==checksecond) // if CycleTimer is On and
setting value is 0:0:0, no beep sound
                {
                    toolkit.beep();
                    checkhour=0;
                    checkmin=0;
                    checksecond=0;
                }
            }
        }
        public void activateCycleTimer(){
            if(isCycleTimerOn==0)
            {
                if(hour==0 && min==0 &&second==0)
                {
                    return;
                }
                isCycleTimerOn=1;
                OnOff.setText("On");
                System.out.println("On");
            }
            else if(isCycleTimerOn==1)
            {
                isCycleTimerOn=0;
                checksecond=0;
                checkmin=0;
                checkhour=0;
                OnOff.setText("Off");
```

```
                    System.out.println("Off");
                }
        }
        public void resetCycleTimer(){
                hour=0;
                min=0;
                second=0;
                checkhour=0;
                checkmin=0;
                checksecond=0;
                OnOff.setText("Off");
                isCycleTimerOn=0;
        }

}
class Worldtime{
        private Font ScreenFont2 =new Font("궁서",Font.BOLD,30);
        private Font ScreenFont3 =new Font("궁서",Font.BOLD,10);
        private Font ScreenFont1 = new Font("궁서",Font.BOLD,15);
        private Calendar c=Calendar.getInstance();
        private int year = c.get(Calendar.YEAR);
        private int mon = c.get(Calendar.MONTH)+1;
        private int day = c.get(Calendar.DAY_OF_MONTH);
        private int hour = c.get(Calendar.HOUR_OF_DAY);
        private int min = c.get(Calendar.MINUTE);
        private int second =c.get(Calendar.SECOND);
        private int[] dayarr=new int[]{31,29,31,30,31,30,31,31,30,31,30,31};
        private int tkyear;
        private int tkmon;
        private int tkday;
        private int tkhour;
        private int tkmin;
        private int tksecond;
        private int setcountry=1;
        private int status=0; // 0 is default display, 1 is setting display
        private JPanel WorldtimePanel= new JPanel();
        private JLabel Mode= new JLabel("WT");
        private JLabel Country= new JLabel("3");
        private JLabel Year= new JLabel("");
        private JLabel Month = new JLabel("");
        private JLabel Day = new JLabel("");
        private JLabel Hour= new JLabel("6");
        private JLabel Minute= new JLabel("7");
        private JLabel Second= new JLabel("8");
        public Worldtime(){
                Mode.setFont(ScreenFont2);
                Country.setFont(ScreenFont3);
                Year.setFont(ScreenFont1);
                Month.setFont(ScreenFont1);
                Day.setFont(ScreenFont1);
                Hour.setFont(ScreenFont2);
                Minute.setFont(ScreenFont2);
                Second.setFont(ScreenFont2);
                WorldtimePanel.setLayout(null);
                WorldtimePanel.add(Mode);
                WorldtimePanel.add(Country);
```

```
        WorldtimePanel.add(Month);
        WorldtimePanel.add(Day);
        WorldtimePanel.add(Hour);
        WorldtimePanel.add(Minute);
        WorldtimePanel.add(Second);
        WorldtimePanel.setSize(300,200);
        Mode.setBounds(60,20,60,40);
        Country.setBounds(140,100,100,20);
        Year.setBounds(140,20,90,40);
        Month.setBounds(210,20,60,40);
        Day.setBounds(250,20,50,40);
        Hour.setBounds(70,130,60,40);
        Minute.setBounds(130,130,60,40);
        Second.setBounds(190,130,60,40);
        Mode.setText("WT");
    }
    public int getStatus(){
        return status;
    }
    public JPanel ShowWorldtime(){
        if(status==0) {
            Country.setForeground(Color.black);
        }
        else if(status==1)
        {
            Country.setForeground(Color.red);
        }
        Year.setText(Integer.toString(year)+" -");
        Month.setText(Integer.toString(mon)+" -");
        Day.setText(Integer.toString(day));
        Hour.setText(Integer.toString(hour));
        Minute.setText(Integer.toString(min));
        Second.setText(Integer.toString(second));
        System.out.println(Integer.toString(hour)+" "+Integer.toString(min)+" "+Integer.toString(second));
        applywt();
        return WorldtimePanel;
    }
    public void ChangeElement(){
        setcountry++;
        if(setcountry==7){setcountry=1;}
        applywt();
    }
    public void startSetting(){
        status=1;
        Country.setForeground(Color.red);
    }
    public void endSetting(){
        status=0;
        Country.setForeground(Color.black);
    }
    public void applywt(){
        switch(setcountry)
        {
            case 1: //Seoul(UTC+9)
                hour=tkhour;
                year=tkyear;
                mon=tkmon;
```

```
                day=tkday;
                min=tkmin;
                second=tksecond;
                Country.setText("Seoul(UTC+09:00)");
                break;
        case 2: //Pyongyang +9
                hour=tkhour;
                year=tkyear;
                mon=tkmon;
                day=tkday;
                min=tkmin;
                second=tksecond;
                Country.setText("Pyongyang(UTC+09:00)");
                break;
        case 3: // +10 Vladivostok
                Country.setText(" Vladivostok(UTC+10:00");
                hour=tkhour+1;
                year=tkyear;
                mon=tkmon;
                day=tkday;
                min=tkmin;
                second=tksecond;
                if(hour>=24)
                {       hour=hour-24; day++;}
                if(day>dayarr[mon-1]){ mon++; day=1;}
                if(mon>12){year++;mon=1;}
                if(year>9999){ year=0;}
                break;
        case 4://Sakhalin
                Country.setText("Sakhalin(UTC+11:00)");
                hour=tkhour+2;
                year=tkyear;
                mon=tkmon;
                day=tkday;
                min=tkmin;
                second=tksecond;
                if(hour>=24)
                {       hour=hour-24; day++;}
                if(day>dayarr[mon-1]){ mon++; day=1;}
                if(mon>12){year++;mon=1;}
                if(year>9999){ year=0;}
                break;
        case 5: //Fiji
                Country.setText("Fiji(UTC+12:00)");
                hour=tkhour+3;
                year=tkyear;
                mon=tkmon;
                day=tkday;
                min=tkmin;
                second=tksecond;
                if(hour>=24)
                {       hour=hour-24; day++;}
                if(day>dayarr[mon-1]){ mon++; day=1;}
                if(mon>12){year++;mon=1;}
                if(year>9999){ year=0;}
                break;
        case 6: //havana(utc-5)
```

```
                        Country.setText("Havana(UTC-05:00)");
                        hour=tkhour-4;
                        year=tkyear;
                        mon=tkmon;
                        day=tkday;
                        min=tkmin;
                        second=tksecond;
                        if(hour<0)
                        {       hour=hour+24; day--;}
                        if(day<1){
                                mon--;
                                if(mon<1)
                                {
                                        year--;
                                        if(year<0)
                                        {
                                                year=9999;
                                        }
                                        mon=12;
                                }
                                day=dayarr[mon-1];
                        }
                        break;
                default:
                        break;
        }
    }
    public void getdatefromTimekeeping(TimeKeeping tkp){
        tkyear=tkp.getyear();
        tkmon=tkp.getmonth();
        tkday=tkp.getdate();
        tkhour=tkp.gethour();
        tkmin =tkp.getminute();
        tksecond=tkp.getSecond();
        applywt();
    }
}
```