Sean Overton SN:6421490

TASK 1: Alarm Management System implementing MVC design

CODE:

```
import java.util.Scanner;
import java.util.ArrayList;

//do whatever but as long as it tests all public methods
class TestCode{
    public static void main(String[] args){
        //create AlarmManagement object
        AlarmManagement controller = new AlarmManagement();

        //call addAlarmFromView()
        int count = 3;
        while(count > 0){
            controller.addAlarmFromView();
            controller.displayAlarmView();
            count--;
        }
    }
}

class AlarmView{
    //returns names of alarms?
    //public methods
    public String[] userCreateAnAlarm(){
        //stores user inputs
        String[] userInputs = new String[3];

        //gathers user input
        Scanner input = new Scanner(System.in);

        //prompts for time
        System.out.println("Please enter a time for alarm (eg. 12:30): ");
        String time = input.nextLine();
        String[] times = time.split(":");
        //assigns hours, mins to userInputs
        userInputs[0] = times[0];
        userInputs[1] = times[1];

        //prompts for alarm name (optional?)
        System.out.println("Please enter a name for alarm (optional): ");
        String name = input.nextLine();

        if(name.equals("")){
            userInputs[2] = null;
        }
        else{
```

```
            userInputs[2] = name;
        }
        return userInputs;
    }

    public int getAlarmIndex(){
        Scanner input = new Scanner(System.in);

        //prompts for input
        System.out.print("Enter the index (NOTE: starts at zero) of the alarm you would
like to view: ");

        //gets int input
        int alarmIndex = Integer.parseInt(input.nextLine());

        return alarmIndex;
    }

    //display alarms using Alarm.toString()
    public void displayAlarm(String alarm){
        System.out.println(alarm);
    }
}

//potentially needs more methods added?
class AlarmManagement{
    //private datafields/instance variables
    private ArrayList<Alarm> alarms;
    private AlarmView view;

    //default constructor
    public AlarmManagement(){
        this.alarms = new ArrayList<Alarm>();
        this.view = new AlarmView();
    }

    //public methods
    public void addAlarm(Alarm alarm){
        alarms.add(alarm);
    }

    public Alarm getAlarmAt(int index){
        return alarms.get(index);
    }

    public void addAlarmFromView(){
        String[] alarmInfo = view.userCreateAnAlarm();
```

```
        //need to convert to integer before instantiating Alarm object
        int hour = Integer.parseInt(alarmInfo[0]);
        int min = Integer.parseInt(alarmInfo[1]);

        //creates alarm object
        Alarm newAlarm;

        //checks if name attribute exists
        if(alarmInfo[2] != null){
            String name = alarmInfo[2];
            newAlarm = new Alarm(name, hour, min);
        }
        else{
            newAlarm = new Alarm(hour, min);
        }

        //adds alarm to the datafield
        addAlarm(newAlarm);
    }

    public void displayAlarmView(){
        int index = view.getAlarmIndex();

        Alarm alarm = this.getAlarmAt(index);

        //convert to string
        String strAlarm = alarm.toString();

        view.displayAlarm(strAlarm);
    }
}

class Alarm{
    //private datafields/instance variables
    private String alarmName;
    private int hour;
    private int min;

    //constructors with parameters with validation
    public Alarm(String name, int hour, int min){
        this.alarmName = name;
        if(hour < 24 && hour >= 0){
            //then it is valid and is set
            this.hour = hour;
        }
        if(min < 60 && min >= 0){
            //then it is valid and is set
            this.min = min;
```

```
        }
    }
    public Alarm(int hour, int min){
        this("", hour, min);
    }

    //default constructor
    public Alarm(){
        this("", 0, 0);
    }

    //public methods
    //accessor methods
    public String getAlarmName(){
        return alarmName;
    }
    public int getHour(){
        return hour;
    }

    public int getMin(){
        return min;
    }

    //modifier methods
    public void setAlarmName(String name){
        this.alarmName = name;
    }

    //validation checking occurs here too
    public void setHour(int hour){
        if(hour < 24 && hour >= 0){
            //then it is valid and is set
            this.hour = hour;
        }
    }

    public void setMin(int min){
        if(min < 60 && min >= 0){
            //then it is valid and is set
            this.min = min;
        }
    }

    //overrided toString method for printing instance object
    public String toString(){
        return getHour() + ":" + getMin() + "    " + getAlarmName();
    }
```

```
}
```

COMPILATION AND TEST:

```
C:\Users\Sean\Desktop\Other stuff\UNI\CSI121 OOP\Labs\Lab2\Task1>javac TestCode.java

C:\Users\Sean\Desktop\Other stuff\UNI\CSI121 OOP\Labs\Lab2\Task1>java TestCode
Please enter a time for alarm (eg. 12:30):
12:30
Please enter a name for alarm (optional):
Lunch
Enter the index (NOTE: starts at zero) of the alarm you would like to view: 0
12:30   Lunch
Please enter a time for alarm (eg. 12:30):
10:15
Please enter a name for alarm (optional):
Morning Tea
Enter the index (NOTE: starts at zero) of the alarm you would like to view: 1
10:15   Morning Tea
Please enter a time for alarm (eg. 12:30):
6:30
Please enter a name for alarm (optional):

Enter the index (NOTE: starts at zero) of the alarm you would like to view: 2
6:30
```

TASK 2: Time management System implementing inheritance

CODE:

```java
class TestCode{
    public static void main(String[] args){
        //test all methods of Time class
        Time time1 = new Time();
        System.out.println("Time object 1: " + time1);
        System.out.println();

        //test all methods of Alarm subclass
        System.out.println("Two Alarm objects created to test different constructors");
        Alarm alarm1 = new Alarm("Lunch time", 12, 0);
        Alarm alarm2 = new Alarm(12, 0);
        System.out.println("Alarm object 1: " + alarm1);
        System.out.println("Alarm object 2: " + alarm2);
        System.out.println("Tests set alarm name: ");
        alarm1.setAlarmName("Dinner time");
        System.out.println("New name of Alarm object 1: " + alarm1);
        System.out.println();

        //test all methods of Timer class
        System.out.println("Three Timer objects created to test different constructors");
        Timer timer1 = new Timer(10, 15, 30);
        Timer timer2 = new Timer(23, 12);
        Timer timer3 = new Timer(12);
        System.out.println("Timer object 1: " + timer1);
        System.out.println("Timer object 2: " + timer2);
        System.out.println("Timer object 3: " + timer3);
```

```
        System.out.println("Testing the multiple set timer methods which also tests methods
in superclass");
        timer1.setTimer(8, 45, 30);
        System.out.println("Timer object 1: " + timer1);
        timer1.setTimer(10, 30);
        System.out.println("Timer object 1: " + timer1);
        timer1.start();
        timer1.stop();
        timer1.reset();

        System.out.println();
        System.out.println("All methods and constructors have been tested either explicitly
or implicitly.");
    }
}

//superclass of timer and alarm
class Time{
    //private data fields/instance variables
    private int hour;
    private int minute;
    private int second;

    //constructors with various parameters and input validation
    public Time(int hour, int minute, int second){
        if(hour >= 0 && hour < 24){
            this.hour = hour;
        }
        if(minute >= 0 && minute < 60){
            this.minute = minute;
        }
        if(second >= 0 && second < 60){
            this.second = second;
        }
    }

    public Time(int hour, int minute){
        this(hour, minute, 0);
    }

    public Time(int hour){
        this(hour, 0, 0);
    }

    //default constructor called when subclass object instantiated
    public Time(){
        this(0, 0, 0);
    }
```

```
    //public methods
    //accessors
    public int getHour(){
        return hour;
    }

    public int getMinute(){
        return minute;
    }

    public int getSecond(){
        return second;
    }

    //modifiers
    public void setHour(int hour){
        if(hour >= 0 && hour < 24){
            this.hour = hour;
        }
    }

    public void setMinute(int minute){
        if(minute >= 0 && minute < 60){
            this.minute = minute;
        }
    }

    public void setSecond(int second){
        if(second >= 0 && second < 60){
            this.second = second;
        }
    }

    public String toString(){
        return Integer.toString(this.getHour()) + ":" + Integer.toString(this.getMinute())
+ ":" + Integer.toString(this.getSecond());
    }
}

//subclasses of Time class
class Alarm extends Time{
    //datafields/instance variables
    private String alarmName;

    //constructors with parameters
    public Alarm(String name, int hour, int min){
        super(hour, min);
```

```
        //uses super class' constructor
        //super should be called first
        this.alarmName = name;
    }

    public Alarm(int hour, int min){
        //calls this class' constructor with 3 parameters
        this("", hour, min);
    }

    //default constructor is not needed N/A as described in design

    //methods
    //accessor
    public String getAlarmName(){
        return alarmName;
    }

    //modifer
    public void setAlarmName(String name){
        this.alarmName = name;
    }

    //this overrides Time toString() method
    public String toString(){
        return this.getAlarmName() + " " + super.toString();
    }
}

class Timer extends Time{
    //constructors
    public Timer(int hour, int min, int second){
        super(hour, min, second);
        //calls superclasses constructors
    }

    public Timer(int hour, int min){
        super(hour, min);
    }

    public Timer(int hour){
        super(hour);
    }

    //modifier methods
    public void setTimer(int hour, int min, int sec){
        super.setHour(hour);
        super.setMinute(min);
```

```
        super.setSecond(sec);
        //superclass methods already has input validation too
    }

    public void setTimer(int hour, int min){
        super.setHour(hour);
        super.setMinute(min);
    }

    //these don't actually need to be implemented
    //can try though
    public void start(){
        System.out.println("Start method is called");
    }

    public void stop(){
        System.out.println("Stop method is called");
    }

    public void reset(){
        System.out.println("Reset method is called");
    }
}
```

COMPILATION AND TEST:

```
C:\Users\Sean\Desktop\Other stuff\UNI\CSI121 OOP\Labs\Lab2\Task2>javac TestCode.java

C:\Users\Sean\Desktop\Other stuff\UNI\CSI121 OOP\Labs\Lab2\Task2>java TestCode
Time object 1: 0:0:0

Two Alarm objects created to test different constructors
Alarm object 1: Lunch time 12:0:0
Alarm object 2:  12:0:0
Tests set alarm name:
New name of Alarm object 1: Dinner time 12:0:0

Three Timer objects created to test different constructors
Timer object 1: 10:15:30
Timer object 2: 23:12:0
Timer object 3: 12:0:0
Testing the multiple set timer methods which also tests methods in superclass
Timer object 1: 8:45:30
Timer object 1: 10:30:30
Start method is called
Stop method is called
Reset method is called

All methods and constructors have been tested either explicitly or implicitly.
```