Your Name: _____ **Student Number:** _____

Lab 6a: to be done in class with a partner – due at the end of this lesson (10 marks)

For this lab you will be using the NumberDisplay class, available on D2L under Week 6. Note: Do not change the NumberDisplay class at all!!!

Create a class called DayDisplay whose only instance variable is **a NumberDisplay object** called dayNumber. Include a javadoc comment above the class including a description, @author tag, and @version tag. Also include symbolic constants for the day numbers and day names, such as:

```
public static final int NUMBER_OF_DAYS_IN_WEEK = 7;

public static final int SUNDAY      = 0;
public static final int MONDAY      = 1;
public static final int TUESDAY     = 2;
public static final int WEDNESDAY   = 3;
public static final int THURSDAY    = 4;
public static final int FRIDAY      = 5;
public static final int SATURDAY    = 6;
```

and

```
public static final String SUNDAY_STRING      = "Sunday";
public static final String MONDAY_STRING      = "Monday";
public static final String TUESDAY_STRING     = "Tuesday";
public static final String WEDNESDAY_STRING   = "Wednesday";
public static final String THURSDAY_STRING    = "Thursday";
public static final String FRIDAY_STRING      = "Friday";
public static final String SATURDAY_STRING    = "Saturday";
```

(i.e., no magic numbers or magic Strings in your code)

Write **three** constructors for this class.

Constructor 1

The first constructor will expect an integer representing the day of the week to store in dayNumber (e.g. 5): **public DayDisplay(int theDayNumber)**. Choose a descriptive name for the parameter but remember it cannot be exactly the same as the field name (unless you use this). Use the parameter to initialize the dayNumber instance variable, but only if is an integer between 0 and 6; otherwise set the day to 0 and print an error message. NOTE: consider carefully how you can store the "theDayNumber" parameter (which is an int!) into dayNumber (which is a NumberDisplay!)... try something like

dayNumber = new NumberDisplay(NUMBER_OF_DAYS_IN_WEEK)

and **dayNumber.setValue(theDayNumber);**

Note: this calls the NumberDisplay constructor and constructs the object.

Constructor 2

The second constructor takes a String parameter for the day of the week: **public DayDisplay(String nameOfTheDay)**. For example, if "Friday" is passed in, store 5 in dayNumber. Note: the parameter is case-insensitive. That means FRIDaY or frIDAY or FRIDAY or friday or Friday, etc., must all count as "Friday". Use the String class's .equalsIgnoreCase() method as follows:

```
if(nameOfTheDay!= null){
    dayNumber = new NumberDisplay(NUMBER_OF_DAYS_IN_WEEK);

    if(nameOfTheDay.equalsIgnoreCase(SUNDAY_STRING)){
        dayNumber.setValue(SUNDAY);
    } else if(nameOfTheDay.equalsIgnoreCase(MONDAY_STRING)){
            dayNumber.setValue(MONDAY);
    } else if(nameOfTheDay.equalsIgnoreCase(TUESDAY_STRING)){
            dayNumber.setValue(TUESDAY);
    } else if(nameOfTheDay.equalsIgnoreCase(WEDNESDAY_STRING)){
```

```
                dayNumber.setValue(WEDNESDAY);
        } else if(nameOfTheDay.equalsIgnoreCase(THURSDAY_STRING)){
                dayNumber.setValue(THURSDAY);
        } else if(nameOfTheDay.equalsIgnoreCase(FRIDAY_STRING)){
                dayNumber.setValue(FRIDAY);
        } else if(nameOfTheDay.equalsIgnoreCase(SATURDAY_STRING)){
                dayNumber.setValue(SATURDAY);
        }else{
                throw new IllegalArgumentException("invalid day name");
        }
        throw new IllegalArgumentException ("day cannot be empty");
}
```

Note: never use == to compare Strings. And never use the .equals() or .equalsIgnoreCase() methods until you are sure the String is not null. If the user passes in a non-day String (e.g. "Birthday", or "hamburger", or null, or "hi mom", etc...), store 0 into the dayNumber instance variable instead and print an error message.

Constructor 3

The third constructor is **public DayDisplay(NumberDisplay theDay)** and simply assigns the parameter to the dayNumber instance variable; nothing else.

HOW TO TEST THIS THIRD CONSTRUCTOR IN BLUEJ:

First, in blueJ, create a NumberDisplay object. Leave the little red box in the bottom of the blueJ screen. Then, in blueJ, create a DayDisplay object using this third constructor (right click the DayDisplay class, choose "new DayDisplay(NumberDisplay day), then for the parameter, click the NumberDisplay red box).

Include a Javadoc comment with @param tags for the parameter above these constructors.

The DayDisplay class must have the following methods (**note: use the theDay NumberDisplay object's getValue() method**):

```
public String getDayOfTheWeek();          // e.g. returns "Friday" if dayNumber is 5
public void incrementDay();                // calls dayNumber's increment() method
public String getWhatDayIsTomorrow(); // e.g. returns "Saturday" if dayNumber is 5
```

Also add the following method: **public void printDetails():**

Which prints in the following format, exactly:

Today is day 5 which is Friday!

or

Today is day 0 which is Sunday!

etc....

Test your project by compiling your class and then creating a DayDisplay object. Use the BlueJ inspector to check the contents of the field (right-click the red-box object at the bottom of the screen, in the object bench). Call each method to be sure it is doing what you expect.

Demonstrate your completed project to your instructor. When your instructor is satisfied, your paper will be signed and you can go home. Lab 6b (below) is due at the next lesson, and there is also a quiz next day.

Checked by: _ _____

NOTE: keep this paper for your instructor to verify your grades later in the course.

NOTE: EVERY SINGLE STUDENT MUST SUBMIT THIS LAB AND GET HIS OR HER PAPER SIGNED....

Lab 6b: to be done at home alone – submit to D2L (15 marks)

For this lab you will continue working with the NumberDisplay class and the ClockDisplay class, both available on D2L under Week 6 course content.

Note: Do not change the NumberDisplay class at all!!! Change only the ClockDisplay class's updateDisplay() method (see details below).

Add some functionality into its updateDisplay() method as described below, and then create a new HistoricalMoment class (see below).

Add the constants below (and perhaps more, if you want) into the ClockDisplay class. There must be no "magic numbers" in your code.

```
public static final int FIRST_MORNING_HOUR      = 0;
public static final int LAST_MORNING_HOUR       = 11;
public static final int FIRST_EVENING_HOUR      = 12;
public static final int LAST_EVENING_HOUR       = 23;
public static final int MINUTES_PER_HOUR        = 60;
public static final String MORNING_SUFFIX       = "a.m.";
public static final String EVENING_SUFFIX       = "p.m.";
public static final int MIDNIGHT_HOUR           = 0;
public static final int HOURS_PER_DAY           = 0;
```

Change the ClockDisplay's updateDisplay() method so it no longer displays the time in 24-hour format. The time is still STORED as 24 hours (i.e., do not change or add any new instance variables), but the time is now DISPLAYED as follows:

5:30 a.m.

12:00 p.m. (noon) 12:01 p.m. // literally; put the word (noon) like that too

12:00 a.m. (midnight) // literally; put the word (midnight) like that too

1:05 p.m.

etc....

Furthermore, create a **HistoricalMoment** class which has two instance variables:
String eventName,

and

ClockDisplay timeOfEvent.

Write **three** constructors for this class. The first is a default constructor:

public HistoricalMoment()

which creates an event named "untitled event" and creates a ClockDisplay for the timeOfEvent instance variable and set its time to midnight by default:

```
public HistoricalMoment()
{
    eventName = "untitled event";
    timeOfEvent = new ClockDisplay(MIDNIGHT_HOUR, MINUTE_ZERO);
}
```

The second constructor will expect just the String representing the name of the event:

public HistoricalMoment(String nameOfTheEvent)

Note: if null is passed in, or empty string "", store "untitled event" as the name of the event; the constructor will create a ClockDisplay object for the timeOfEvent instance variable and set its time to midnight by default:

```
public HistoricalMoment(String nameOfTheEvent)
{
  if ( (nameOfTheEvent == null) || (nameOfTheEvent.equals("")) )
  {
    eventName = "untitled event";
    timeOfEvent = new ClockDisplay(MIDNIGHT_HOUR, MINUTE_ZERO);
  }
  else
  {
    eventName = nameOfTheEvent;
    timeOfEvent = new ClockDisplay(MIDNIGHT_HOUR, MINUTE_ZERO);
  }
}
```

The third constructor:

```
public HistoricalMoment(String nameOfTheEvent, ClockDisplay theTime)
```

...takes a String parameter for the name of the event and also a ClockDisplay object for the time of the event.

N<u>ote</u>: if null is passed in for the name of the event, or empty string "", store "untitled event" as the name of the event and print an error message.

<u>**Note:**</u> if null is passed in for the time of the event, the constructor will create a ClockDisplay object for the timeOfEvent instance variable and set its time to midnight:

```
public HistoricalMoment(String nameOfTheEvent, ClockDisplay theTime)
{
  if ( (nameOfTheEvent == null) || (nameOfTheEvent.equals("")) )
  {
    throw new IllegalArgumentException ("illegal event name");
  }
```

```
    else
    {
        eventName = nameOfTheEvent;
        timeOfEvent = theTime;
    }
}
```

Add a method called **public void addMinuteToTimeOfEvent()** which calls the timeOfEvent's timeTick() method to add one minute to the timeOfEvent.

Add a method called public void printDetails() which calls the ClockDisplay instance variable's getTime() method and which prints details in this format:

At 11:02 a.m., March 2013 Equinox!

or

At 11:40 p.m., Titanic hit an iceberg!

or

At 12:00 a.m. (midnight), untitled event!

Specifically, create objects with the following initial values and show your instructor that you can print their details and add one minute to the time of the event:

March 2013 Equinox	11:02am	
Titanic hit an iceberg	11:40pm	
Halifax Explosion	9:04am	
untitled event	12:00 am	(e.g. call default constructor)

This take-home lab is due next class. Finish it before next class (on your own...no partner) and bring it in person so your instructor can review it with you during the lab period after the lecture. Do not upload your lab to BCIT's server.

Submission (15 marks)

Test your code and submit it to the Lab 6 folder on D2L (Activities -> Assignments -> Lab6) by **Thursday, Oct. 25, 2018 at midnight**. The submission must include the **ClockDisplay.java** file from your BlueJ project.

Mark Breakdown:

- 0 points if the ClockDisplay class does not compile
- (3 points) Valid JavaDoc (class and constructor), good names, data types, constants (i.e., no magic numbers)
  - 1 point taken off for each violation
- (12 points) Meets the specifications in the description and returns correct results for each method
  - Your class will be evaluated against a set of tests to make sure the methods return the correct results for a given set of inputs