ISM3230 **Individual Assignment 5 – Classes** **Spring 2017**

In this assignment you will generate a printout of departure times for all bus stops for a bus schedule. A schedule has several departure times (rows), each row consists of several stops in an ordered sequence. These stops have names (for example, A through E) and the bus takes some minutes of travel between the adjacent stops (for example, 7 minutes between A and B, 2 minutes between B and C, etc.) The schedule is a table with bus stop names listed horizontally and different departure start times listed vertically. See an example of a five-stop route below, with buses leaving from stop A at 06:00 and every 10 minutes afterwards.

```
run:
      7       2       4       9
  A  ---  B  ---  C  ---  D  ---  E
06:00   06:07   06:09   06:13   06:22
06:10   06:17   06:19   06:23   06:32
06:20   06:27   06:29   06:33   06:42
06:30   06:37   06:39   06:43   06:52
06:40   06:47   06:49   06:53   07:02
06:50   06:57   06:59   07:03   07:12
BUILD SUCCESSFUL (total time: 0 seconds)
```

Requirements:

Use classes to break down this complex problem into manageable pieces. The idea is to create a hierarchy of classes that cooperate. A BusStop object has a name and duration to the next stop in minutes. The Schedule object keeps an array of bus stops and has the ability to print out the header information (duration of travel and bus stop names), the individual schedule rows given the starting departure time and the bus stop information, and all rows of the schedule given the overall starting time and time increment between rows. The time manipulation is done with Clock objects that can advance their times and display the String representation of the times they keep. The main driver program sets up some initial parameters for the schedule, such as the number of stops, the overall starting time for the schedule and the time increment for the row departure times. It initializes the Schedule object, instructs it to print the header info and calls its method to print out the rest of the schedule.

Normally, the bus stop names and duration of travel would be passed to schedule as additional parameters for the constructor. For this assignment, we will initialize the bus stop names to letters starting with 'A', and assign random durations to travel times that will range from 1 to 9 minutes by using the Random class.

The solution should use the `Clock` class that was created in the Lab9 to format the time for display, and to advance the clock to generate the different times between stops. These instructions will not cover the `Clock` class, refer to Lab9 for details. In addition to Lab9, the `Clock` class should have public getters for minutes and hours.

The overall approach is: Create a Clock class (reused from previous labs), create a BusStop class to represent the schedule columns, create the Schedule class to handle initialization of bus stops, the

printout of the header information and the printout of the actual scheduled times, and finally the main driver program that initializes a schedule object and calls its methods to do the printing. It should display 6 different departure lines starting at 06:00 in 15 minute intervals for 5 bus stops.

Instructions:

1) Create a new Project. Add a new class file for `Clock`. Copy the code from previous labs. Add getters for minutes and hours.
2) Add a `BusStop` class:
   a) attribute: private `char stopName`
   b) attribute: private `int durationToNext`
   c) getters and setters for both attributes
3) Add a `Schedule` class:
   a) Attribute: private int `routeLength` will hold the number of stops
   b) Attribute: private BusStop[] array called `stops` will hold the bus stops
   c) Attribute: private Clock called `startingTime` will hold the overall start time for the schedule (time at first stop at first departure row)
   d) Attribute: private int `deptIncrementMinutes` will hold the number of minutes between departure rows, i.e., how many minutes between two departures from the same stop
   e) Attribute: private int `numberOfDepartures` will hold the number of rows for the schedule
   f) A constructor that takes these parameters and:
      i) initializes the attributes: routeLength, startingTime, deptIncrementMinutes, numberOfDepartures
      ii) Creates the `stop` array to the required length and in a loop:
         (1) instantiates the bus stop objects
         (2) sets the bus stop names to characters starting with 'A' for `0th` element. You can use an expression
         ```
         (char)( 'A'+i)
         ```
         to assign it to the array element, where `i` is your counter variable starting at zero. A character can be treated like an integer (you get its ASCII value), so you can add the counter to it and get a higher value. Then you must cast it back to a character to get the actual character from the new ASCII value.
         (3) sets the `durationToNext` attribute to a random integer from the range 1 to 9, inclusive. Use the `Random` class's `nextInt(int n)` method. For example, `nextInt(9)` returns numbers from 0 to 8. Adjust this so you get 1 to 9.
      iii) **Hint**: you must create the `stops` array and also the individual bus stop objects in that array with `new`
   g) Method `public void printHeader()`: This method prints the table header, that is, the duration between stops, and the stop names on the next line. It should leave the printing position on a new line. See the output for layout.
      i) Assume that the table entries for the times are going to be 5 characters long with 3 characters separating the columns.

ii) Position the minutes appropriately from the start of the screen (6 spaces from the start).
iii) Use a `for` loop to print all the duration minutes. Use 7 spaces between the them for the column separations.
iv) Position the stop names appropriately from the start of the screen (2 spaces).
v) Use a `for` loop to print all the stop names. Use "space space dash dash dash space space" as the column separator. Do not print a separator after the last stop name.
vi) Leave the printing cursor on the next line by using `println()`.

h) Method `public void printRow (Clock departureTime)` : This method will print one line of the schedule. You will pass it a parameter - a clock object that will hold the first departure time for that row. This way the method can use the clock object's methods to do the work of formatting the output and advancing the time between stops.
   i) Use a `for` loop to:
      (1) print horizontally the current time of the departure clock object that was passed to this method
      (2) print the separator which is 3 spaces, but only if not at the last stop
      (3) advance the clock's time by the appropriate duration
   ii) Leave the printing cursor on the next line by using `println()` so this method can be called repeatedly and always starts on the current line and ends at the beginning of the next one

i) Method `public void printTimes ()` : This method will call `printRow()` in a loop for `numberOfDepartures` times. In that loop it must create a new clock object based on the `startingTime` clock and pass it to the `printRow()` method. It then advances the `startingTime` clock by the `deptIncrementMinutes` amount so that a new departure time clock for a new row can be made.

4) Driver program: All of the work is done in the `Schedule` class and the `Clock` class. All you have to do here is to instantiate the `Schedule` object with some initial parameters (5 stops, 6:00 starting time, 6 departures in 15 minute increments) and to call its methods to print the header and the schedule table.
   a) Instantiate the new schedule object with the appropriate parameters.
   b) Call the `printHeader()` to get the table header.
   c) Call the `printTimes()` method with to print out the schedule.

SAMPLE OUTPUT

Note: These results have randomly generated durations, your results will vary.

```
run:
      7       2       1       3
  A  ---  B  ---  C  ---  D  ---  E
06:00   06:07   06:09   06:10   06:13
06:15   06:22   06:24   06:25   06:28
06:30   06:37   06:39   06:40   06:43
06:45   06:52   06:54   06:55   06:58
07:00   07:07   07:09   07:10   07:13
07:15   07:22   07:24   07:25   07:28
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
run:
      2       1       2       2
  A  ---  B  ---  C  ---  D  ---  E
06:00   06:02   06:03   06:05   06:07
06:15   06:17   06:18   06:20   06:22
06:30   06:32   06:33   06:35   06:37
06:45   06:47   06:48   06:50   06:52
07:00   07:02   07:03   07:05   07:07
07:15   07:17   07:18   07:20   07:22
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
run:
      8       4       1       6
  A  ---  B  ---  C  ---  D  ---  E
06:00   06:08   06:12   06:13   06:19
06:15   06:23   06:27   06:28   06:34
06:30   06:38   06:42   06:43   06:49
06:45   06:53   06:57   06:58   07:04
07:00   07:08   07:12   07:13   07:19
07:15   07:23   07:27   07:28   07:34
BUILD SUCCESSFUL (total time: 0 seconds)
```

Extra easy things to do once it works:

You can change the number of stops to 10:

```
run:
      8       1       2       9       8       9       1       4       5
  A  ---  B  ---  C  ---  D  ---  E  ---  F  ---  G  ---  H  ---  I  ---  J
06:00   06:08   06:09   06:11   06:20   06:28   06:37   06:38   06:42   06:47
06:15   06:23   06:24   06:26   06:35   06:43   06:52   06:53   06:57   07:02
06:30   06:38   06:39   06:41   06:50   06:58   07:07   07:08   07:12   07:17
06:45   06:53   06:54   06:56   07:05   07:13   07:22   07:23   07:27   07:32
07:00   07:08   07:09   07:11   07:20   07:28   07:37   07:38   07:42   07:47
07:15   07:23   07:24   07:26   07:35   07:43   07:52   07:53   07:57   08:02
BUILD SUCCESSFUL (total time: 0 seconds)
```

And change the departure times to increment by 6 minutes while having 10 departures:

```
run:
      5       1       4       9       7       8       2       1       8
  A  ---  B  ---  C  ---  D  ---  E  ---  F  ---  G  ---  H  ---  I  ---  J
06:00   06:05   06:06   06:10   06:19   06:26   06:34   06:36   06:37   06:45
06:06   06:11   06:12   06:16   06:25   06:32   06:40   06:42   06:43   06:51
06:12   06:17   06:18   06:22   06:31   06:38   06:46   06:48   06:49   06:57
06:18   06:23   06:24   06:28   06:37   06:44   06:52   06:54   06:55   07:03
06:24   06:29   06:30   06:34   06:43   06:50   06:58   07:00   07:01   07:09
06:30   06:35   06:36   06:40   06:49   06:56   07:04   07:06   07:07   07:15
06:36   06:41   06:42   06:46   06:55   07:02   07:10   07:12   07:13   07:21
06:42   06:47   06:48   06:52   07:01   07:08   07:16   07:18   07:19   07:27
06:48   06:53   06:54   06:58   07:07   07:14   07:22   07:24   07:25   07:33
06:54   06:59   07:00   07:04   07:13   07:20   07:28   07:30   07:31   07:39
BUILD SUCCESSFUL (total time: 0 seconds)
```

And now change the starting time to 22:45 and everything rolls over midnight as it should thanks to the Clock's advancetime():

```
run:
      6       6       4       9       9       1       7       1       7
  A  ---  B  ---  C  ---  D  ---  E  ---  F  ---  G  ---  H  ---  I  ---  J
22:45   22:51   22:57   23:01   23:10   23:19   23:20   23:27   23:28   23:35
22:51   22:57   23:03   23:07   23:16   23:25   23:26   23:33   23:34   23:41
22:57   23:03   23:09   23:13   23:22   23:31   23:32   23:39   23:40   23:47
23:03   23:09   23:15   23:19   23:28   23:37   23:38   23:45   23:46   23:53
23:09   23:15   23:21   23:25   23:34   23:43   23:44   23:51   23:52   23:59
23:15   23:21   23:27   23:31   23:40   23:49   23:50   23:57   23:58   00:05
23:21   23:27   23:33   23:37   23:46   23:55   23:56   00:03   00:04   00:11
23:27   23:33   23:39   23:43   23:52   00:01   00:02   00:09   00:10   00:17
23:33   23:39   23:45   23:49   23:58   00:07   00:08   00:15   00:16   00:23
23:39   23:45   23:51   23:55   00:04   00:13   00:14   00:21   00:22   00:29
BUILD SUCCESSFUL (total time: 0 seconds)
```