OOP LAB 7: Collection Class and Nested Class

Vinayak, Divya, Atharv

15th Oct 2019

General Instructions:

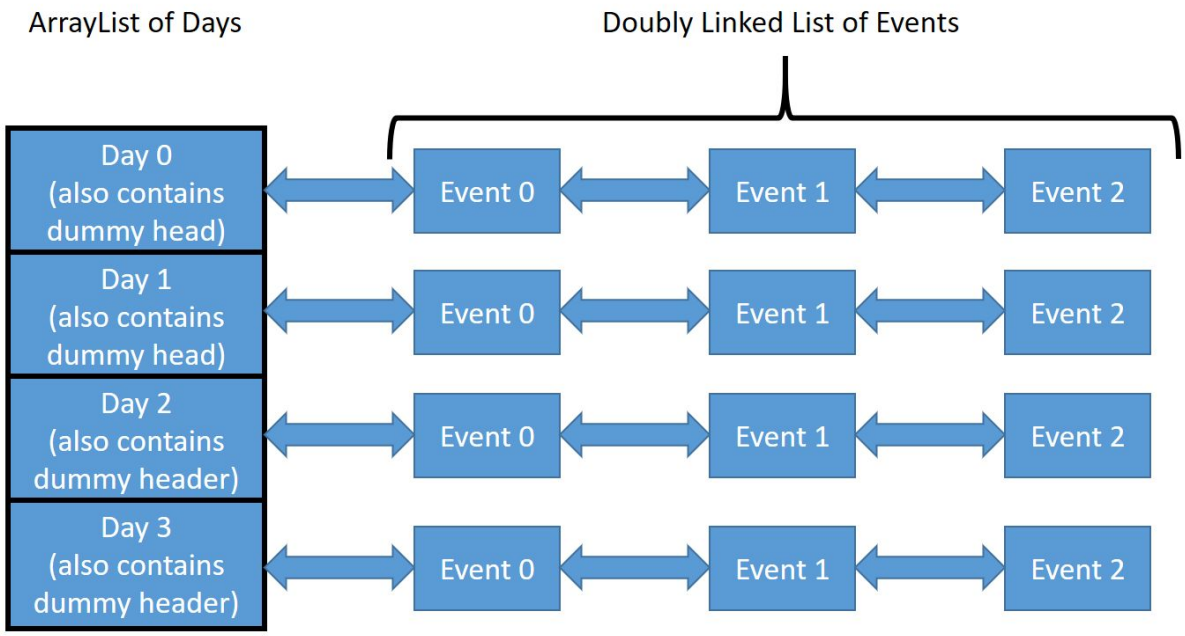
1. **Read the question carefully.**
2. Indent your code to make it readable and easier to debug

Since we have already finished creating the Clubs and Department of our very own BITS Pilani, Chandigarh Campus, it is time to host **WAVES '19**. Since there is very little time left and all the coordinators are already stressed out with the preparation so GuptaJi has asked us to ease the work of allotments by helping them keep the track of events happening in real-time and most importantly taking into account the BST.

In this lab, we will be focusing on the concept of collection and nested classes and for this, you will be required to create just 2 class files (^^).

1. **Waves** will be the actual class that will constitute of all the Days.
2. **Day** will keep track of all the events and the order in which they occur also keeping into consideration the BST as the day progresses.

Visually, it will look something like this:



Event 0 represents the event at eventIndex=0.

Files and Class Structures:

Class structure for Day.java: (It has three classes in nested fashion)

```
class Day { ...
        class Node {...
                class Event {...
                }
        }
}
```

Class structure for Waves.java (It has one class, with no nested classes here)

```
Class Waves {...
}
```

You will have to implement various methods in each of these classes and their respective nested classes as per the JavaDoc in order to successfully simulate the fest.

Marking Scheme:

Function Name	**Class in which it resides**	**Marks Allotted**
addBst()	Event	1
Constructor for class Day	Day	1
addEvent()	Day	1
modifyTime()	Day	1
addDay()	Waves	1
deleteDay()	Waves	1
addEvent()	Waves	1
deleteEvent()	Waves	1
clash()	Waves	1
cascadeBST()	Waves	1

Pointers to keep in mind:

1. Make sure to import **java.io.*** and **java.util.***

2. Run it by creating a folder java and put both Waves.java and Day.java file. When you want to run it, type the following terminal in command:
   java -jar test.jar (Remember to put test.jar in the same folder as 'java')
   Please type this out (don't copy it from the PDF).

3. We suggest you to solve it in the following order:
   Event -> Node -> Day -> Waves

4. The ordering of events is based on start time only (not a combination of start and end times).
   Example 1:
   Existing events – (0,4) (5,8) (10,11). Event to be added – (8,9)
   New events DLL - (0,4) (5,8) (8,9) (10,11)
   Example 2:
   Existing events – (0,4) (5,8) (10,11). Event to be added – (5,6)
   New events DLL - (0,4) (5,8)(5,6) (10,11)
   Example 3:
   Existing events - (0,4) (5,8) (10,11). Event to be added – (12,14)
   New events DLL - (0,4) (5,8) (10,11) (12,14)

All the times will be positive integers(and contain only the hour). The time would not exceed 24 in any case.

5. The first node in the DLL is a dummy node. So make sure you make the constructor of Day properly.
6. You might not get marks for some of the other methods if your constructors are not correct
7. BST here refers to the Bits Standard Time and not Binary Search Tree.