Programming and Data Structures
Programming Project 1

Objectives of the assignment

Students should demonstrate the following abilities:

1. Implement classes and class relationships
2. Create Exception classes that inherit Java class exceptions
3. Use Java Exception Handling mechanisms to handle exceptions
4. Access text files for reading and writing to upload and save data in Java programs

---

The assignment consists in creating a program named "`CalendarManager`" to help the user manage events in a calendar. Follow the steps listed below to create your program.

1. Create a class **`Date`** that has three data members of type integer: day, month, and year. Create two constructors, accessors and mutators for the class **`Date`**. Override the method **`toString()`** from class **`Object`**. **`toString()`** should return the date in the format **`mm/dd/yyyy`**. Make the class **`Date`** implement the interface **`Comparable<E>`** and write the definition for **`compareTo()`**. **`compareTo()`** should order two dates based on the year first, then the month, and finally the day.

2. Create a class **`Time`** that has two data members of type integer: hours and minutes. Create two constructors, accessors and mutators for the class **`Time`**. Override the method **`toString()`** from class **`Object`** to return the time in the format **`hh:mm`**. Make the class **`Time`** implement the interface **`Comparable<E>`** and write the definition for **`compareTo()`**. **`compareTo()`** should order two times based on the hours and minutes.

3. Create an abstract class **`Event`** that has four data members: **`description`** of type `String`, **`date`** of type `Date`, **`time`** of type `Time`, and **`location`** of type `String`. Create two constructors, accessors and mutators for the class **`Event`**. Override the methods **`toString()`** and **`equals()`** from class **`Object`**. **`toString()`** should return a formatted string that contains the four attributes of an event. **`equals()`** should

return true if two events have the same description. The class should implement the interface **`Comparable<E>`**. Define the method **`compareTo()`** to compare two events based on their dates and times (use **`compareTo()`** from classes **`Date`** and **`Time`**).

4. Create a class **`Appointment`** that extends the class **`Event`** and has one additional data member **`contact`** of type **`String`** to hold the name of the person to contact about the appointment. Create two constructors for the class, accessor and mutator methods. Override the method **`toString()`** to return a formatted string that contains all the attributes of an appointment.

5. Create a class **`Meeting`** that extends the class **`Event`** and has one additional data member **`host`** of type **`String`** to hold the name of the host for the meeting. Create two constructors for the class, accessor and mutator methods. Override the method **`toString()`** to return a formatted string that contains all the attributes of a meeting.

6. Create a class **`InvalidDateException`** that inherits the class **`Exception`**. The class must have two constructors: one with no parameters and one with one **`String`** parameter **`message`**. The default constructor should call the super constructor with the argument "**`Invalid Date.`**", and the constructor with one parameter should call the super constructor with the argument **`message`**.

7. Create a class **`InvalidTimeException`** that inherits the class **`Exception`**. The class must have two constructors: one with no parameters and one with one **`String`** parameter **`message`**. The default constructor should call the super constructor with the argument "**`Invalid Time.`**", and the constructor with one parameter should call the super constructor with the argument **`message`**.

8. Create a class **`Calendar`** that has two data members: an array **`eventList`** of type **`Event`** and an integer **`count`** for the number of used elements in the array **`eventList`**. Create two constructors for the class **`Calendar`**. The default constructor creates the array **`eventList`** with size 100 and initializes **`count`** to 0. The second constructor

accepts a parameter of type `String`, **`filename`**, and creates the array **`eventList`** with size 100, reads the events from the text file **`filename`**, stores the information read from the file as **`Event`** objects in the array **`eventList`**, and sets the variable **`count`** to the number of events read from the file. In addition to the two constructors, class Calendar should have at least the following methods:

a. **`Event findEvent(String d)`**: searches in the array **`eventList`** for the event that has the same description as the parameter **`d`**. The method returns the reference to the **`Event`** object if found, or **`null`** otherwise.
b. **`boolean addEvent(Event e)`**: adds **`e`** at the end of the array **`eventList`** if the array is not full and increments the variable **`count`**. The method returns **`true`** if the operation is successful, or **`false`** if the array is full.
c. **`boolean removeEvent(Event e)`**: removes **`e`** from the array **`eventList`** if **`e`** is found and decrements the variable **`count`**. The method returns **`true`** if the operation is successful, or **`false`** if the event is not found.
d. **`void viewAllEvent()`**: prints the list of events in **`eventList`** in a tabular format (see sample run).
e. **`void sortEvent()`**: orders the events in **`eventList`** based on their date and time using the method **`java.util.Arrays.sort()`**.
f. **`void saveToFile(String filename)`**: saves the events from the array **`eventList`** to the text file **`filename`**. Make sure the attributes of the events are written to the file such that each attribute is on a separate line (see the format of the provided file **`events.txt`**).

9. Create a class **`CalendarManager`** with a **`main`** method to do the following:

a. Create an instance of class **`Calendar`** using the list of events provided in the file **`events.txt`**. Name the instance **`myCalendar`**.

b. Prompt the user to select one of the following operations:

i. Find an event with a given description

ii. *Add a new event*
iii. *Remove an existing event*
iv. *View all events*
v. *Sort events*
vi. *quit* the program.

c. When the user wants to add an event, prompt the user to enter the attributes of the event (date, time, location, description, and contact/host) depending on the type of event they want to add. Your program must check the validity of the data entered for date and time. If the date entered by the user is not in the format **"mm/dd/year"**, throw an **InvalidDateException** with the message "**Invalid Date Format**". If **dd** is not in the range 1-31, throw an **InvalidDateExcpetion** with the message "**Invalid day number – must be from 1 to 31.**". If **mm** is not in the range 1-12, throw an **InvalidDateException** with the message "**Invalid month number – must be from 1 to 12.**". Finally, if **year** is less than 1972 or greater than 2030, your program should throw an **InvalidDateException** object with the message "**Invalid year – must be between 1972 to 2030.**". If the time entered by the user is not in the format **"hh:mm"**, throw an **InvalidTimeExceptioon** with the message "**Invalid time format – must be hh:mm.**". If **hh** is not in the range 0-23 or the **mm** field is not in the range 0-59, your program should throw an **InvalidTimeException** object with the message "**Invalid hours – must be from 0 to 23.**", or "**Invalid minutes – must be from 0 to 59.**" respectively. You should also include catch blocks for the two types of exception. The catch blocks should only display the message of the exception argument. For all exceptions, go back to the main menu to select an operation after displaying the error message.

d. When the user chooses to quit the program, your program should save all the events in the calendar to the text file "**events.txt**".

10. Test your program for all user selected operations and exceptions.

11. Draw the UML diagrams for the following classes: **`Exception`**, **`InvalidTimeExceptio`**n, **`InvalidDateException`**, **`Date`**, **`Time`**, **`Event`**, **`Appointment`**, **`Meeting`**, **`Calendar`**, and **`CalendarManager`**. Show all the relationships between the classes and label each relationship with one of the following labels: "**`is`**", "**`has`**", or "**`uses`**". Submit the UML diagram as a pdf file named **`UML.pdf`**.

12. Submit the following java files on Github: **`InvalidTimeException.java`**, **`InvalidDateException.java`**, **`Date.java`**, **`Time.java`**, **`Event.java`**, **`Appointment.java`**, **`Meeting.java`**, **`Calendar.java`**, and **`CalendarManager.java`**. Do not forget to include appropriate Javadoc comments in your code.

Important Note: The text file **`events.txt`** contains six (6) lines for each event with one attribute per line. The order of the attributes is as follows:

`Type of the event (meeting or appointment)`
`Date`
`Time`
`Location`
`Description`
`Host or contact`

-------- Sample run 1 (find/add/remove/view/sort/save)---------

```
Select an operation:
1: Find an event
2: Add a new event
3: Remove an event
4: View all events
5: Sort events
6: Exit
4
Type           Date          Time   Location              Description         Host/Contact
Meeting        01/22/2021    14:15  Zoom                  Office hours        Houria Oudghiri
Meeting        01/22/2021    18:45  PA-115                department meeting  George Lester
Appointment    02/22/2021    11:10  Lehigh Valley Clinic  Doctor              Kathy Brown
Meeting        03/16/2020    09:20  Neville Hall          CSE017 class        Robert Nealy
Appointment    05/12/2019    10:30  Wells Fargo           Bank                Sarah Blank
Appointment    02/10/2021    15:30  Bethlehem             Dentist             Lisa Zuppe

Select an operation:
1: Find an event
2: Add a new event
3: Remove an event
4: View all events
5: Sort events
6: Exit
1
Enter the description of the event:
Doctor
Event found:
Appointment    02/22/2021    11:10  Lehigh Valley Clinic  Doctor              Kathy Brown

Select an operation:
1: Find an event
2: Add a new event
3: Remove an event
4: View all events
5: Sort events
6: Exit
2
Enter the type of the event(meeting/appointment):
meeting
Enter the date of the event(mm/dd/yyyy):
09/22/2020
Enter the time of the event(hh:mm):
10:45
Enter the location of the event:
PA-114
Enter the description of the event:
Programming Club
Enter the contact of the event:
Enter the host of the event:
Jack Buster

Select an operation:
1: Find an event
2: Add a new event
3: Remove an event
4: View all events
5: Sort events
6: Exit
4
Type           Date          Time   Location              Description         Host/Contact
Meeting        01/22/2021    14:15  Zoom                  Office hours        Houria Oudghiri
Meeting        01/22/2021    18:45  PA-115                department meeting  George Lester
Appointment    02/22/2021    11:10  Lehigh Valley Clinic  Doctor              Kathy Brown
Meeting        03/16/2020    09:20  Neville Hall          CSE017 class        Robert Nealy
Appointment    05/12/2019    10:30  Wells Fargo           Bank                Sarah Blank
Appointment    02/10/2021    15:30  Bethlehem             Dentist             Lisa Zuppe
Meeting        09/22/2020    10:45  PA-114                Programming Club    Jack Buster
```

```
Select an operation:
1: Find an event
2: Add a new event
3: Remove an event
4: View all events
5: Sort events
6: Exit
3
Enter the description of the event:
Dentist
Event found:
Appointment    02/10/2021     15:30   Bethlehem             Dentist             Lisa Zuppe

Select an operation:
1: Find an event
2: Add a new event
3: Remove an event
4: View all events
5: Sort events
6: Exit
4
Type           Date           Time    Location              Description         Host/Contact
Meeting        01/22/2021     14:15   Zoom                  Office hours        Houria Oudghiri
Meeting        01/22/2021     18:45   PA-115                department meeting  George Lester
Appointment    02/22/2021     11:10   Lehigh Valley Clinic  Doctor              Kathy Brown
Meeting        03/16/2020     09:20   Neville Hall          CSE017 class        Robert Nealy
Appointment    05/12/2019     10:30   Wells Fargo           Bank                Sarah Blank
Meeting        09/22/2020     10:45   PA-114                Programming Club    Jack Buster

Select an operation:
1: Find an event
2: Add a new event
3: Remove an event
4: View all events
5: Sort events
6: Exit
5

Select an operation:
1: Find an event
2: Add a new event
3: Remove an event
4: View all events
5: Sort events
6: Exit
4
Type           Date           Time    Location              Description         Host/Contact
Appointment    05/12/2019     10:30   Wells Fargo           Bank                Sarah Blank
Meeting        03/16/2020     09:20   Neville Hall          CSE017 class        Robert Nealy
Meeting        09/22/2020     10:45   PA-114                Programming Club    Jack Buster
Meeting        01/22/2021     14:15   Zoom                  Office hours        Houria Oudghiri
Meeting        01/22/2021     18:45   PA-115                department meeting  George Lester
Appointment    02/22/2021     11:10   Lehigh Valley Clinic  Doctor              Kathy Brown

Select an operation:
1: Find an event
2: Add a new event
3: Remove an event
4: View all events
5: Sort events
6: Exit
6
```

---------- Sample run 2 (Exception Handling) ----------

```
Select an operation:
1: Find an event
2: Add a new event
3: Remove an event
4: View all events
5: Sort events
6: Exit
4
Type           Date           Time   Location              Description         Host/Contact
Appointment    05/12/2019     10:30  Wells Fargo           Bank                Sarah Blank
Meeting        03/16/2020     09:20  Neville Hall          CSE017 class        Robert Nealy
Meeting        09/22/2020     10:45  PA-114                Programming Club    Jack Buster
Meeting        01/22/2021     14:15  Zoom                  Office hours        Houria Oudghiri
Meeting        01/22/2021     18:45  PA-115                department meeting  George Lester
Appointment    02/22/2021     11:10  Lehigh Valley Clinic  Doctor              Kathy Brown

Select an operation:
1: Find an event
2: Add a new event
3: Remove an event
4: View all events
5: Sort events
6: Exit
2
Enter the type of the event(meeting/appointment):
meeting
Enter the date of the event(mm/dd/yyyy):
13/21/2020
Invalid Date. The month must be between 1 and 12

Select an operation:
1: Find an event
2: Add a new event
3: Remove an event
4: View all events
5: Sort events
6: Exit
2
Enter the type of the event(meeting/appointment):
meeting
Enter the date of the event(mm/dd/yyyy):
11/32/2020
Invalid Date. The day must be between 1 and 31

Select an operation:
1: Find an event
2: Add a new event
3: Remove an event
4: View all events
5: Sort events
6: Exit
2
Enter the type of the event(meeting/appointment):
meeting
Enter the date of the event(mm/dd/yyyy):
10/22/2020
Enter the time of the event(hh:mm):
24:50
Invalid Time. The hours must be between 0 and 23

Select an operation:
1: Find an event
2: Add a new event
3: Remove an event
4: View all events
5: Sort events
```

```
6: Exit
2
Enter the type of the event(meeting/appointment):
meeting
Enter the date of the event(mm/dd/yyyy):
01/12/2021
Enter the time of the event(hh:mm):
12:69
Invalid Time. The minutes must be between 0 and 23

Select an operation:
1: Find an event
2: Add a new event
3: Remove an event
4: View all events
5: Sort events
6: Exit
6
```