Homework 6

CS 1323, Fall 2015

**Name (5 points): Hunter Black**

**This assignment is handed in through the dropbox on Janux by 11:59 on Wednesday, December 9.**

1. (5 points) Draw a UML diagram that describes a class that will be used to describe the current weather. The weather is described by the wind, humidity, dew point, pressure, visibility and UV index (as shown here: <http://www.weather.com/weather/today/l/USOK0388:1:US>).

|  |
| --- |
| Weather |
| * wind: int * humidity: int * dewPoint: int * pressure: double * visibility: double * uvIndex: int |
| + Weather();  + Weather(int wind, int humidity, int dewPoint, double pressure, double visibility, int uvIndex);  + getWind(): int  + getHumidity(): int  + getDewPoint(): int  + getPressure(): double  + getVisibility(): double  + getUVIndex(): int  + setWind(int wind): void  + setHumidity(int humidity): void  + setDewPoint(int dewPoint): void  + setPressure(double pressure): void  + setVisibility(double visibility): void  + setUVIndex(int uvIndex): void  + getWeatherInfo(): String |

1. (10 points) Suppose that we have the two classes below. Trace the execution of the main program in the Driver class.

public class FakeArrayList

{

private int size;

private double[] data;

public FakeArrayList(int capacity)

{

data = new double[capacity];

size = 0;

}

public int getSize()

{

return size;

}

public double get(int index)

{

return data[index];

}

public void set(int index, double value)

{

data[index] = value;

}

}

public class Driver

{

public static void main(String[] args)

{

FakeArrayList first = new FakeArrayList(3);

first.set(2, 7.1);

first.set(1, 4.9);

FakeArrayList second = new FakeArrayList(5);

second.set(0, 9.1);

second.set(1, 8.4);

}

}

main stack frame

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Identifier** | **Address** | **Contents** |
|  | first | 100 | 1000 |
|  |  | 101 |  |
|  |  | 102 |  |
|  |  | 103 |  |
|  |  | 104 |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | heap | |
|  | **Identifier** | **Address** | **Contents** |
|  | size | 1000 | 0 |
|  |  | 1001 |  |
|  |  | 1002 |  |
|  |  | 1003 |  |
|  |  | 1004 |  |
|  |  | 1005 |  |
|  |  | 1006 |  |
|  |  | 1007 |  |
|  |  | 1008 |  |
|  |  | 1009 |  |
|  |  | 1010 |  |
|  |  | 1011 |  |
|  |  | 1012 |  |
|  |  | 1013 |  |
|  |  | 1014 |  |

1. (15 points) For the problem description below, pick out three classes that you would implement. Do not include a Driver class or a main program, although that would be necessary if you were to write the whole program. Show the class names and data (only) in UML. Include aggregation relationship(s) that exist between classes. Do not implement any code.

You are writing a calendar program. The calendar stores many appointments. Appointments have a start time, and ending time, a location, and a topic. Appointment topics may not be changed, although locations and starting and ending times may be. The calendar also stores meetings. Meetings are similar to appointments, except they also have the names of other people who will be attending. A meeting can change starting and ending times and locations, just like an appointment. It also can change attendees.

|  |
| --- |
| Calendar |
| * calenderName: String * appointments: Appointment[] * meetings: Meeting[] |
| + Calender()  + Calender(String name)  + addAppointment(): void  + addMeeting(): void  + getCalenderName(): String  + getCalenderInfo(): String |

Calendar

Appointment

Calendar

Meeting

|  |
| --- |
| Appointment |
| * startTime: double * endTime: double * location: String * topic: String |
| + Appointment()  + Appointment(double startTime, double endTime, String location, String topic)  + setStartTime(double time): void  + setEndTime(double time): void  + setLocation(String location): void  + setTopic(String topic): void  + getStartTime(): double  + getEndTime(): double  + getLocation(): String  + getTopic(): String  + getAppointmentInfo(): String |

|  |
| --- |
| Meeting |
| * attendees: ArrayList<String> * startTime: double * endTime: double |
| + Meeting()  + Meeting(ArrayList<String> list, double startTime, double endTime)  + setStartTime(double time): void  + setEndTime(double time): void  + addAttendee(String name): void  + removeAttendee(String name): void  + getStartEndTime(): String  + getAttendeeList(): ArrayList<String> |

1. (15 points; 3 points each) Ordered pairs are used in mathematics. They are written in parentheses (first, second). Here is how addition works: (a, b) + (c, d) = (a+c, b+d). Scalar multiplication that works like this: c(a,b) = (ca, cb). The magnitude of (a,b) is . Don’t worry about how you would use these things if you aren’t familiar with them.

Suppose we were to write an OrderedPair class in Java. The design for the class name and data is below.

OrderedPair

-first:double

-second:double

Select the correct signature for the following methods in the OrderedPair class.

* 1. An instance method that adds two OrderedPair objects, storing the sum in the current OrderedPair.

-addPair(OrderedPair x, OrderedPair y): void

* 1. An instance method that adds two OrderedPair objects and returns the sum in a new OrderedPair object.

-addNewPair(OrderedPair x, OrderedPair y): OrderedPair

* 1. A class method that adds two OrderedPair objects and returns the sum in a new OrderedPair object.

+addNewPair(OrderedPair x, OrderedPair y): OrderedPair

* 1. An instance method that performs scalar multiplication and stores the result in the current OrderedPair

-multiplyPair(OrderedPair x, int c): void

* 1. An instance method that returns the magnitude of an OrderedPair object.

-getMagnitude(OrderedPair x): double