Solution Homework 6

CS 1323, Spring 2015

1. (5 points) Draw a UML diagram that describes an online advice column. The column has an author, a title, a publisher, a body consisting of paragraphs of text, and a date published.

Column

-author: String

-publisher: String

-body: ArrayList<String>

-published: Date

+Colunm(author: String, publisher: String, body: ArrayList<String>, published: Date)

+getAuthor(): String

+getPublisher(): String

+getBody(): ArrayList<String>

+getDatePublished(): Date

+changeParagraph(newParagraph: String, number: int): void

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

public class Email

{

private String to;

private String from;

private String subject;

private String message;

public Email(String to, String from, String subject, String message)

{

this.to = to;

this.from = from;

this.subject = subject;

this.message = message;

}

public String getTo()

{

return to;

}

public String getFrom()

{

return from;

}

public String getSubject()

{

return subject;

}

public String getMessage()

{

return message;

}

}

public class Driver

{

public static void main(String[] args)

{

Email first = new Email(“dt@ou.edu”, “mt@ou.edu”, “Mother’s Day”,

“Get gift.”);

Email second = new Email(“am@ou.edu”, “af@ou.edu”, “Copy handouts”,

“Today, please.”);

}

}

|  |  |  |
| --- | --- | --- |
|  | main stack frame | |
|  | **Address** | **Contents** |
| first | 100 |  |
| second | 101 |  |
|  | 102 |  |
|  | 103 |  |
|  | 104 |  |

|  |  |  |
| --- | --- | --- |
|  | heap | |
|  | **Address** | **Contents** |
| to | 1000 | 1008 |
| from | 1001 | 1011 |
| subject | 1002 | 1014 |
| message | 1003 | 1017 |
| to | 1004 | 1020 |
| from | 1005 | 1023 |
| subject | 1006 | 1026 |
| message | 1007 | 1030 |
|  | 1008 | dt@o |
|  | 1009 | u.ed |
|  | 1010 | u |
|  | 1011 | mt@o |
|  | 1012 | u.ed |
|  | 1013 | u |
|  | 1014 | Moth |
|  | 1015 | er’s |
|  | 1016 | Day |
|  | 1017 | Get |
|  | 1018 | gift |
|  | 1019 | . |
|  | 1020 | am@o |
|  | 1021 | u.ed |
|  | 1022 | u |
|  | 1023 | af@o |
|  | 1024 | u.ed |
|  | 1025 | u |
|  | 1026 | Copy |
|  | 1027 | han |
|  | 1028 | dout |
|  | 1029 | s. |
|  | 1030 | Toda |
|  | 1031 | y, p |
|  | 1032 | leas |
|  | 1033 | e. |

Grading note: Student may show strings in cells instead of using references. Also, if students show more than 4 letters per cell that is fine too.

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

-meetings: ArrayList<Meeting>

-appointments: ArrayList<Apopintment>

Appointment

-start: String

-end: String

-location: String

-topic: String

Meeting

-appointment: Appointment

-attendees: String[]

1. (15 points; 3 points each) Ordered pairs are used in mathematics. They are written in parentheses (x,y). 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

-x:double

-y: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.

void add(OrderedPair pair)

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

OrderedPair add(OrderedPair pair)

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

OrderedPair add(OrderedPair first, OrderedPair second)

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

void scalarMultiplication(double multiplier)

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

double magnitude()