CS401 MPP - Midterm

Al-Tarawneh

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ StudentId:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Part I**  **(14)** | **Part II.1**  **(5)** | **Part II.2**  **(4)** | **Part II.3**  **(4)** | **Part II.4**  **(5)** | **Part II.5**  **(8)** |
|  |  |  |  |  |  |

**Part I: Short Answer**

**1) Write ‘True’ or ‘False’ for the following statements: (4 points – each 1)**

|  |  |
| --- | --- |
| 1.1) To implement the class diagram below in code, a list of type B objects must be placed inside the class A. | ( T ) |
| 1.2) Class Diagram shows attributes and operations in each of the (primary) classes of the system as well as relationships between them. | ( T ) |
| 1.3) If the class diagram below has been implemented in code, it would be possible to navigate from class A to class B. | ( T ) |
| 1.4) If class ‘C’ was implementing two interfaces, both had the same method name and same signature, but no implementation in both. No compile errors will occur when class ‘C’ overrides the method that is in both interfaces with the same signature? | ( T ) |

**2) Circle the correct answer: (6 points – each 2)**

2.1) Given the following codes, which of the following is correct regarding the relationship between Employer and Gardener? Circle one answer.

|  |  |
| --- | --- |
|  |  |

1. There is a dependency from Employer to Gardener
2. There is a one-way association from Employer to Gardener
3. There is a two-way association between Employer and Gardener
4. Not possible to determine from the code shown

2.2) Given the following codes, what would it output when the main method is executed?

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |
| 1. Super   Super  Super   1. Supreme   Super  Super | | 1. Supreme   Supreme  Super   1. Supreme   Super  Supreme |

2.3) Given the following codes, which of the following UML diagrams correctly models the relationship between Guest and Reservation classes?

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | |
| A. | B. | |
| C. | D. | |

**3) Given the following codes, write the expected results in the corresponding fields for each Test class, you may write ‘compile error’ if you assume it is. (4 points – each 1)**

|  |  |
| --- | --- |
| **public class A {  private int x;  public A() {  x = 10;  }  public A(int x) {  this.x = x;  }  public void print(){  System.*out*.println("class A");  }  public int getValue() {  return x;  }  public void setValue(int x) {  this.x = x;  } }** | **public class B extends A {   @Override  public void print(){  System.*out*.println("class B");  } }** |
| **public class Test1 {  public static void main(String[] args) {  B o1 = new B();  System.*out*.println(o1.getValue());  } }** | 10 |
|  | class B |
| **public class Test3 {  public static void main(String[] args) {  A o1 = new A();  A o2 = new B();  A o3 = o1;  o3.setValue(5);  o1.print();  o2.print();  o3.print();  } }** | class A  class B  class A |
|  | // compilation error |

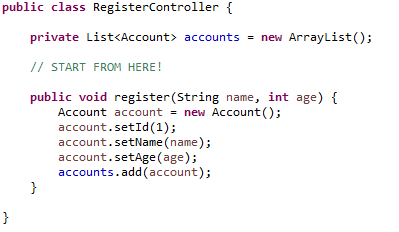
**Part II: Skill Questions**

**1) Draw a UML class diagram consisting of three classes: Library, Book, and Author. Your class diagram should reflect all the following descriptions: (5 Points)**

Your class diagram should illustrate the following description: A **Library** has a name and maintains a collection of one or more books. All the books of a library can be accessed from the library, but not the other way around. A book may belong to multiple libraries.A **Book** has a title and an ISBN, and it can be written by one or more authors. Each book can be associated with multiple authors, reflecting the collaborative nature of book writing. Books can be part of multiple libraries, indicating that a single book can be available in different libraries.  
An **Author** has a name and a list of books they have written. Each author can write multiple books, and this relationship should be reflected in the class design. The author class should provide methods to access and modify the author's name and manage the books they have written.

//Draw the diagram including associations and its multiplicities in the space below…

**2) Draw a sequence diagram for the method register() from the code below. ( 4 Points )**

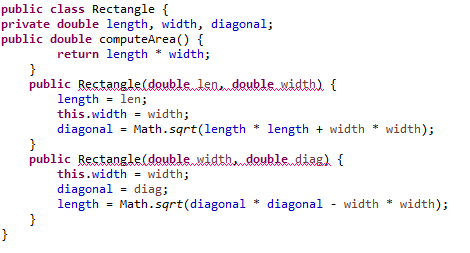


//Draw the diagram in the space below…

A diagram of a computer program

Description automatically generated

**3) A rectangle can be created by specifying its width and length, or by specifying is width and the length of its diagonal. Here is a Rectangle class that provides two constructors that will create a Rectangle object in either of these two ways.**



A) In the space provided below, rewrite the code for Rectangle (from Part A) so that it supports the two ways of constructing a rectangle. Use a technique described in the course. **( 4 points )**

public class Rectangle{

A screenshot of a computer code

Description automatically generated

**4)** A system that consists of three classes: Publisher, Library, and Books.

The **Publisher** is specified with a name, contact information, and may manage one or more libraries. Each **Library** is identified by a library number and contains one or more books. Every **Book** has a price. Each class has a method getBookPrice(), and when executed from the Publisher class, it should be delegated in a distributed manner to the Library class and then to the Books class to calculate the total price of all books managed by a specific publisher. Below is a sequence diagram for this problem. Write the code for all three classes. **( 5 points )**

Book

Library

Publisher

getBookPrice()

getBookPrice()

getBookPrice()

class Publisher {

private String name;

private String contactInfo;

private List<Library> libraries;

public Publisher(String name, String contactInfo) {

this.name = name;

this.contactInfo = contactInfo;

this.libraries = new ArrayList<>();

}

public void addLibrary(Library library) {

this.libraries.add(library);

}

public double getBookPrice() {

double totalPrice = 0;

for (Library library : libraries) {

totalPrice += library.getBookPrice();

}

return totalPrice;

}

}

class Library {

private int libraryNumber;

private List<Book> books;

public Library(int libraryNumber) {

this.libraryNumber = libraryNumber;

this.books = new ArrayList<>();

}

public void addBook(Book book) {

this.books.add(book);

}

public double getBookPrice() {

double totalPrice = 0;

for (Book book : books) {

totalPrice += book.getPrice();

}

return totalPrice;

}

}

class Book {

private double price;

public Book(double price) {

this.price = price;

}

public double getPrice() {

return price;

}

}

**5)** A channel service subscription system, you are required to write the code for the mentioned classes in the class diagram below according to the following:

The customer can subscribe to one or more services, and it is possible to remove a service as well. A method in the Customer class **totalFee()** should traverse through all subscribed services for a customer and calculate the total fee. As shown in the class diagram below, there are two types of services:

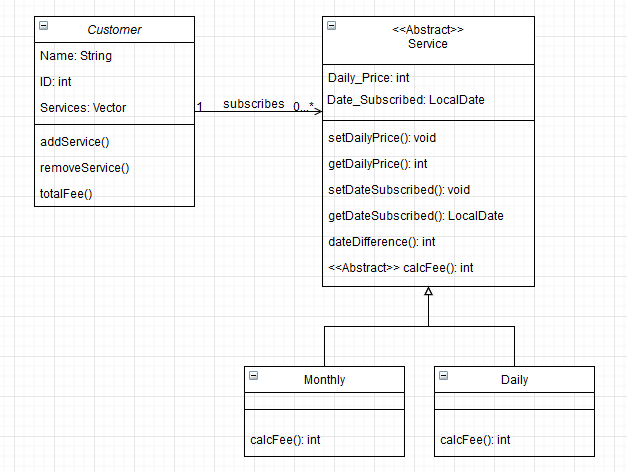
‘**Monthly**’, which calculates the fee by multiplying the **Daily\_Price** by 30.   
The method **calcFee()** in the **Monthly** class applies the following:

**Daily\_Price \* 30**

‘**Daily**’ takes the result value of **dateDifference()** method and multiplies it with the **Daily\_Price**. The method **calcFee()** in the **Daily** class applies the following:

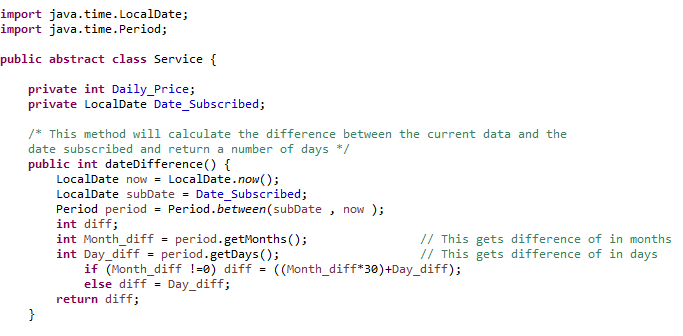
**Daily\_Price \* (result value of dateDifference( ))**

The initial code for the Service class is given which contains the implementation of the **dateDifference()** method. This method will return a number based on the difference between the date subscribed and the current date. **( 8 points )**



Publisher

Publisher



abstract class Service {

public abstract double calcFee(double dailyPrice);

}

class Monthly extends Service {

@Override

public double calcFee(double dailyPrice) {

return dailyPrice \* 30;

}

}

class Daily extends Service {

@Override

public double calcFee(double dailyPrice) {

return dailyPrice \* dateDifference();

}

}

class Customer {

private String name;

private List<Service> subscribedServices;

public Customer(String name) {

this.name = name;

this.subscribedServices = new ArrayList<>();

}

public void addService(Service service) {

subscribedServices.add(service);

}

public double totalFee() {

double total = 0;

for (Service service : subscribedServices) {

total += service.calcFee(getServicePrice(service));

}

return total;

}

// Helper method to retrieve the daily price of a service

private double getServicePrice(Service service) {

// Logic to retrieve the daily price based on the service type (Monthly or Daily)

return dailyPrice;

}

}

**Bonus Question**

It was mentioned more than once that the knowledge gained from this course exists in all programming languages. How can we relate this point to the science of creative intelligence?