B3.2

OOP for multiple classes

CHAPTER ASSIGNMENT · 108 MARKS

Name _____ Class _____ No. _____ Date _____

Answer every question in the space given. Show your working. You may use the chapter page to help you.

Questions from *Computer Science for the IB Diploma*, 1st edition (2025), Paul Baumgarten, Ioana Ganea, Carl Turland (Hachette Learning), for classroom use.

Linking Question (p. 428)

1 How can design patterns in OOP facilitate the architecture of scalable and maintainable machine learning models? (A4)

Review Questions (pp. 435–437)

1 In UML diagrams, which symbol is used to represent inheritance between two classes?

(a) A dashed line with an arrow

(b) A solid line with a hollow arrow

(c) A solid line with a filled arrow

(d) A dashed line without any arrows

2 What keyword is used in Java to inherit a class?

(a) Implements

(b) Extends

(c) Inherits

(d) Superclass

3 Which principle of OOP is primarily used to enhance code flexibility and maintainability through interfaces?

(a) Encapsulation

(b) Inheritance

(c) Polymorphism

(d) Abstraction

4 What is required for method overriding to occur in object-oriented programming?

(a) The method must have the same name and different parameters in the subclass

(b) The method must have the same name and parameter list in the subclass, and be marked with `final`

(c) The method must have the same name and parameter list in the subclass

(d) The method must have a different name but the same parameters in the subclass

5 Which statement is true about abstract classes?

(a) Abstract classes can be instantiated

(b) Abstract classes cannot have any method implementations

(c) Abstract classes can contain both abstract and implemented methods

(d) All methods in an abstract class must be abstract

6 What is the purpose of declaring a class as abstract?

(a) To force a class to provide implementations of all its methods

(b) To prevent the class from being instantiated directly

(c) To ensure that the class can only contain static methods

(d) To make the class available only to other classes in the same package

7 Which of the following best describes aggregation?

(a) A strong "has-a" relationship where the lifetime of the contained objects depends on the lifetime of the container

(b) A weak "has-a" relationship where the contained objects can exist independently of the container

(c) An "is-a" relationship between two entities

(d) None of the above

8 Which scenario is an example of composition?

(a) A library owns books

(b) A university has students

(c) An apartment building includes apartments

(d) A shopping cart contains products

9 Which design pattern ensures that a class has only one instance and provides a global point of access to it?

(a) Factory pattern

(b) Singleton pattern

(c) Observer pattern

(d) Builder pattern

10 In the observer design pattern, what is the role of the "Subject"?

(a) To notify all observers about any changes

(b) To keep track of all dependencies

(c) To update the state of various subjects

(d) To request updates from observers

11 Describe why inheritance is considered a powerful feature of OOP.

12 Discuss how polymorphism enhances software maintainability.

13. Discuss two benefits that abstract classes provide over using normal inheritance.

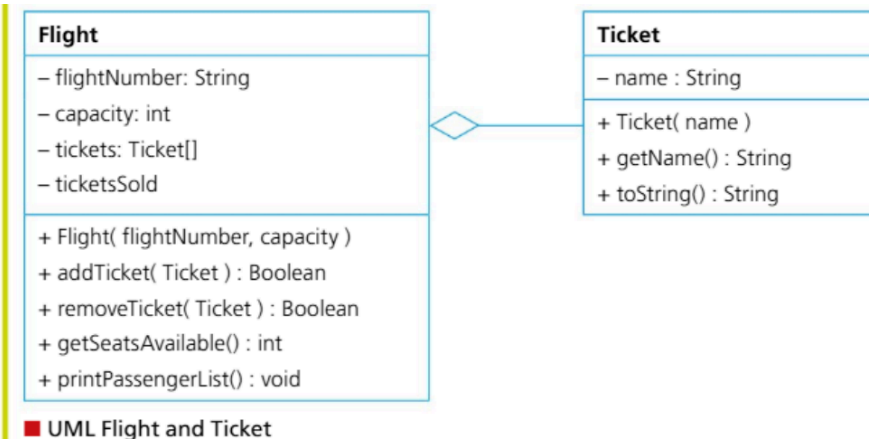
14. Outline an example, not based on any given in this book, that illustrates the difference between composition and aggregation.

15. Describe the factory design pattern and give an example of its use.

Programming Exercises (pp. 437–443)

1. **Extend bank accounts:** Implement the bank account example provided at the end of Section B3.1. Use file-reading techniques from B2.5 to read `names.txt` and `bank-transactions.txt` into arrays of strings, create an array of bank accounts, process the transactions, apply interest calculations on all accounts, and print all accounts' final balances.

2 **Flight reservation system:** You are constructing a ticket reservation system for a budget airline based on the following UML diagram:



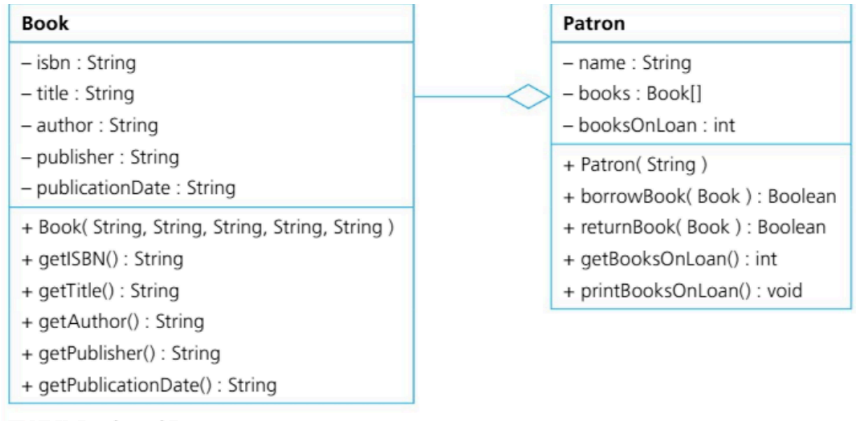
UML Flight and Ticket

Create the `Ticket` class and `Flight` class in code to adhere to the rules:

- The `capacity` variable in the `Flight` constructor indicates the number of seats available on a given flight. Use this variable to determine the size of your `tickets` array.
- Implement `addTicket()` and `removeTicket()` to add or remove a ticket from the `tickets` array. `ticketsSold` should increment whenever a new ticket is added, and decrement whenever a ticket is removed.
- `getSeatsAvailable()` should return `capacity` minus `ticketsSold`.
- `printPassengerList()` should print a list of all the names of tickets for a given flight.

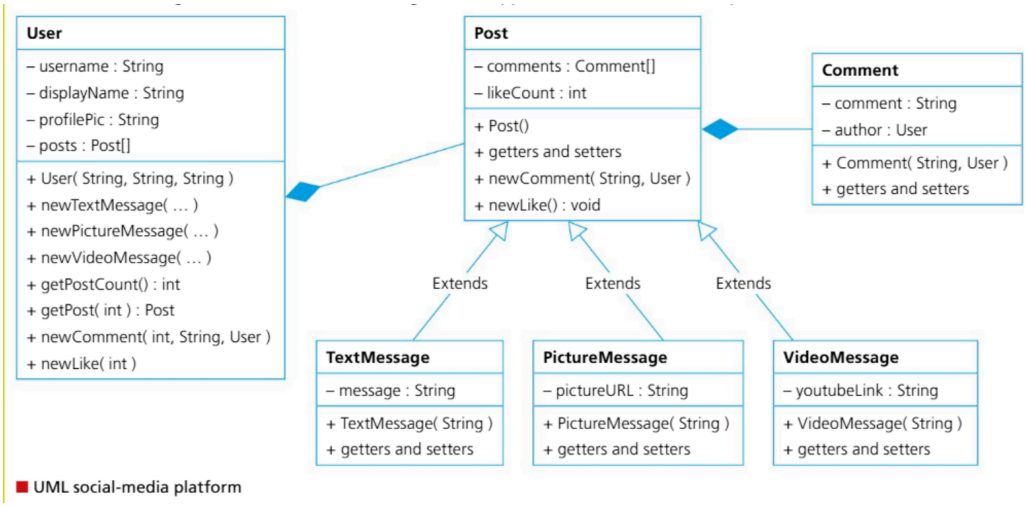
3 **School enrolments and grade book:** Extend the `Student` and `Assessment` example from Section B3.1 to include a new class called `Course` that contains an array of students who are enrolled in it. `Course` should contain `addStudent()`, `removeStudent()`, `printClassList()` (in name sort order) and `getClassAverage()`.

4 **Library system:** Create a `Book` class and `Patron` class as per the following UML diagram:



Each patron can only have a maximum of three books on loan at a time; if a patron attempts to borrow a fourth book, it should be denied.

5 **Social-media platform (HL):** Use the following UML as the basis for creating an OOP application for a social-media platform:

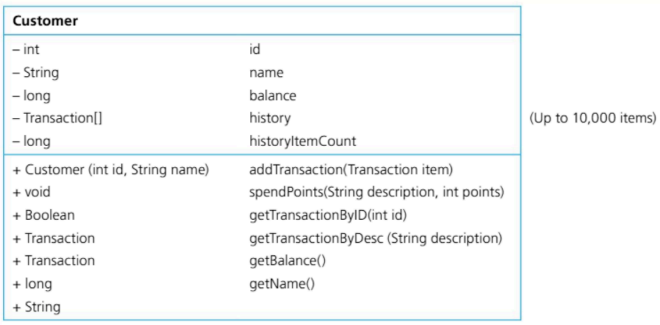


Construct the `User`, `Post`, `Comment`, `TextMessage`, `PictureMessage` and `VideoMessage` classes.

Exam Practice Questions — Topic B3 OOP (pp. 443–452)

1 **Customer loyalty system:** A chain of stores has launched a new customer loyalty program, where each dollar customers spend accrues loyalty points that can be exchanged for discounts on future purchases. The following classes exist in the system:

- `Customer`: Represents customer information, including a list of all their purchase history items
- `Transaction`: Represents the purchase of a single item by a customer.



UML Customer class

(a) State the relationship between `Customer` and `Transaction`. [1]

(b) Construct a simplified UML diagram showing the relationships between `Customer` and `Transaction`. [2]

(c) Outline the significance of the minus sign in front of `long balance` in the UML diagram. [2]

(d) Construct the code for the constructor of `Customer`. [3]

(e) Describe the purpose of the `historyItemCount` property. [2]

(f) Construct the code for the `getTransactionByDesc(String description)` function using a linear search. [4]

(g) Construct the code for the new `sort()` function to be added to the `Customer` class using a selection sort to sort transactions alphabetically by `description`. [6]

(h) Now that the `history` array is sorted, construct new code for `getTransactionByDesc(String description)` that will implement a binary search algorithm. [5]

(i) The `spendPoints()` function should first check whether the customer has enough points for the transaction and return `false` if not. Assuming enough points exist, it should create a new `Transaction` object that is added to the history, and deduct the points spent from the balance. Construct the code for the `spendPoints()` function. [6]

(j) The `getBalance()` function should iterate over all items in the `history` to calculate the correct balance, then update the value stored in the `balance` property appropriately, and finally return that value. Construct the code for the `getBalance()` function. [6]

(k) State the output from the following test block in `main()`:

```
t = Transaction(250, "Special deal")
customers = [None for _ in range(10)]
customers[0] = Customer(0, "Ava")
customers[1] = Customer(1, "Brian")
customers[2] = Customer(2, "Cherry")
customers[0].add_item(50, "Burger meal deal")
customers[0].add_item(Transaction(100, "Birthday bonus"))
customers[1].add_item(200, "Bluetooth earphones")
customers[1].spend_points(50, "Discount for shopping")
customers[2].spend_points(80, "Discount for shopping")
customers[2].add_item(t)
print(customers[0].get_balance())
print(customers[1].get_balance())
print(customers[2].get_balance())
print(customers[3].get_balance())
```

[4]

2 **Animal shelter:** A programmer created the classes `AnimalShelter` (with `location` and fixed-capacity array `animals[]`) and `Animal` (with private attributes `id`, `species`, `age`, `isHealthy`) to manage animals and the adoption process.

(a) Explain the purpose of encapsulation in object-oriented programming and how it is applied in the classes above. [2]

(b) Construct the missing code for the constructor of the `Animal` class (`public Animal(String id, String species)` / `def __init__(self, id, species)`). [3]

(c) Explain the use of the keyword `this` in the `setAge` method of the `Animal` class. [2]

(d) Construct code to create an instance of the `Animal` class with the ID `"C4T00123"` and species `"Cat"`. [2]

(e) Construct code to set the age of the object created above to 3 years. [2]

(f) Construct the method `addAnimal(Animal a)` that will add an `Animal` to the first empty position of the array `animals[]` in the `AnimalShelter` class and return the position at which it has added the animal. If the array is full and the animal cannot be added, the method should return `-1`. [5]

(g) **HL extension:** Two subclasses, `Dog` (with `isVaccinated`) and `Cat` (with `isNeutered`), inherit from `Animal`. Construct a UML diagram that shows the relationships between the `AnimalShelter`, `Animal`, `Dog` and `Cat` classes. Include only the class names and relationships. [3]

(h) Construct a method `getDescription()` in the `Animal` class that returns a string describing the animal, including its ID and species. [2]

(i) The array `animals[]` in the `AnimalShelter` class is used to store instances of any kind of animal, including `Dog` and `Cat`. Justify why `Animal` is a suitable type for this array. [2]

(j) The shelter rewards every tenth volunteer with a free pet-food voucher using `Rewards.printPetFoodVoucher()`. A `getVolunteerCount()` method has been added to `AnimalShelter`. Describe the necessary changes to the `AnimalShelter` class and any other methods to integrate the volunteer reward program into the system. [5]

(k) The method `removeAnimal(String ID)` in `AnimalShelter` searches the array for an `Animal` object with a specified ID and removes it by setting that index to `null`, returning a reference to the removed `Animal` object. Construct the `removeAnimal(String ID)` method. [5]

3 **Streamify music service:** A programmer created the classes `MediaLibrary` (with `name` and dynamic list `tracks`) and `Track` (with private attributes `id`, `title`, `artist`, `duration`, `genre`).

(a) Outline why the programmer may have decided to use a list instead of an array to store the tracks within the `MediaLibrary` class. [2]

(b) Describe the relationship between the classes `Track` and `MediaLibrary`. [3]

(c) Discuss the importance of using the keyword `this` in the constructor of the `Track` class. [2]

(d) Construct code to create an instance of the `Track` class with ID `TRK12345`, Title `Oceans`, Artist `Dive Deep`, Duration `215` seconds, and Genre `Ambient`. [2]

(e) Construct a method in the `MediaLibrary` class that allows a user to add a `Track` to the library. [2]

(f) Construct a method `searchByGenre(String genre)` in the `MediaLibrary` class that returns an array of `Track` objects that match the genre. [4]

(g) Discuss the process of overriding methods of the `Track` class if it were to be extended by `Podcast` and `Video` classes. [4]

(h) Two classes, `Playlist` (with `name` and `trackList`) and `User` (with `username` and `playlists`), are introduced. Construct a UML diagram that shows the relationships between the `MediaLibrary`, `Track`, `Playlist` and `User` classes. Include only the relationships without the attributes or methods of each class. [4]

(i) Construct a method in the `Playlist` class that returns the total duration of the playlist. [2]

(j) Explain why the `Playlist` class uses a list to store tracks instead of an array. [2]

(k) Discuss why composition is used between the `User` and `Playlist` classes, and between the `Playlist` and `Track` classes. [2]

(l) Streamify rewards users with a free month of premium subscription for every 100 tracks they add to their playlists via `rewardUser` in `User`. Without writing code, describe any changes required to the `addTrack` method in the `Playlist` class and the `User` class to make the new reward system work. [5]

(m) Construct the `removeTrack()` method for the `Playlist` class that allows users to remove a track from their playlist by specifying the track's unique ID. [4]

End of assignment · Check your answers against the chapter at gurujazzy.com/subjects/ib-computer-science/b3-2-oop-multiple-classes/