Java OOP Assessment: Peer Tutor Booking Platform

Answer Paper

Step 02

Task 01

1st Question Answer – Because we don't know how to implement getRole() method in User class. Because it is different according to User type. So we want to keep it without implementation. If we create this User class without using abstract we should have to implement it. At least we should have to put {}. When we keep it like this, in child classes never show any errors though we keep it without implementation. But it is affected to our final product. So when we use abstract for User class, Then we can keep that method without implementation in User class. But in child class we should implement it according to that. If we keep it without implement in child class also it shows error massage to us. So finally if we want to keep any method without implementing (If we don't know how to implement it according to child classes) we can use abstract classes to do this.

2nd Question Answer – When we use private access modifier to our variables, it helps to protect our class information. Then in another classes unable to change this variables directly. When we use this we can check, record, or change data before give access to updates.

Task 03

3rd Question Answer - We are using the OOP principle of inheritance to inherit Student and Mentor from User. Both of Student and Mentor classes have same variables (Ex: id, name) and some methods also. So we can create a common classes call User and we can put that common variables and common methods to that User class. Then we can use that variables and methods to child classes also by using inheritance. So we shouldn't want to write that same code lines again and again in student and mentor classes. If we don't use inheritance we should have to write common variables and common methods in Student and Mentor classes separately.

4th Question Answer - We used two ways to connect classes and First one is we made Student and Mentor both types of User. This means Student and Mentor get all the User features automatically. Second one is we can put one class inside another class, like when a Student has a Course or when a Mentor keeps record of their Students. Our classes don't do this yet, but we can use this to get bigger and more complex.

5th Question Answer – If we don’t use static keyword to nextId variable, it becomes a instance variable. Then when we create a instance that variable create newly for that instance. It means When we create a object, all of the objects get same number (In this scenario number 1). Because All of objects get fresh and newly created one. But in this scenario we need this variable as a class level variable. It means we need this variable as a common for all objects. So we can do this with using static keyword. Now when we run this program this static variable save in heap with class and objects creates in stack and they access this variable from heap.

Step 03

Task 06

6th Question Answer – When we use SubjectSession and CareerSession, they can do different things even when we treat them as regular Session objects. This is called polymorphism. The same code works differently based on what kind of session it really is. Java knows which version of a method to use when the program runs. Simply we can change that code which have in parent class, according to the session type.

7th Question Answer – In this scenario we use Session abstract class to create a structure for sessions. Now we can create all common things in this class, but sometimes we can’t implement some methods. Because we don’t know how to implement it according to session type, but we want to create this structure also. So we have to create that method in this class and we want to keep it without implementing. So we should have to create this class as a abstract class to create this abstract methods. We can do this with interfaces also but interfaces only define method contracts and we unable to implement any methods or variables. So we can’t create a structure with this interfaces for Session. So I think in this scenario the most suitable option is the abstract class to create this structure for Session.

8th Question Answer – If we want to create a new session type call GroupStudySession firstly we should create a new class for it (GroupStudySession.java) and after that it should extends from Session class. Now that session structure automatically comes to this new class also. So this is how our inheritance structure helps to new feature creations. After that we should create specific variables and we should definitely override and implement our abstract methods.

Step 04

Task 08

9th Question Answer – When we use interface to add notify() method to Users classes we can implement Users classes from Notifiable class. But if we use java class for this Notifiable we should have to extends Student and Mentor from Notifiable class. But in java we unable to extends from multiple classes. Because we already extends this student and mentor classes from User class. Other thing is if we want to send notifications to others also (excepting Students and Mentors) we can do it very easily with interfaces also. So finally this makes our system easier to change.

10th Question Answer –

Abstract Class

- When we want to inherit we can extend only one class.
- In abstract class we can create both of abstract and concrete methods.
- We can define instance variables.
- We can create a constructor.

Interface

- When we want to inherit we can implement multiple interfaces.
- In interfaces we can create only abstract methods. (Excepting default, static methods)
- We can define only public static final variable.
- We are unable to create a constructor.