Steps for Exporting and Saving the Source Code from Eclipse:

1. From the Eclipse menu (the bar at the top), select `File -> Export`.
2. In the window that opens, select `Archive File` from the `General` category and click `Next`.
3. Select the project on the top left (tick the box to the left of the name) and click the `Browse...` button to choose the path where the file will be saved.
4. Finally, click the `Finish` button to save your code in compressed format in a zip file, which you will send via eclass.

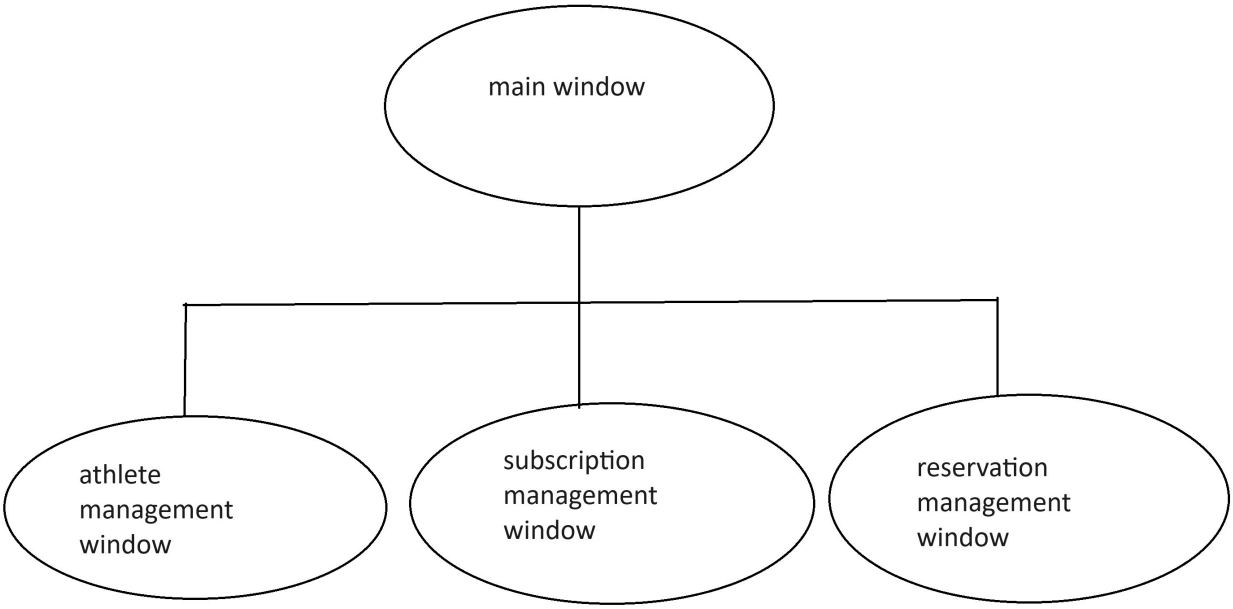
Assignment Description

This assignment focuses on developing a software system for managing sports academies. The system will allow administrators to register and manage information about athletes, coaches, sports, facilities, reservations, subscriptions, and payments.

Develop a program in the Java programming language that will include the following classes and interfaces:

1. **User**: A base class that includes common fields for athletes and coaches: unique ID (incremental number), first name, last name, gender, date of birth, contact information.
2. **Athlete**: Extends the User class and includes additional fields: professional (boolean) and experience level (from 1 to 5).
3. **Coach**: Extends the User class and includes additional fields: Sport and degrees.
4. **Sport**: A class that includes the fields: name of the sport, and whether it is only for professionals.
5. **Facility**: A base class that will have data: the name of the facility and its maximum capacity in athletes.
6. **TrainingProgram**: A class that includes the fields: unique ID (incremental number), Sport, Facility, Coach, minimum experience level for participating athletes, whether a reservation is required every week for athletes to participate (Boolean), participant gender (male, female, mixed), duration in minutes, and day of the week.
7. **TrainingProgramReservation**: A class describing a reservation made by an Athlete who has a subscription for a TrainingProgram that requires a reservation each week to participate, if there is space available in the Facility for that day and time. It includes the fields: unique ID, Athlete, TrainingProgram, and Reservation Date.
8. **Subscription**: A class implementing the Pricelist interface and includes the fields: unique ID, Athlete, TrainingProgram, and the monthly cost.
9. **Enrollment**: A class implementing the Pricelist interface representing the registration of the Athlete in the academy, which is done once. It includes the fields: unique ID, Athlete, date, cost, and the discount percentage the athlete is entitled to on their subscriptions.
10. **Payment**: A class implementing the Pricelist interface where the payment is recorded. It includes the fields: unique ID (incremental number), date, payment method, the subscription or registration it concerns, and the total cost.
11. **Interface PriceList**: Includes the charges for each service of the sports academy (based on the following price list). It also includes a method, `calculateTotalPrice()` (returns double).

The application will have a main window from which the user can access the windows where all the above functions are performed, as shown in the following diagram.



An indicative, simple form of the main screen is shown in the following diagram, where the transition to the various windows is done using buttons.

This are the words for you to copy paste them. Διαχείριση Αθλητικών Ακαδημιών, Διαχείριση Αθλητων, Διαχείριση Συνδρομών, Διαχείριση Κρατήσεων

Athlete Management

In this window, the administrator can register athletes in the system and change their details. Specifically, the following functions will be supported:

1. **Athlete Registration**: A new user-athlete will be created with all the necessary details. The unique ID of the athlete is an incremental number.
2. **Registration Payment**: To complete the process, the user's registration fee must be paid, after calculating its price, and the final discount percentage the athlete is entitled to will be recorded in the registration. During this process, the Enrollment and Payment objects will be created. The Enrollment code will be the same as the athlete's code since each athlete will have a unique registration. The code of each Payment will be an incremental number.
3. **Display Athlete List**: A table will display all athletes, their basic details, the registration date, and the amount they have paid for it.
4. **Edit Athlete Details**: The user will be able to change the details of the selected athlete from the list.

Subscription Management

The window through which the management of the sports academy subscriptions will be done and will have a table showing all the subscriptions that have been made. The following functions will be supported:

1. **Create New Subscriptions**: One or more Subscription objects will be created. The user will select Athlete, TrainingProgram, and fill in all the other required details to create each subscription for payment. Note that for each subscription, the criteria mentioned above for the athlete's level, if they are professionals, gender, etc. must be met. Also, for subscriptions to be created, the Athlete must have already registered to utilize the discount they are entitled to and must not already have a subscription to the specific TrainingProgram.
2. **Payment**: When the "Payment" button is pressed, the Subscription code will be automatically calculated, and a Subscription object will be created (using all available data), as well as the corresponding Payment object. The unique code of the Subscription, which is calculated automatically, cannot be changed and will be in the format <AthleteID_TrainingProgramID_<incremental number>>.
3. **Display Subscription List**: A table will display the list of an athlete's or all athletes' subscriptions.
4. **Cancel Subscription**: The administrator will be able to select a subscription from the table and cancel it.

Reservation Management

The window through which the management of the reservations made by athletes for each TrainingProgram that requires a reservation each week. The window will have a table showing the scheduled TrainingProgramReservations, and the following functions will be supported:

1. **Create New Reservation**: A new TrainingProgramReservation object will be created. The unique ID of the object will be automatically calculated and cannot be changed. Specifically, the unique ID will be in the format <TrainingProgramID_AthleteID_yyyyMMdd>, where yyyyMMdd is the date the reservation concerns. When the user fills in all the required details and clicks the "Reservation" button, the code will be automatically calculated, and a TrainingProgramReservation object will be created using all the available data. The athlete

must have a subscription to the specific TrainingProgram and must not have made another reservation for the same date in the same TrainingProgram.

2. **Display Reservation List**: A table will display the list of reservations that have been made. The display can change to show only the reservations of an athlete, reservations for a specific TrainingProgram, as well as an option for future reservations or history.
3. **Cancel Reservation**: The user will be able to select a reservation from the table and cancel it up to 1 day before the scheduled date.

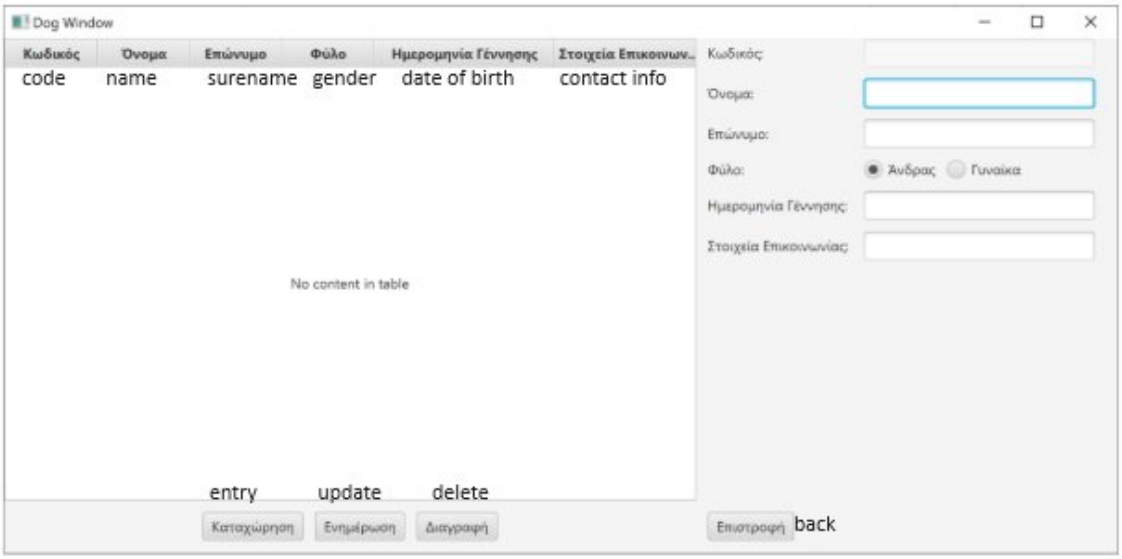
Academy Price List:

Registration Price List

- Regular athlete: 50 €
- Professional athlete: 20 €

Subscription Discount Percentage

- Levels 1-2: 10%
- Levels 3-4: 20%
- Level 5: 30%
- Additional discount for professional athletes: 20%



Notes:

1. In all classes, fields will be defined as private.
2. In all classes, setter and getter methods will be defined to set and retrieve the values of each class's fields.
3. In each class, a constructor will be defined with parameters for all the class's fields, and any other constructor deemed necessary for the assignment's needs.
4. ArrayLists will be used to store the objects of the system's classes.
5. When the program starts, system initialization will occur. Specifically, instances of all classes will be created.
6. The calculation of any cost will be done automatically based on the user's choices and will appear in a relevant field on the screen, which will NOT be editable.
7. In all cases of data entry by the user, data validity checks should be performed (e.g., if details are not entered, if integers are entered in integer fields, etc.). It should also be ensured that the user does not make other errors, such as pressing delete to delete an entry from the table (i.e., an object of a class) without first selecting the object they want to delete.
8. The source code must contain sufficient comments.
9. An example window for athlete management is shown below (the table includes only a subset of the athlete-related details). As shown in the screen, for fields that must be filled with a specific value from a set of available options (e.g., Title of movie, etc.), appropriate components (e.g., radio buttons) should be used.