Ticket Reservation - Encapsulation

We have been constantly listening that encapsulation is one of the best OO practices and is implemented by marking all variables as private and having getter and setter methods. Lets' try and understand it through a scenario.

You have developed a ticket booking application used by one of the leading travels. The travels have many other booking agents as well. Since your app works perfectly fine, The travels have adviced the other vendors / agents to call your methods for booking. During the lean period, The travels come out with an interesting statergy. They start giving discounts to children and old-age people to encourage them to book tickets at a lower price. The travels decide on day - to - day basis the discounts for various segments, namely children, OldAge and females.

In the above scenario, the discounts have to be communicated to each and every vendor, so that they can include in their ticket price calculation. There is every chance some of them might miss the inclusion and the whole cost goes wrong.

Encapsulation comes to the rescue. The simple solution would be, the vendors would not calculate the total amount directly by accessing your variable. They would just call your getAmount method which would calculate the total amount based on discounts. So, the discounts have to be changed at only one place.

Lets just go ahead and implement our Ticket Reservation Functionality.

All the attributes going forward has to be marked private.

Please have
A Ticket class with passengerCount , price and totalAmount as its attributes, an addPerson (Person person) which would increase the passengerCount and total amount when a new person is added.
A Person Class with name, gender and age as its attributes.

Use Appropriate Getters Setters & Constructors as you feel necessary.

Assume the following discounts

For Children age less than 16 - 50% discount
For Old age greater than 64 - 25% discount
For Ladies - 10% discount

Sample input output:

Enter Price of a ticket :

```
100
Enter Number of Persons :
3
Enter Details for Person 1 :
Name :
Amar
Gender (M or F) :
M
Age :
14
Enter Details for Person 2 :
Name :
Thana
Gender (M or F) :
M
Age :
20
Enter Details for Person 3 :
Name :
Rajini
Gender (M or F) :
M
Age :
70
Ticket Details are :
Number of Passengers : 3
Price of a ticket : 100.0
Total Amount : 225.0
```

Ticket Reservation - Association

A trivial concept in OOPS is association. Association is relationship between objects. If we look at our earlier example on Ticket Reservation, you would find that a ticket has many persons. (OR) A Ticket is composed of persons. Lets try and emulate the same by adding a Person Array in the Ticket.java.

```
private Person[] passengers;"
```

In the addPerson method, add the new person parameter to the above Person Array.
Loop through the array to print the person details as given in sample output.

sample input output :

```
Enter Price of a ticket :
200
Enter Number of Persons :
5
Enter Details for Person 1 :
Name :
Ram
Gender (M or F) :
M
Age :
30
Enter Details for Person 2 :
Name :
Sita
Gender (M or F) :
F
Age :
28
Enter Details for Person 3 :
Name :
Kowsalya
Gender (M or F) :
F
Age :
70
Enter Details for Person 4 :
Name :
Lavan
Gender (M or F) :
M
Age :
12
Enter Details for Person 5 :
Name :
Kusan
Gender (M or F) :
M
Age :
12
Ticket Details are :
Number of Passengers : 5
```

```
Price of a ticket : 200.0
Total Amount : 730.0
Passenger Details :
1) Ram (M 30)
2) Sita (F 28)
3) Kowsalya (F 70)
4) Lavan (M 12)
5) Kusan (M 12)
```

Display the Student Details

Problem Statement:
Write a java program to get the details of the student and display them . Use parameterized constructor to initialize the student object.(Refer Sample Input and Output).

Input & Output Format :
The input consists of 2 Strings (name),(department), 1 integer (year) and float (percentage).

Sample Input and Output :

```
Enter the Student Details
Enter the Name :
ram
Enter the Department :
cse
Enter the Year :
2
Enter the Percentage :
8.8
The Details of the Students are
Name : ram
Year : 2
Department : cse
Percentage : 8.8
```

Album songs interface

Problem Statement :
Write a java program to get the album name , music directors name , price and number of songs , singer(s) name .

Problem Constraints:

Create class Album,Song.
Album class having the attributes are directorName, price, albumName,ArrayList<String>Song.
Song Class having the atributes are songName,duration,ArrayList<String>Singer.
Write the Program according to the above constaints.

Sample Input and Output :

Enter the album name
Punjabiyaan Di Battery
Enter the music director name
Sachin Gupta
Enter the price
120.00
Enter the number of songs
2
Enter the song 1 details
Enter the song name
Mere Dad Ki Maruti
Enter the duration
4.30
Enter the number of singers
1
Enter the singer 1 name
Diljit Dosanjh
Enter the song 2 details
Enter the song name
Jogi Mahi
Enter the duration
7.20
Enter the number of singers
3
Enter the singer 1 name
Sukhwinder Singh
Enter the singer 2 name
Shekhar Ravjiani
Enter the singer 3 name
Himani Kapoor
albumname : Punjabiyaan Di Battery
musicdirector : Sachin Gupta
price : 120.0

```
songs :
name : Mere Dad Ki Maruti, duration : 4.3
singers : Diljit Dosanjh
name : Jogi Mahi, duration : 7.2
singers : Sukhwinder Singh , Shekhar Ravjiani , Himani Kapoor
```

SAMPLE INPUT & OUTPUT 2:

```
Enter the album name
Gem Of Music
Enter the music director name
A.R. Rahman
Enter the price
250
Enter the number of songs
0
Album is empty
```

Runtime Polymorphism for Vehicle Class

Problem Statement:

Write a java program to perform runtime polymorphism .

Problem Constraints:

- Create a super class Vehicle and two sub classes Car and Bike.
- In Car class and Bike class override the method displayDetails() to display the details of Car and Bike.
- In main method read the corresponding inputs and display the results.

SampleInput & Output 1:

```
1)Car
2)Bike
2
Enter the details of bike
Enter the color
black
Enter the maximum speed
120
Enter the number of seats
2
Enter the number of wheels
2
Enter the status of diskbreak(true/false)
false
Bike Details
```

```
Color : black
Maximum Speed : 120
Number of Seats : 2
Number of Wheels : 2
Disk Break : false
```

SampleInput & Output 2:

```
1)Car
2)Bike
1
Enter the details of car
Enter the color
red
Enter the maximum speed
200
Enter the number of seats
4
Enter the number of wheels
4
Enter the number of doors
4
Enter the status of AC(true/false)
true
Car Details
Color : red
Maximum Speed : 200
Number of Seats : 4
Number of Wheels : 4
Number of Doors : 4
Ac : true
```

SampleInput & Output 3:

```
1)Car
2)Bike
3
Invalid Option
```