Assignment 2: Inheritance and Encapsulation

Before attempting this project, be sure you have completed all the reading assignments, non-graded exercises, discussions, and assignments to date.

Design and implement Java program as follows:
(1) There will be a *Snack* class with following attributes: id (combination of numbers and letters), size (values S, M, or L), and price
(2) There will be two child classes *FruitSnack* and *SaltySnack* with the following additional attributes:

- *FruitSnack***:** It includes citrus fruit indication (value of this variable of is true or false)
- *SaltySnack***:** It includes nut snack indication (value of this variable of is true or false)

(3) The Snack class hierarchy must provide the following functionality:

- On creation, a snack instance must be given all attribute values except its price, which must be calculated
- Price is calculated as follows:
  - There is a flat fee of $19.99 for S snack, $29.99 for M snack, and $39.99 for L snack.
  - `FruitSnack` has an additional fee of $5.99 when it has a citrus fruit. Please add only a single citrus fruit, and no preventing coding is required to limit adding more than one.
  - `SaltySnack` has an additional fee of $4.50 when it has a nut snack. Please add only a single nut snack no preventing coding is required to limit adding more than one.
- Each class must have a method to return or display the class's values to the console

(4) Implement OrderSystem class with main method with following functionality:

- Order a snack and after ordering it will display the snack type, its size, id and price
- Exit program

(5) Your classes must be coded with correct encapsulation: private/protected attributes, get methods, and set methods
(6) There should be appropriate overloading and overriding methods
(7) OrderSystem should take advantage of the inheritance properties (e.g., use Snack variable regardless which snack instance as appropriate)
(8) The prices for S, M, L, citrus fruit, nut snack can be hard coded in the program.

Style and Documentation:
Make sure your Java program is using the recommended style such as:

- Javadoc comment with your name as author, date, and brief purpose of the program
- Comments for variables and blocks of code to describe major functionality
- Meaningful variable names and prompts
- Class names are written in upper CamelCase
- Constants are written in All Capitals
- Use proper spacing and empty lines to make code human readable

Capture execution:
You must capture the screen shots of running the code for different test scenarios and include the screen shots of the test runs in the documentation. The following is a sample of a few of the test scenarios for you to follow, and you can add more. Any variation with the menu shown below, or any change in the language of the display or the display format for the menu for a justified reason is accepted.

In the test scenarios the symbol of question mark (?) means that is the value your program will compute. For example, your program will display the value of id, that is noted as ?
Your program will compute the value of the price and display it, and that is also indicated as ?.

Test Scenario 1:

```
MENU
1: Order a Snack
2: Exit program
Enter your selection: 2
Thank you for using the program. Goodbye!
```

Test Scenario 2:

```
MENU
1: Order a Snack
2: Exit program
Enter your selection: 1
Do you want Fruit Snack (1) or Salty Snack (2): 1
What size do you want: S, M, or L: S
Do you want citrus fruit included? true/false: true
You have chosen snack type = Fruit Snack, of type = S, id = ? and price = ?
```

Test Scenario 3:

```
MENU
1: Order a Snack
2: Exit program
Enter your selection: 1
Do you want Fruit Snack (1) or Salty Snack (2): 1
What size do you want: S, M, or L: M
Do you want citrus fruit included? true/false: false
You have chosen snack type = Fruit Snack, of type = M, id = ? and price = ?
```

Test Scenario 4:

```
MENU
1: Order a Snack
2: Exit program
Enter your selection: 1
Do you want Fruit Snack (1) or Salty Snack (2): 1
What size do you want: S, M, or L: L
Do you want citrus fruit included? true/false: false
You have chosen snack type = Fruit Snack, of type = L, id = ? and price = ?
```

Test Scenario 5:

```
MENU
1: Order a Snack
2: Exit program
Enter your selection: 1
Do you want Fruit Snack (1) or Salty Snack (2): 2
What size do you want: S, M, or L: S
Do you want nut snack included? true/false: true
You have chosen snack type = Salty Snack, of type = S, id = ? and price = ?
```

Test Scenario 6:

```
MENU
1: Order a Snack
2: Exit program
Enter your selection: 1
Do you want Fruit Snack (1) or Salty Snack (2): 2
What size do you want: S, M, or L: M
Do you want nut snack included? true/false: false
You have chosen snack type = Salty Snack, of type = M, and id = ? and price = ?
```

Submission Requirements:

Deliverables include Java program (.java files) and a single documentation including the screen shots of the test runs either as a Microsoft Word file, or as a PDF. The Java and Word/PDF files should be named appropriately for the assignment (as indicated in the Submission Requirements document. The word (or PDF) document should include screen captures showing the successful compiling and running of each of the test scenario. Test scenarios should include all required functionality of the program. Each screen capture should be properly labeled clearly indicated what the screen capture represents. Do not submit the screen shots of the test runs separately as image files but include (embed) them in the document and finally submit a single document.

Grading Rubric:

The following grading rubric will be used to determine your grade:

Criteria	**Level 3**	**Level 2**	**Level 1**
The Snack hierarchy classes	60 points Correct or almost correct attributes and inheritance structure	40 points Mistakes in implementation	20 points Missing or significantly incorrect implementation
Encapsulation	60 points Correct or almost correct code for encapsulation	40 points Mistakes in implementation	20 points Missing or significantly incorrect implementation
Calculation	20 points Correct or almost correct code to calculate price using overriding	14 points Mistakes in implementation	10 points Missing or significantly incorrect implementation
Test cases	30 points Correct or almost correct code to meet required functionality	20 points Mistakes in implementation	10 points Missing or significantly incorrect implementation
Program documentation and style, screen captures	30 points Correct or almost correct menu, program comments,	20 points Mistakes or incomplete menu, documentation and/or	10 points Missing or significantly incorrect menu, documentation

	identifiers, and screen captures	style, and screen captures	and/or style, or screen captures