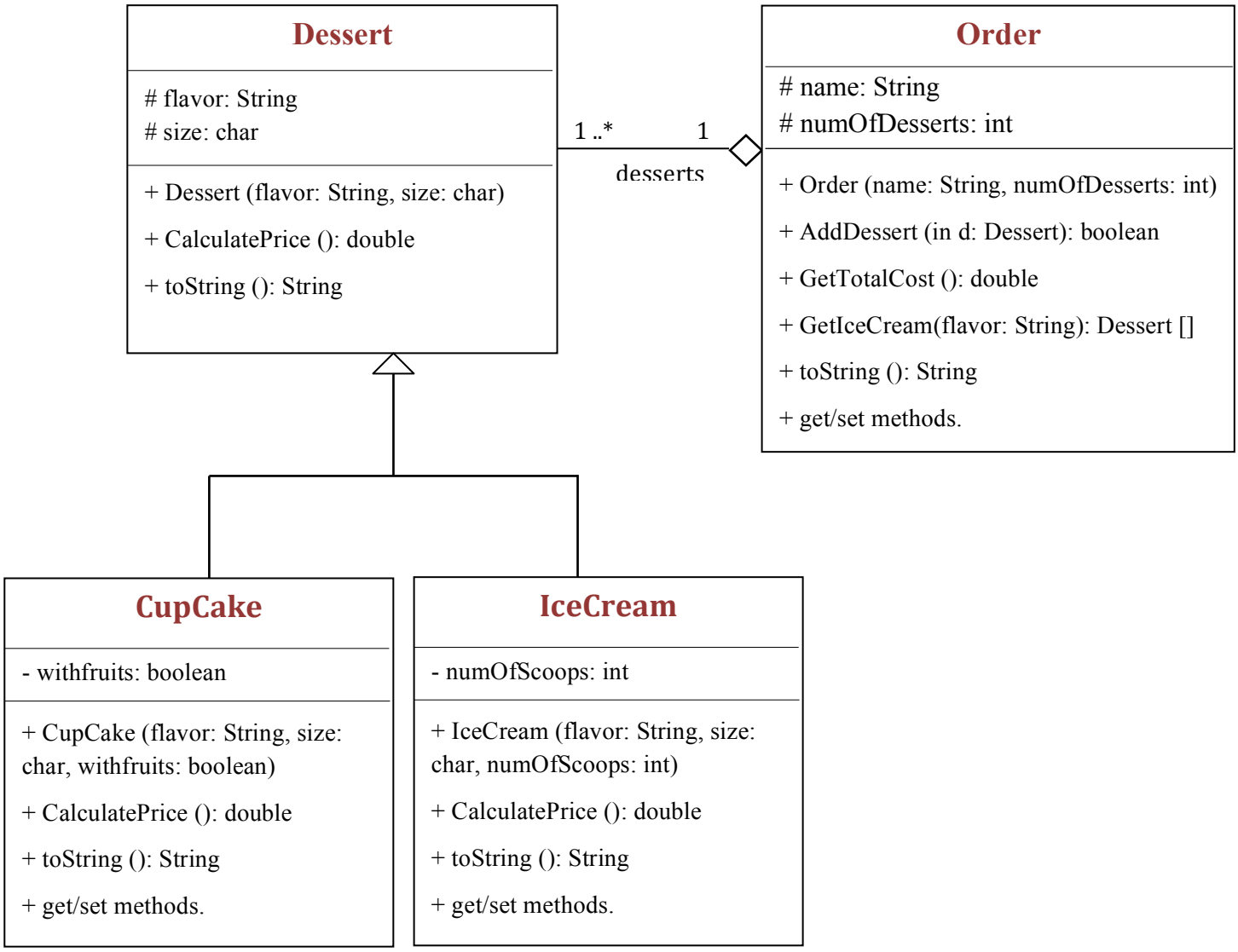
KING SAUD UNIVERSITY COLLEGE OF COMPUTER AND INFORMATION SCIENCES Computer Science Department		
CSC 113: Introduction to Programming II	**Lab_Sheet#4**	**2nd Semester 1438**



Given the above UML diagram, write the complete java implementation for all the classes according to the following description:

Dessert

1. Attributes
   - **Flavor:** a string to represent dessert flavor, such as Vanilla, Chocolate, Strawberry…etc.

- **Size:** a character to represent dessert size where **S** for small, **M** for medium, and **L** for large size.

2. Methods
   - **Dessert (flavor: String, size: char) >>** constructor to initialize flavor and size with the received parameters.
   - **CalculatePrice (): double >>** returns price of the dessert, it should return 0.0.
   - **toString (): String >>** returns a string representation of Dessert object .

CupCake

1. Attributes
   - **Withfruits:** a boolean to indicate whether the cupcake includes fruits or not.
2. Methods
   - **CupCake (flavor: String, size: char, withfruits: boolean) >>** constructor to initialize flavor, size, and withfruits with the received parameters.
   - **CalculatePrice (): double >>** calculates the price of the cupcake based on its size:
     S costs **6** SR
     M costs **10** SR
     L costs **12** SR
     There is an extra **3** SR for adding fruits.
   - **toString (): String >>** returns a string representation of CupCake object .

IceCream

1. Attributes
   - **NumOfScoops:** an integer to indicate the number of scoops in the ice cream (originally it comes with only one scoop but you can pay for more).
2. Methods
   - **IceCream (flavor: String, size: char, numOfScoops: int) >>** constructor to initialize flavor, size, and numOfScoops with the received parameters.
   - **CalculatePrice (): double >>** calculates the price of the ice cream based on its size:
     S costs **8** SR
     M costs **12** SR
     L costs **15** SR

There is an extra **3** SR for each additional scoop.

- **toString (): String >>** returns a string representation of IceCream object .

Order

1. Attributes
   - **Name:** a string to represent the customer name.
   - **NumOfDesserts:** an integer to track the number of the desserts currently in the order
2. Methods
   - **Order (name: String, numOfDesserts: int) >>** constructor to initialize the name and the size of the DessertList array that you will create.
   - **AddDessert (in d: Dessert): boolean >>** adds the received object in the first empty location in the DessertList array. The method returns true if the addition was successful and false otherwise.
   - **GetTotalCost (): double >>** returns the total cost of the whole order**.**
   - **GetIceCream (flavor: String): Dessert [] >>** returns an array of IceCream objects that are in DessertList with the given flavor.
   - **toString (): String >>** returns a string representation of Order object and all the desserts in the list .

Finally, implement **TestOrder** class with **main** method to do the following tasks:

- Prompt the user to enter customer name and maximum number of desserts in the order.
- Create an object (*order1*) from class Order with the entered information.
- Then display the following menu until the user choose to exit:
  1. Add a new dessert (CupCake or IceCream) to *order1*.
     - Ask the user to choose between adding a CupCake or IceCream.
     - Prompt the user to enter all the required information based on that.
     - Display a meaningful message reflecting the success of the addition.
  2. Print all desserts in the *order1*.
  3. Print total cost of the order.
  4. Print all the ice creams in *order1* with a specific Flavor (entered by the user).
  5. Exit.