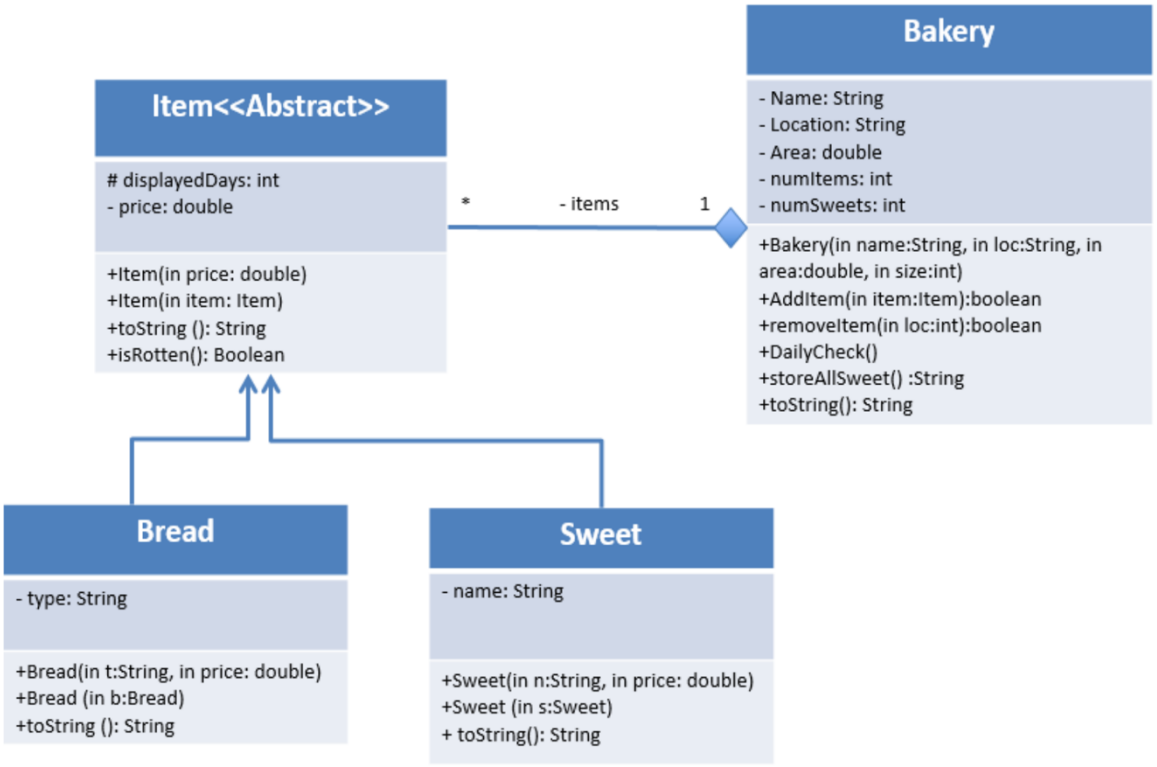
KING SAUD UNIVERSITY COLLEGE OF COMPUTER AND INFORMATION SCIENCES Computer Science Department		
CSC 113: Introduction to Programming II	**Sheet#2**	**2nd Semester 2020-2021**



You should implement a program to manage the Bakery operations. The Bakery offers two types of items for selling, which are *Bread* and *Sweet*. Given the above UML diagram, write the complete java implementation for all classes according to the following description:

Class Item (Abstract):

1. Attributes
    - displayedDays: Represents the number of days the item has been displayed in the bakery.
    - Price: Represents the price of an item.
2. Methods
    - Item(in price: double): A constructor to initialize the price to the value sent in the parameter. The constructor should check for any mismatch exception and prompt the user to enter a valid price until a valid value is entered.
    - Item(in item:Item): A copy constructor to initialize the attribute in Item class.

- toString(): Returns a string representation of the object information.
- isRotten(): An abstract method to check if a bakery item is rotten. An item rots when it is displayed more than its maximum freshness range.

  Note: the maximum freshness for an item is based on the bread type or the sweet name.

  - Bread (Type):
    - Arabic 3 days.
    - Toast 5 days.
    - Others 4 days.

  - Sweet (Name):
    - doughnut 2 days
    - English cake 4 days
    - cookies 3 days

Class Bread:

1. Attributes:

   - type: A string represents the bread type.

2. Methods:

   - Bread(in t:String, in price:double): A constructor to initialize the class attributes.
   - Bread(in b:Bread): A copy constructor to initialize the class attributes.

   - toString(): Returns a string representation of the object information.
   - Implement the following getters:
     - getType(): Returns the type of the bread.

Class Sweet:

1. Attributes:

   - name: A string represents the sweet's name.

2. Methods:

   - Sweet (in n:String, in price:double): A constructor to initialize the class attributes. The constructor should check the name of the sweet if it is a *doughnut*, *English* cake, or *cookies.* Otherwise, it will throw *"IllegalNameException"* to the calling environment. The program should prompt the user to enter a valid name until a valid one is entered.
   - Sweet (in s:Sweet): A copy constructor to initialize the class attribute.
   - toString(): Returns a string representation of the object information.

Class Bakery:

1. Attributes:

    - Name: A String represents the Bakery name.
    - Location: A String represents the location of the Bakery.
    - Area: a double represents the Bakery area.
    - numItems: an integer represents the number of items added to the list of items.
    - numSweets: an integer represents the number of sweets added to the list of items.

2. Methods:

    - Bakery(in name:String, in loc:String, in area:double, in size:int): A constructor to initialize class attributes.
    - addItem(in item:Item): Adds item object (bread or sweet) to the first empty location in the items list.
    - removeItem(in loc:int): Removes the item in the location loc from the list. Handel any exception that could be thrown in this method using try and catch blocks. This method returns true when the operation is performed successfully or false otherwise.
    - DailyCheck(): This method performs the daily check by:
        - Increasing the displayeddays by one for all items in the bakery.
        - Removing all rotten items from the bakery.
        - When the operation is completed, the information of the bakery (including all its items) should be saved to an object file "bakery.dat".
    - storeAllSweet(): Write the bakery name, number of sweets, and the total price for all the sweets in a text file "sweets.txt". The method returns the information stored in the file.
    - toString(): Returns a String representation of class information.
    - Implement the following getters:
        - getItems(): Returns an array of items.
        - getNumItems():Returns the number of items in the array items.
        - getNumSweets: Returns the number of sweets in the array items.

3. Test Class:

- Create a Bakery object with the name “Diplomat”, area “200”, the location “King Abdullah road” that can hold a maximum of 1,00 items.
- Add the following sweets and bread types:

Item	**Name/ Type**	**price**
Sweet	cookies	4
Bread	Arabic	2
Bread	Toast	4
Sweet	eclair	8

- Perform a daily check on all bakery items.
- Store all sweets information in “sweets.txt” text file and display this information in a friendly GUI using a text Area.
- Remove the item at index 1 in *items*.