Billing Software

Instructions to students:

1. Read the Problem Description carefully and understand the Scenario and Class Diagram.
2. Read the implementation details and understand the requirements you need to implement.
3. Demo.java file contains all the necessary classes and the classes are partially coded to implement the given scenario.

Guidelines:

1. You should only code your program at the space mentioned as "//Replace Your Code".
2. Your Code should satisfy the requirements specified in the Implementation Details.
3. Don't add additional methods in any class.

NOTE: NOT adhering to the above instructions and guidelines may lead to drastic reduction in your score even if the code is executing without any errors

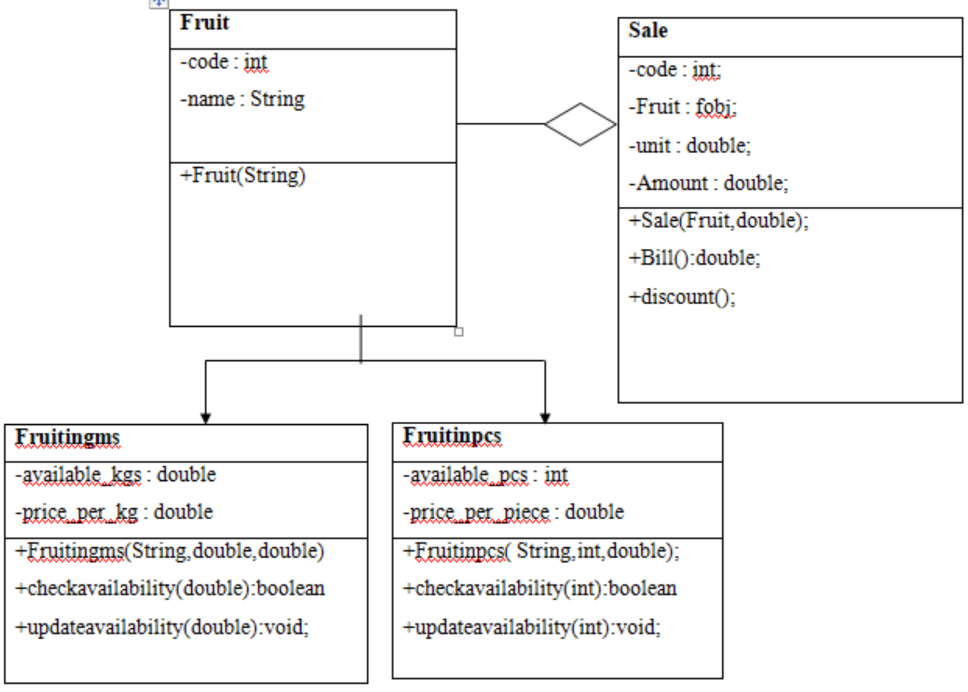
Problem Description:

"A to Z Fruit Shop" is a growing fruit shop in New Clark city. Since they are offering quality and fresh fruits to customers in best price, their customer strength is increasing tremendously. In this regard they need billing software to provide a fast and accurate service. Billing software also supports high to maintain sales data.

Business Rules:

1. *Every item stock should be verified for availability before billing.*
2. *5% discount should be provided for purchase more than Php 1,500.00*
3. *All fruits except banana are selling for price-per-kilo gram.*
4. *Selling Banana for price-per-piece.*

Class Diagram:



Rules:

1. Don't add additional instance variable in the given class.
2. Do Type conversions wherever it is necessary.
3. Don't make change in the Demo Class.

Implementation Details:

Fruit Class

This class is already coded for you.

Fruitingms Class

- Write constructor method **fruitingms()**for this class.
- Write **Check _availability()** method which takes "**need**" in kilograms as input and returns false if **available_kgs** is less than "**need"** else return true.
- Write **update_availability ()** method which takes **"need"** in kilograms as input and reduce this **"need"** from **available_kgs**.

Fruitinpcs Class

- Write constructor method **fruitinpcs()** for this class.
- Write **Check _availability()** method which takes "**need**" in pieces as input and returns false if **available_pcs** is less than "**need"** else return true.
- Write **update_availability ()** method which takes **"need"** in pieces as input and reduce this **"need"** from **available_pcs.**

Sale Class

- Constructor is already coded for you.
- Bill() method
    - Validate availability of fruit before billing
    - Calculate amount by product of unit and price
    - Provide discount by calling discount() if amount is greater than 500
    - Update availability of fruit after billing
- Discount method calculates discount of 5%.

Demo Class:

- This Class is a Starter Class.
- Code is already provided for you.
- You can modify the class for testing purpose.
- This class will not be evaluated.

This Code includes solution for the above question. The code below comment line // write your code here is the solution.

Solution Code:

```
class Fruit {
      private int code;
      private String name;
      static int counter = 1000;

      Fruit(String name){
            this.name = name;
      }
} //end class Fruit

class Fruitingms extends Fruit {
      private double available_kgs;
      private double price_per_kg;

      Fruitingms(String name, double available_kgs, doubleprice_per_kg) {
            super(name);
            this.available_kgs = available_kgs;
            this.price_per_kg = price_per_kg;
      }

      public double getprice() {
            return price_per_kg;
      }

      public boolean checkavailability(double need) {
            //write your code here
            if(available_kgs<need) {
                  return false;
            }
            return true;
      }

      public void updateavailability(double need) {
            //write your code here
            available_kgs = available_kgs - need;
      }
} //end class Fruitingms

class Fruitinpcs extends Fruit {
      private double available_pcs;
      private double price_per_piece;

      Fruitinpcs(String name, double available_pcs, doubleprice_per_piece) {
            //write your code here
            super(name);
            this.available_pcs = available_pcs;
            this.price_per_piece = price_per_piece;
      }

      public double getprice() {
            return price_per_piece;
      }
      public boolean checkavailability(double need) {
            //write your code here
            if(available_pcs<need) {
                  return false;
            }
            return true;
      }

      public void updateavailability(double need) {
            available_pcs = available_pcs - need;
      }
} //end class Fruitinpcs

class Sale {
      private int code;
      private Fruit fobj;
      private double unit;
      private double amount;
      static int counter = 2000;

      Sale(Fruit fobj, double unit) {
            this.fobj = fobj;
            this.unit = unit;
      }

      public double Bill() {
            //write your code here
            if(!fobj.checkavailability(unit)) {
                  System.out.println("Item not available");
                  return false
            }
            amount = fobj.getprice() * unit;
            if(amount > 1500) {
                  amount = discount();
            }
            fobj.pdateavailability(unit);
            return amount;
      }

      public double discount() {
            int disc = amount/100 * 5;
            amount = amount - disc;
            return amount;
      }
} //end class Sale

class Demo {
      public static void main(String args[]) {
            Fruit fobj = new Fruitingms("apple",50.6,80.0);
            Sale s = new Sale(fobj,2.0);
            System.out.println(s.bill());
            Fruit fobj1 = new Fruitinpcs("banana",200,3.0);
            Sale s1 = new Sale(fobj,10);
            System.out.println(s1.bill());
      }
}
```