Name: _____ Date:_____

Student #: _____

COMP3021: Java Programming In-class Exercise

The following table shows the class design of an OO program with no use of inheritance and interfaces. This has resulted in a lot of redundancy that needs to be removed. You are asked to "refactor" the classes so that no two classes have the same field names or method names. In the refactoring process,

1. You cannot change the meaning of the program. That is, each class will essentially have the same set of fields and methods either defined in itself or from super classes.
2. You can only move the fields and methods around. You cannot rename anything.
3. You can create super classes and interfaces and the associated fields and methods.
4. A class can at most extend from one super class. This is a Java program.

Orange	Pork
- sweetness: float - freshness: int - itemID:long -weight: float	- freshness: int - itemID:long - weight: float
+ isGreeen():boolean + unitPrice():float	+ unitPrice():float + isImported():boolean

Toothpaste	PaperPlate
- itemID:long - packageSize:float	- itemID:long - packageSize:float
+ unitPrice():float + isImported():boolean	+ unitPrice():float + isGreen():boolean

Fields:

sweetness: an index of the sweetness of the fruit; **freshness**: how many days in stock; **itemID**: the barcode ID in the super market; **weight**: the weight of the item; **packageSize**: the volume of the package;

Methods:

isGreen: decide if the item is produced by green methods, e.g., organic or recycled; **unitPrice**: return the price; **isImported**: decide if the product is imported

```
class Orange
{



}

class Pork
{

}

class Toothpaste
{

}

class PaperPlate
{


}
```

Additional classes: