**Project 3 – Objects, methods, selection, looping – MICHAEL BLANCO**

**Problem:**

The Ankh-Morpork Railway is introducing a dining car! Every day, the dining car is supplied with three delectable Food options by your local food supplier Cut-Me-Own-Throat Dibbler. What options are available, how many are on hand and what day cost changes from day to day.

|  |
| --- |
| Food |
| * type : String * units : int * price : double * allFood : int |
| + Food()  + Food( dish : String, amount : int , price : double )  + getType() : String  + getUnits() : int  + getPrice() : double  + subtractSoldUnits( int sold ) : void  + getRemainingFood() : int |

Finish the program that would let you dining car attendant offer the menu items to your passengers, keep track of remaining inventory and revenue from sold items. Close down food service when all food is sold.

* + - The standard constructor should set all instance variables to neutral values.
    - Watch out for the static variable allFood and the static method getRemainingFood(). allFood’s value is the amount of all food items on hand (available to sell), regardless of their type. Think about when allFood needs to be updated.
    - The subtractSoldUnits method needs to not only update the remaining units of the food item but also update how many items altogether are left to sell
    - There is only one place where the value of allFood can be initially set to hold the sum of units for all three menu items

* + - The *go* method will drive the program. Write it next. It must make calls to the *printMenu* and *sell* methods to print the menu and to communicate with the user about their purchases.
    - *go* will repeat printing the menu and selling until there is no more food of any sort left to sell,

i.e. the inventory of all three food items has been depleted.

* + - Inventory and revenue information is output as soon as the program starts. At that point your revenue should be zero.
    - The *go* method will display the remaining inventory after a valid sale has been made.
    - Not all user choices will be valid choices. Assume all choices the user makes are whole numbers.
    - THE *printMenu* method will only give a food item a numeric choice on the menu if there is some of it left to sell. If it’s been sold out, output the appropriate message in the menu (see sample output screenshot at the end of the handout).
    - The *sell* method decides how many units you can sell. If the client is requesting e.g. 5 units of food item 1 and you have 7 left, sell the user the 5 that are on hand. However, if the request is for 5, and you only have 3 of that food, only sell and charge for 3.
    - The *sell* method either how much money the user owes for their purchase, or to announce the choice is sold out, or that only units on hand will be sold.
    - The *sell* method will also calculate how much money was made on the sale, and it’ll pass that value back to where the method was called.
    - Updated inventory numbers and revenue value are output after each sale.
    - When the total inventory reaches zero, it’s time to close the dining car.

