CSG 100 Data Structures Fall 2004

Problem Set #1: Playing with Objects in Java

Goal:

In this exercise you are to play the role of a developer who is given a specification and description of the data structures and their operations. You then have to provide a Java implementation for this specification.

Instructions:

For each exercise make sure you provide a `main` method to run your code as well as **all** the test cases that you have used to test your code. Comment all of your code and provide any information about your homework in a separate text file called README.txt. Send **all** your files to *skotthe@ccs.neu.edu*.

1. Create a Java class with the name `Orange`. This class represents oranges in a supermarket. The class should be able to hold information about an orange, specifically its `weight` as an integer, and its `price` as a double. The class should allow for the access and modification of these variables through public methods.

Orange
weight:int price:double
getWeight():int setWeight(int):void getPrice():double setPrice(double):void

(20 points)

2. Create a Java class with the name `OrangeBasket`. `OrangeBasket` represents a small collection of *at most* 10 oranges. Provide methods to add to the basket as well as remove from the basket. (**NOTE**: you should only remove an orange if the index is valid **and** there is an orange in that index to be removed). `OrangeBasket` must also have the ability to calculate and report back on the total price and the total weight of the basket of oranges. Further more, `OrangeBasket` also provides a method called `contents` that prints on the screen all the information about its contents. i.e. the weight and price of each orange in the basket as well as the grand total for weight and price.

OrangeBasket
capacity:Orange[10]
addToContents(Orange):boolean removeFromContents(int):boolean calculateTotalWeight():int calculateTotalPrice():double contents():void

(25 points)

3. Create a Java type with the name `ArrayManipulator`. The `ArrayManipulator` should be able to take 2 integer arrays of size 20 each and allow the following operations on these arrays:
   - *identical():boolean* , returns true if the two arrays have the same elements at the same indexes.
   - *join():int[40*], joins the two arrays by concatenating one at the end of the other
   - *splice():int[40]*, joins the arrays at each index. i.e,. calling splice on the following two arrays {1,2,3} {4,5,6} should give {1,4,2,5,3,6} as result.
   - *isSame(int[20]):boolean*, returns true if and only if all the indexes contain the exact same value i.e, {1,1,1,1}.
   - *noDuplicates(int[20]):boolean*, returns true if and only if there are not duplicate values in the array.
   - *showDetails():void*, should "pretty print" the contents of the 2 arrays held as instance variables in `ArrayManipulator`.

(25 points)

4. Create a class `Palindrome`. The class should have an instance variable of type `String` and a method:
   - *isPalindrome():boolean*, returns true if and only if the instance variable of the class is a palindrome.

     Palindrome: A word, phrase, verse, or sentence that reads the same backward or forward.
   - Some examples of palindromes :
     - deed
     - level
     - pop
     - madam
     - radar
     - eye
     - civic

(30 points)