Tutorial 2 – Java OOP #2

Copy the content of `tutes.oop1` package to another package named `tutes.oop2` and complete the following tasks:

1. Change the `Pizza` class design to support the following types of pizzas and that `Pizza` is made abstract: ham-and-cheese pizza, pepperoni pizza, and tropical pizza. Each type of pizza should have its own constructor and description:
   a) ham-and-cheese: only have ham and cheese toppings
   b) pepperoni: only have cheese and pepperoni toppings
   c) tropical: have a combination of the three toppings

   Note:
   i. Add an instance variable name to `Pizza` which is initialised to the class name inside the constructor
   ii. Add a `Pizza.getName()` method to return the pizza name
   iii. You should change `Pizza.getDescription()` to protected and have it overridden by the subclasses
   iv. You should also add protected getter methods for the topping instance variables so that the subclasses can access them
2. Each pizza object is now defined in terms of the toppings that it has. Create a `Topping` inner class of `Pizza` that has the following instance variables (and the appropriate methods): name (e.g. cheese, pepperoni, ham), quantity and cost. Add a `Topping.calcCost()` method as well to calculate the cost for each topping. You also need to override the `toString()` method to give a description of a topping (e.g. `3` cheese toppings at `$2` each should output `3@$2 - cheese`). Change the `Pizza`'s constructor so that it takes `Topping` objects as parameters.

   Note: You will need to define `Topping` with the `static` keyword.
3. Make `Pizza` implement the `Comparable` interface to compare two `Pizza` objects based on cost
4. Add method `PizzaOrder.sort()` that sorts the pizzas in the ascending order of cost.
   a) Use the quick sort algorithm that operates over a Comparable array of objects. A version of this algorithm is provided in the attached Arrays class.
   b) You should add a private Pizza.reduceOrder method which remove all null items before sorting
5. Change the `PizzaDemo` class to work with the new design and the sorting function.
6. [Extra] More advanced sorting:
   a) Copy the content of `tutes.oop2` package to another package named `tutes.oop2b` to complete this task.
   b) Define a `Sorting` interface to sort a `Vector` of `Comparable` objects with the following method:
      - `sort(Comparable[]);`
   c) Define a derived interface SortingP of Sorting to perform the following additional types of sorting for Pizza objects:
      - `sortByName(Pizza[]);`
        sort by name in ascending order
      - `sortByPrice(Pizza[]);`
        sort by topping in ascending order
   d) Define a `PizzaSorting` class that implements the `SortingP` interface:

- the constructor method must take a sorting criteria argument (whose value must be either “name” or “cost”)
- sort should check the type of sorting to invoke the appropriate method
- use `Arrays.sort` (the quick sort function, in the provided file `Arrays.java`) in all the sorting functions

e) Add a method `PizzaOrder.sort(SortingP)` that sorts the pizza orders

f) Change the `PizzaDemo` application to use a `PizzaSorting` object to sort pizza orders:

- experiment with the price and name sorting criteria