Tutorial 3 – 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 inherit from `Pizza` and have its own constructor:

   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

   Notes:

   i. The constructor of a specific type of pizza (e.g. `HamdAndCheesePizza`) should only take the necessary toppings (i.e. ham and cheese, not pepperoni) as parameters.

   ii. Add an instance variable `name` to class `Pizza` which is initialised to the *actual class name* inside the constructor.

   iii. Add a `Pizza.getName()` method to return the pizza name.

   iv. You should change `Pizza.getDescription()` to protected and have it overridden by the subclasses.

   v. You should also add protected getter methods for the topping instance variables so that the subclasses can access them.

   vi. You should define the constructor of `Pizza` to take all three types of topping (ham, cheese, pepperoni) and let the constructors of its subclasses (`HamAndCheesePizza`, `PepperoniPizza`, `TropicalPizza`) call `Pizza`'s constructor using the `super` keyword. Try to re-use as much code as you can.

2. 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.

   Notes: You will need to define `Topping` with the `static` keyword.

3. In the `tutes.oop1` package, the `PizzaOrder` class used three instance variables to store pizzas and three setter methods (`setPizza1`, `setPizza2`, `setPizza3`) to add pizzas to the pizza order. This design is not scalable (it does not work with a pizza order of more than 3 pizzas) and cumbersome (it requires 3 methods for adding pizzas). To overcome these design issues, make `PizzaOrder` store pizzas in an array instead and implement the `addPizza(Pizza)` method in `PizzaOrder` to add a pizza to the pizza order. You should make other appropriate changes to the `PizzaOrder` class to realize the above design idea.

4. Make `Pizza` implement the `Comparable` interface to compare two `Pizza` objects based on cost

5. 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
6. Change the `PizzaDemo` class to work with the new design and the sorting function.
7. **[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