1. Sorting algorithms

In the first version of the solution (`solution1.py`), I've simply followed the classic Strategy pattern. As you can see, there's an abstract `SortStrategy` class that defines the `sort` method. The `sort` method takes a list of items (integers in this example) and returns a sorted list of items. The `sort` method is implemented by the concrete strategies, such as `BubbleSortStrategy` and `QuickSortStrategy`.

The `SortingContext` class is the context class. It takes a `SortStrategy` object as an argument to its constructor (I used a dataclass to achieve this). The `SortingContext` class has a `sort_data` method that simply calls the `sort` method of the `SortStrategy` object that was passed to the constructor.

Alternatively, you can follow a more functional approach. You can find an example of this in `solution1_fun.py`. In this version, I've used functions instead of classes. The `sort_data` function takes a list of integers and an algorithm as arguments. The only thing this function does is retrieve the appropriate sorting function from a dictionary and call it with the list of items as an argument.

You can simplify this even further by passing the actual sorting function as an argument. In that case, `sort_data` would become trivial. It would simply call the sorting function and return the result.

2. Shopping carts

You can find the solution for this exercise in `solution2.py`. I've used the classic Strategy pattern to implement the different shipping methods. The `ShippingStrategy` class defines the `calculate_shipping_cost` method. This method takes a list of items as an argument and returns the shipping cost for those items. The `calculate_shipping_cost` method is implemented by the concrete strategies, such as `StandardShippingStrategy` and `ExpressShippingStrategy`.

The `ShoppingCart` class is the context class. This class has a `calculate_total_cost` method that computes the items cost and then calls the `calculate_shipping_cost` method of the `ShippingStrategy` object that was passed as an argument. The method then returns the sum of the items cost and the shipping cost.