A value proposition

Your friend Michael has just been given a gift certificate to use at the latest consolidated merchandise website, *SellU*. The distinctive feature of *SellU* is that it does not tell you a price until you put together a complete shopping cart. This is because it is really just forwarding your order to various other suppliers, and it tries to make gains by combining shipping, using discounts etc.

Michael's experimentation with the site has revealed that its prices are consistent in the sense that if you add items to a shopping cart then the price always increases, and if you remove items it always decreases, but except for that single constraint there seems to be no observable rhyme or reason for the total prices announced.

Having a certain amount to spend, and a list of items which he might buy, Michael wants to optimise his purchase. Each item for him has a certain intrinsic value, so he wants to find a shopping cart that he can afford such that the value (to him) of the items it contains is as large as possible (Michael, somewhat more rational than the site, thinks that the value of a collection of items is equal to the sum of their individual values). He can buy at most one of each type of item (that's a condition of the gift card). Hours of fiddling about later, he wonders if there is any good way to find his perfect bundle of items.

Task

For reasons that will become apparent this task must be completed in Java. Two interfaces `SiteInfo` and `CustomerInfo` and a class `BargainFinder` are provided. You must implement the `shoppingList()` method of `BargainFinder` so that it returns the "best" shopping list for the site, customer and budget provided. You may assume that whatever implementation of `SiteInfo` is provided to your client program meets the criteria laid out above (more items cost more, fewer items cost less). You may add additional classes if need be, but please don't change the name of the `BargainFinder` class.

(Group 2)