

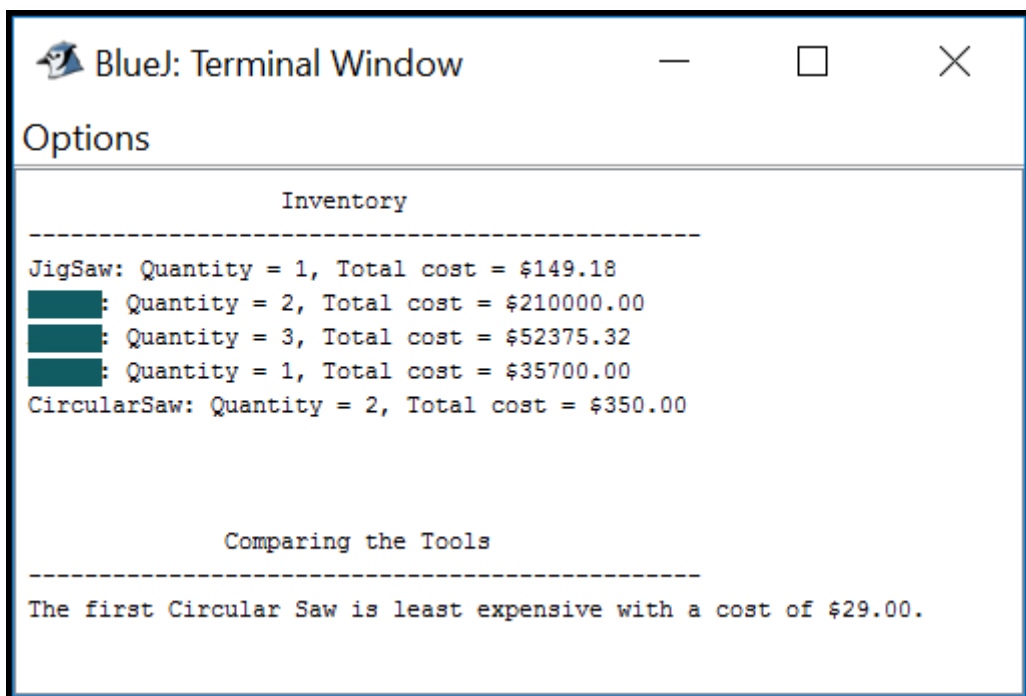
15.05 Assignment Instructions

Instructions: For this assignment, create an inventory project utilizing abstract classes, interfaces, and concrete classes.

1. Create a new project called 15.05 Assignment in the Mod15 Assignment folder.
2. As you work through this project, create new classes in the new project.
3. Read the entire description of the project and classes needed. Take time to plan the project and then start coding.
4. Create an interface named `Product`.
 - a. All products will need methods for getting and returning a name and cost. Add the necessary code to the interface for these two methods. Use appropriate return types.
5. Create abstract class for your first type of product that implements the interface. A suggestion of possible products may be vehicles.
 - a. Include instance variables for the name and cost.
 - b. Establish a constructor to initialize the name and cost.
 - c. Define the methods from the implementation class.
 - d. Include an abstract method that should be overridden by subclasses. An example, if the abstract class is for vehicles, the abstract method may be drive.
6. Create at least two concrete classes that extend the previous abstract class.
 - a. Include a constructor.
 - b. Establish any needed instance variables.
 - c. Define any abstract methods.
7. Create second product class that will implement the `Product` interface and `Comparable<T>`. A suggestion might be a tool.
 - a. Include instance variables for the name and cost.
 - b. Establish a constructor to initialize the name and cost.
 - c. Define the methods from the implementation class.
 - d. Add a `compareTo` method that compares objects of this second type of product based upon cost.
8. Create a client class to manage your inventory of products:

- a. Create instances of each type of product you've defined. Your inventory should contain multiples of some items. The multiples will have the same name but may have different costs.
- b. Organize all of your products in an array list.
- c. Create a static method to take inventory. This method, when passed the name of a product, will go through the list and look for all items of the same name. As it does, a count for the quantity and total cost is maintained. Finally, it displays output stating the item name, total quantity, and total cost. See sample output below.
- d. Test the `compareTo` method established in the second product class. Create two instances of this product. (You can use previously declared ones if you like.) Use the `compareTo` method and based on the results, display a message communicating how their costs compare.

Expected Output: This is a sample of the expected output. The details will vary based on design choices you make while completing the project.



```
BlueJ: Terminal Window
Options
-----
Inventory
-----
JigSaw: Quantity = 1, Total cost = $149.18
[REDACTED]: Quantity = 2, Total cost = $210000.00
[REDACTED]: Quantity = 3, Total cost = $52375.32
[REDACTED]: Quantity = 1, Total cost = $35700.00
CircularSaw: Quantity = 2, Total cost = $350.00

Comparing the Tools
-----
The first Circular Saw is least expensive with a cost of $29.00.
```



