Project 4: Character Inventory

NOTE: *If you turn in a solution to this problem based on something from a pay-for-answers website such as Chegg, CourseHero, etc., you will receive an F in this course (not just on this assignment). Start early and seek help from a lab assistant, the help room, or your professor if you need it. If you don't have time to do that, it is better to get a low grade on the project than to cheat and fail the course.*

This project asks you to develop an Item class, a Character class, and a driver program to test the two.

The **Item** class should have fields for the item's name and its value, along with a constructor and getter/setter methods for those fields. In addition, items should be sortable in descending order by value (i.e from largest value to smallest). The string representation of this class (i.e. what you see when you print it out using System.out.println) should be of the form `name (value)`.

The **Character** class should have fields for the character's name, the number of credits (i.e. money) the character has, and the list of the Item objects the character possesses. Characters should be sortable lexicographically by name. The string representation of this class should be of the form `name (credits)`. The class should also support the operations listed below.

`public void addItem(String itemName, int itemValue)`

Preconditions: none

Return value: none

Postconditions: A new item with the name itemName and the value itemValue has been added to this character's item list. It is ok to have multiple items with the same name and/or value in the list.

`public boolean dropItem(String itemName)`

Preconditions: the character's item list contains one or more items named itemName

Return value: true if the precondition was met, false otherwise

Postconditions: no change if the preconditions were not met; otherwise the character contains one less item named itemName in their list than before the method was run

```
public boolean sellItemToVendor(String itemName)
```

Preconditions: the character's item list contains one or more items named itemName

Return value: true if the precondition was met, false otherwise

Postconditions: no change if the preconditions were not met; otherwise the character contains one less item named itemName in their list than before the method was run and the character's credits are increase by itemValue

```
public boolean sellItemToCharacter(String itemName, Character buyer)
```

Preconditions: this character's item list contains at least one item named itemName with a value less than or equal to the buyer's credits

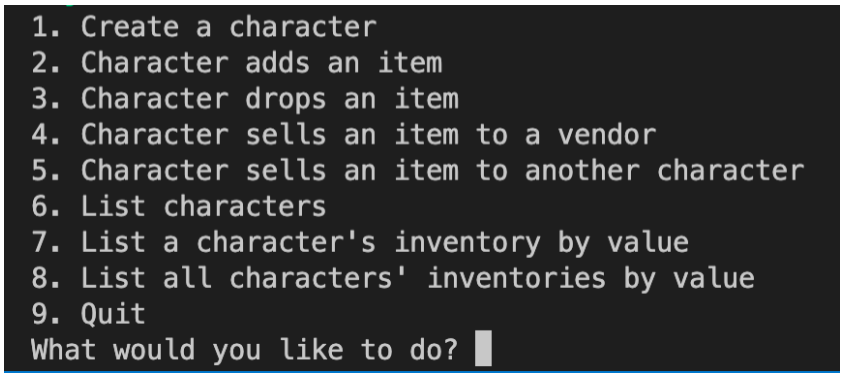
Return value: true if the precondition was met, false otherwise

Postconditions: no change if the preconditions were not met; otherwise this character's item list contains one less item named itemName than before the method was run, the buying character's item list contains one more item named itemName (with the same itemValue as the item removed from this character) than before the method was run, this character's credits are increased by itemValue, and the buying character's credits are decreased by itemValue.

NOTE: Your Item and Character classes can have any additional methods you need/want to implement the required functionality, but they are not allowed to read anything in from the user or print anything out – that must be done through the driver, as described below.

The **driver** class (where the main method is), should be in a file called either Main.java or Project4.java. It should display the following menu, sanity check the user's choice, prompt the user for any required information, and then display the relevant output (as described below). If a menu item requires a valid character name, such as "Character adds an item" or "List a character's inventory by value" and the user enters a character name that does not

exist, display an error message stating that and re-display the menu.



Characters must have unique names, so if the user attempts to name a new character the same thing as an existing character, re-prompt them for a new name. (Item names do not have to be unique, even for multiple items belonging to the same character.)

Relevant output for each menu option:

1. "<characterName> added"
2. "<characterName> has acquired <itemName>"
3. Either "<characterName> has dropped <itemName>" or "<characterName> could not drop <itemName>"
4. Either "<characterName> has sold <itemName> to a vendor" or "<characterName> could not sell <itemName> to a vendor"
5. Either "<characterName> has sold <itemName> to <otherCharacterName>" or "<characterName> could not sell <itemName> to <otherCharacterName>"
6. A list of all current characters, one per line, in the form "<name> (<credits>)"
7. A list of the chosen character's inventory, one item per line and sorted in descending order by value, in the form "<itemName> (<itemValue>)"
8. A list of all characters' item inventories, where each character's inventory starts with that character's name and credits and is followed by the their inventory, one item per line and sorted in descending order by value, in the form "<itemName> (<itemValue>)"

Example:

```
1. Create a character
2. Character adds an item
3. Character drops an item
4. Character sells an item to a vendor
5. Character sells an item to another character
6. List characters
7. List a character's inventory by value
8. List all characters' inventories by value
9. Quit
What would you like to do? 1
What is the character's name? Alice the Awesome
How many credits does the character have? 100
Alice the Awesome added

1. Create a character
2. Character adds an item
3. Character drops an item
4. Character sells an item to a vendor
5. Character sells an item to another character
6. List characters
7. List a character's inventory by value
8. List all characters' inventories by value
9. Quit
What would you like to do? 1
What is the character's name? Alice the Awesome
Error: a character with that name already exists.

1. Create a character
2. Character adds an item
3. Character drops an item
4. Character sells an item to a vendor
5. Character sells an item to another character
6. List characters
7. List a character's inventory by value
8. List all characters' inventories by value
9. Quit
What would you like to do? 1
What is the character's name? Bob the Brave
How many credits does the character have? 1000
Bob the Brave added

1. Create a character
2. Character adds an item
3. Character drops an item
4. Character sells an item to a vendor
5. Character sells an item to another character
6. List characters
7. List a character's inventory by value
8. List all characters' inventories by value
9. Quit
What would you like to do? 6
Alice the Awesome (100)
Bob the Brave (1000)

1. Create a character
2. Character adds an item
3. Character drops an item
4. Character sells an item to a vendor
5. Character sells an item to another character
6. List characters
```

```
7. List a character's inventory by value
8. List all characters' inventories by value
9. Quit
What would you like to do? 2
Which character is adding the item? Fred
Error: no character with that name exists.

1. Create a character
2. Character adds an item
3. Character drops an item
4. Character sells an item to a vendor
5. Character sells an item to another character
6. List characters
7. List a character's inventory by value
8. List all characters' inventories by value
9. Quit
What would you like to do? 2
Which character is adding the item? Alice the Awesome
What is the name of the item? Sword of Swatting
What is the item's value? 80
Alice the Awesome has acquired Sword of Swatting

1. Create a character
2. Character adds an item
3. Character drops an item
4. Character sells an item to a vendor
5. Character sells an item to another character
6. List characters
7. List a character's inventory by value
8. List all characters' inventories by value
9. Quit
What would you like to do? 2
Which character is adding the item? Bob the Brave
What is the name of the item? Sword of Smiting
What is the item's value? 900
Bob the Brave has acquired Sword of Smiting

1. Create a character
2. Character adds an item
3. Character drops an item
4. Character sells an item to a vendor
5. Character sells an item to another character
6. List characters
7. List a character's inventory by value
8. List all characters' inventories by value
9. Quit
What would you like to do? 5
Which character is selling the item? Bob the Brave
Which character is buying the item? Alice the Awesome
What is the name of the item? Sword of Smiting
Bob the Brave could not sell Sword of Smiting to Alice the Awesome

1. Create a character
2. Character adds an item
3. Character drops an item
4. Character sells an item to a vendor
5. Character sells an item to another character
6. List characters
7. List a character's inventory by value
8. List all characters' inventories by value
9. Quit
What would you like to do? 5
Which character is selling the item? Alice the Awesome
Which character is buying the item? Bob the Brave
```

```
What is the name of the item? Sword of Swatting
Alice the Awesome has sold Sword of Swatting to Bob the Brave

1. Create a character
2. Character adds an item
3. Character drops an item
4. Character sells an item to a vendor
5. Character sells an item to another character
6. List characters
7. List a character's inventory by value
8. List all characters' inventories by value
9. Quit
What would you like to do? 8

Alice the Awesome (180)

Bob the Brave (920)
Sword of Smiting (900)
Sword of Swatting (80)
```

Rubric:

Note: Code that does not compile will receive a zero.

[10 pts] The Item class has the required fields and string representation, and it is sortable in descending order by value. The class does do any console I/O.

[10 pts] The Character class has the required fields and string representation, and it is sortable lexicographically by name. The class does not do any console I/O.

[10 pts each x 4 = 40 pts] The Character class implements each of the required methods (addItem, dropItem, sellItemToVendor, sellItemToCharacter, as described.

[10 pts] The driver program displays the menu, gets and sanity checks the required input, performs the requested action, and displays the result.

[10 pts] The driver program prevents two characters from having the same name.

[20 pts] The program is clearly organized, commented, and follows standard coding practices, including using descriptive/meaningful variable and class names.

Note that there are more points related to program style, organization, and commenting this time.