LAB ALT-10

Task: Hold details about cars in a Car class.

What will the application do?

- Display a list of at least 6 cars (at least 3 new and 3 used).
- Let the user select one of the cars.
- Re-display the information for car selected.
- Ask the user if they want to buy the car. If they do, remove the car.
- Re-display the list.

Build Specifications

- Create a class named Car to store the data about this car. This class should contain:
    a. Data members for car details
        1. A String for the make
        2. A String for the model
        3. An int for the year
        4. A double for the price
    b. A no-arguments constructor that sets data members to default values (blanks or your choice)
    c. A constructor with four arguments matching the order above
    d. Getters and setters for all data members
    e. A toString() method returning a formatted string with the car details.
- Create a subclass of Car named UsedCar. UsedCar has additionally:
    a. Data member: A double for mileage.
    b. Constructor: Takes five arguments (same order as constructor from last lab with the mileage last).
    c. toString: overrides Car's toString() to include (Used) and the mileage.
- For this lab, you can instantiate your Car and UsedCar instances in the code—you don't need to take input for them.
- Make your menu system as you'd like (for choosing which car from the list—numbers, letters, any other way)
- Store all Car and UsedCar instances together in the same ArrayList. (This time, don't use an array.)
- Create a main program that gets the user input, creates multiple instances of a Car object, and displays their information.

Hints:

- Use the right visibility modifiers!
- Make sure to match the signature of toString() from Object.

- You can just use \t tab escape characters to line things up, or get fancier with some of the string formatting stuff we've talked about.
- Let polymorphism work for you.
- Remember casting and instanceOf.

Extended Challenge:

- Modify or create a class named Validator with static methods to validate the data in this application.

Console Preview:

```
1. Nikolai    Model S     2017        $54,999.90
2. Fourd      Escapade    2017        $31,999.90
3. Chewie     Vette       2017        $44,989.95
4. Hyonda     Prior       2015        $14,795.50 (Used) 35,987.6 miles
5. GC         Chirpus     2013        $8,500.00 (Used) 12,345.0 miles
6. GC         Witherell   2016        $14,450.00 (Used) 3,500.3 miles
7. Quit

Which car would you like?  {6}
GC            Witherell   2016        $14,450.00 (Used) 3,500.3 miles
Would you like to buy this car? {y}
Excellent!  Our finance department will be in touch shortly.

1. Nikolai    Model S     2017        $54,999.90
2. Fourd      Escapade    2017        $31,999.90
3. Chewie     Vette       2017        $44,989.95
4. Hyonda     Prior       2015        $14,795.50 (Used) 35,987.6 miles
5. GC         Chirpus     2013        $8,500.00 (Used) 12,345.0 miles
6. Quit

Which car would you like?  {6}
Have a great day!
```

Note: The output above involves formatting columns and dollar figures, one of the extended challenges.

LAB ALT-11

Task: Create a CarLot class to store your new and used cars. This is a more open-ended and design-focused lab.

What will the application do?

- Display a list of at least 6 cars (at least 3 new and 3 used).
- Let the user select one of the cars.
- Display the price and (if a used car) mileage for the car selected.
- Ask if they want to select again and repeat if they do.

Build Specifications

1. Pair with a classmate for this lab. Choose one person's Car and UsedCar to use, or merge your two together.
2. Discuss with your partner how a CarLot class could best store information. In what cases would each of these make more sense?
    - A two-dimensional array of Cars
    - An ArrayList of Cars
    - Any other option?
3. Plan out your methods for CarLot. Decide if any other information is needed in the CarLot class or in Car/UsedCar.
4. Write a CarLotApp class which instantiates and puts cars in your CarLot class. It should invoke CarLot methods to let a user:
    - List all cars.
    - Add a car.
    - Remove a car.
    - Look up a car in a given position.
    - Replace a car in a given position.

Extended Challenge:
Provide search features:

- View all cars of an entered make.
- View all cars of an entered year.
- View all cars of an entered price or less.
- View only used cars or view only new cars.

Your output will vary based on decisions you make with your partner.