Name _____
Date _____

AP Computer Science A
ArrayList Programs

Directions: You can include the three programs in the same class and main method. However, the third program will require a separate class to create Word objects.

Hungry Herman: Herman is hungry. You are to write a program that will:

1. Ask the user for 4 items of food to put on Herman's plate. Each of these items should be added to Herman's "plate" (ArrayList) using the add method.
2. Herman doesn't like everything that was put there, so he removes the 3rd item added.
3. Herman decides he doesn't want the last item added to his plate, but instead wants "ice cream". Change the last item added to "ice cream".
4. Herman's mom asks if Herman got some "green beans". Did he? Use the contains method to determine if Herman has "green beans" and print to the screen whether he does or does not.
5. Herman eats his food in order so he would like to add a salad to the front of his plate.
6. Herman's mom then asks where his salad is located. Print the location of his salad.
7. Munch, munch. Herman eats his food. Go through Herman's plate and munch all his food by using the remove method. Print out the ArrayList after each food item is eaten.

Grocery List: You are tasked with writing a program that will simulate creating and crossing off a grocery list.

1. The user should be provided with a menu system that allows them to:
   (1) Add an item to the list.
   (2) Remove an item on the list (if it is on the list – otherwise inform them that the item was not on the list).
   (3) Replace an item on the list.
   (4) Display the list and the total number of items in the list.
   (5) Quit
2. The program should not end until the user decides to quit.

Dictionary: You are tasked with writing a program that allows the user to create their own dictionary. The dictionary will be comprised of words and their definitions.

1. Create a Word class that has two String instance variables – word and definition. The class should have one constructor and five methods.
    a. The constructor should take two String parameters that you can assign to word and definition.
    b. Two of the methods should be setter methods (one String parameter each).
    c. Two of the methods should be getter methods (no parameters).
    d. One of the methods should be toString. This method should return a String formatted for output so that the ArrayList can print the Word object directly.
2. Your ArrayList will hold Word objects.
3. The user should be provided with a menu system that allows them to:
   (1) Add a new word and its definition to the dictionary.
   (2) Remove a word and its definition from the dictionary (if it is in the dictionary – otherwise inform them that the word was not in the dictionary).
   (3) Change a definition to a word in the dictionary.
   (4) Get the definition of a word, or be informed that the word is not in the dictionary.
   (5) Display all the words in the dictionary and the total count of words in the dictionary.
   (6) Quit
4. The program should not end until the user decides to quit.

Bonus:

- Store the words in the dictionary in alphabetical order (don't forget to keep their definitions matched to the correct words!)
- Allow the user to "backwards" search the dictionary. Allow the user to enter a word that might be found in the definition and then provide the user with all words (and definitions) that might be the word they are looking for.

Name _____
Date _____

AP Computer Science A
Yahtzee Lab

In the game of Yahtzee, players put dice in a cup, shake it to roll the dice, then empty the dice onto the table. Your task is to build a Die class, a DiceCup class, and a client program DiceGame to test DiceCup. Here are the program specifications:

1. Die Class
   - Fields:
     - dots: int variable that stores the number of dots on the die
   - Methods:
     - roll: use Math.random() to assign a random integer between 1 and 6.
     - getNumDots: returns the number of dots on the die.

2. Dice Cup
   - Fields:
     - dice: an array of Die objects
     - values: an int array of face values. The index represents the face values, 0-6. Each element in this array is a count of the number of dice that have this face value. For example, if three of the dice have a face value of 1, then values [1] = 3.
   - Constructor:
     - One parameter: the number of dice in the cup.
     - The constructor must instantiate and build the arrays. Call the rollDice method to initialize the values array.
   - Methods:
     - rollDice: clear the value array. Roll each die. Update the face values array after each dice roll.
     - getValues: return the values array.
     - toString: return a string representation of the dice array. (Hint: call getNumDots for each Die object.)

3. DiceGame
   - This client program will test all methods of your DiceCup class. You may put the code for this program in one main method.
   - Main Method
     - Declare an ArrayList of DiceCup objects.
     - Using a while loop, ask the user to enter a number of dice or 0 to end the game. Store each turn in the ArrayList.
     - Declare and instantiate a new dice cup of the given size.
     - Roll the dice.
     - Print the dice cup.
     - Print the counts of each face value.
     - If all the dice values are the same print “Yahtzee!”
     - Brenda wants to know whether she “Yahtzee!-ed” on her second turn. Write code that lets her know what happened on her second turn.