```
//
// Copyright (c) 2023 Promineo Tech
// Author:  Promineo Tech Academic Team
// Subject:  StringBuilders, Lists, Sets, & Maps
// Java Week 04 Lab
//
package week4labs;

import java.util.List;

public class Week04StringBuilderListSetMapLab {

        public static void main(String[] args) {

                // 1. Why would we use a StringBuilder instead of a
String?
                //              a. Instantiate a new StringBuilder
                //              b. Append the characters 0 through 9
to it separated by dashes
                //                              Note:  make sure no
dash appears at the end of the StringBuilder
                System.out.println("\nQuestion 1:");
                StringBuilder sb = new StringBuilder();

                for (int i = 0; i < 10; i++) {
                        sb.append(i);
                        if (i != 9) {
                                sb.append("-");
                        }
                }

                System.out.println(sb.toString());



                // 2. List of String:
                //              a. Create a list of Strings
                //              b. Add 5 Strings to it, each with a
different length

                System.out.println("\nQuestion 2:");
                       List<String> strings = new ArrayList<String>();

                strings.add("Mon");
                strings.add("Dad");
                strings.add("Sister");
                strings.add("Brother");
                strings.add("Cat");

                for (String string: strings) {
```

```
				System.out.println(string);
			}

			// 3. Write and test a method that takes a list of 
strings
			//			and returns the shortest 
string


					System.out.println("\nQuestion 3:");
			System.out.println(findShortestString(strings));



			// 4. Write and test a method that takes a list of 
strings
			//				and returns the list 
with the first and last element switched


			System.out.println("\nQuestion 4:");

			List<String> swapped = swapFirstAndLast(strings);

			for (String string : swapped) {

				System.out.println(string);
			}

			// 5. Write and test a method that takes a list of 
strings
			//			and returns a string with all 
the list elements concatenated to each other,
			//			separated by a comma

			System.out.println("\nQuestion 5:");

			System.out.println(combineStrings(swapped));


			// 6. Write and test a method that takes a list of 
strings and a string
			//			and returns a new list with 
all strings from the original list
			//			containing the second string 
parameter (i.e. like a search method)

			System.out.println("\nQuestion 6:");

			System.out.println("------------");
```

```
		List<String> searchResults = search(strings, "am");

		for (String result : searchResults) {

			System.out.println(result);
		}

		// 7. Write and test a method that takes a list of
integers
		//			and returns a
List<List<Integer>> with the following conditions specified
		//			for the return value:
		//		a. The first List in the returned
value contains any number from the input list
		//			that is divisible by 2
		//		b. The second List contains values
from the input list that are divisible by 3
		//		c. The third containing values
divisible by 5, and
		//		d. The fourth all numbers from the
input List not divisible by 2, 3, or 5
	System.out.println("\nQuestion 7:");
		List<Integer> numbers = Arrays.asList(0, 9, 8, 7, 6,
5, 4, 3, 2, 1, 10, 11, 12, 100, 115);

		List<List<Integer>> sortedNumbers =
sortDivisibleNumbers(numbers);
		int count = 1;
		for (List<Integer> list : sortedNumbers) {
			for (Integer number : list) {
				System.out.println(number);
			}
			if (count < sortedNumbers.size()) {
				System.out.println("Next list
----------");
			}
			count++;
		}

		// 8. Write and test a method that takes a list of
strings
		//			and returns a list of integers
that contains the length of each string

		System.out.println("\nQuestion 8:");

		List<Integer> stringsLengths =
calculateStringLengths(strings);
```

```
		for (Integer i : stringsLengths) {

		}
			System.out.println(i);
		}


		// 9. Create a set of strings and add 5 values
		System.out.println("\nQuestion 9:");


		Set<String> set = new HashSet<String>();

		set.add("Dolly");
		set.add("Wood");
		set.add("Parton");
		set.add("Music");
		set.add("Guitar");

		for (String word : set) {
			System.out.println(word);
		}



		// 10. Write and test a method that takes a set of
strings and a character
		//			and returns a set of strings
consisting of all the strings in the
		//			input set that start with the
character parameter.

		System.out.println("\nQuestion 10:");

		Set<String> startsWithJ = findStartWith(set, 'J');

		for (String word : startsWithJ) {
			System.out.println(word);
		}


		// 11. Write and test a method that takes a set of
strings
		//			and returns a list of the same
strings

		System.out.println("\nQuestion 11:");

		List<String> resultList = convertSetToList(set);
```

```
                for (String listString : resultList) {

                        System.out.println(listString);



                // 12. Write and test a method that takes a set of
integers
                //                      and returns a new set of
integers containing only even numbers
                //                      from the original set


                        System.out.println("\nQuestion 12:");
                        Set<Integer> integerSet = new
HashSet<Integer>();
                        integerSet.add(9);
                        integerSet.add(4);
                        integerSet.add(6);
                        integerSet.add(99);

                        Set<Integer> extractedEvens =
extractEvens(integerSet);
                        for (Integer number : extractedEvens) {
                                System.out.println(number);
                        }


                // 13. Create a map of string and string and add 3
items to it where the key of each
                //                      is a word and the value is the
definition of the word
                        System.out.println("\nQuestion 13:");
                        Map<String, String> dictionary = new
HashMap<String, String>();

                        dictionary.put("DollyParton", "An actual on
earth Angel");


                        dictionary.put("Clouds", "Usually white");

                        dictionary.put("Curtains", "Cloth that hangs
in front of windows");

                        System.out.println(dictionary);
```

```
		// 14. Write and test a method that takes a
Map<String, String> and a string
		//			and returns the value for a
key in the map that matches the
		//			string parameter (i.e. like a
language dictionary lookup)
			System.out.println("\nQuestion 14:");

			String value = lookupValue(dictionary,
"Rock");

			System.out.println("Dictionary Result for
'Rock': " + value);


		// 15. Write and test a method that takes a
List<String>
		//			and returns a Map<Character,
Integer> containing a count of
		//			all the strings that start
with a given character

			System.out.println("\nQuestion 15:");

			Map<Character, Integer> counts =
countStartingLetters(resultList);

			for (Character character : counts.keySet()) {
				System.out.println(character + " - " +
counts.get(character));
			}
	}


	// Method 15:
		public static Map<Character, Integer>
countStartingLetters(List<String> list) {
			Map<Character, Integer> results = new
HashMap<Character, Integer>();

			for (String string : list) {

				char j = string.charAt(0);

				if (results.get(j) == null) {
					results.put(j, 1);

				} else {
					results.put(j, results.get(j)
+ 1);
```

```
                                }
                        }
                        return results;
                }


        // Method 14:
                public static String lookupValue(Map<String, String>
map, String string) {
                        for (String key : map.keySet()) {

                                if (key.equals(string)) {

                                        return map.get(key);
                                }
                        }
                        return "";
                }





        // Method 12:

                public static Set<Integer> extractEvens(Set<Integer>
set) {
                        Set<Integer> results = new HashSet<Integer>();
                        for (Integer number : set) {
                                if (number % 2 == 0) {
                                        results.add(number);
                                }
                        }
                        return results;
                }

        // Method 11:
                public static List<String>
convertSetToList(Set<String> set) {
                        List<String> results = new
ArrayList<String>();


                        for (String string : set) {
                                results.add(string);
                        }


                        return results;
                }
        // Method 10:

                public static Set<String> findStartWith(Set<String>
set, char j) {

                        Set<String> results = new HashSet<String>();


                        for (String string : set) {

                                if (string.charAt(0) == j) {
                                        results.add(string);
                                }

                        }

                        return results;
                }


        // Method 8:

                public static List<Integer>
calculateStringLengths(List<String> list) {

                        List<Integer> lengths = new
ArrayList<Integer>();

                        for (String string : list) {

                                lengths.add(string.length());
                        }
                        return lengths;
                }


        // Method 7:

                public static List<List<Integer>>
sortDivisibleNumbers(List<Integer> list) {

                        List<List<Integer>> results = new
ArrayList<List<Integer>>();

                        results.add(new ArrayList<Integer>());
                        results.add(new ArrayList<Integer>());
                        results.add(new ArrayList<Integer>());
                        results.add(new ArrayList<Integer>());
                        for (Integer number : list) {
                        if (number % 2 == 0) {
                                results.get(0).add(number);

                        }
                                if (number % 3 == 0) {
                                        results.get(1).add(number);

                                }
                                if (number % 5 == 0) {

                                        results.get(2).add(number);
                                }
                                if (number % 2 != 0 && number % 3 != 0
&& number % 5 != 0) {

                                        results.get(3).add(number);
                                }
                        }

                        return results;
                }



        // Method 6:

                public static List<String> search(List<String> list,
String query)
                {
                        List<String> results = new
ArrayList<String>();
                        for (String string : list) {

                                if (string.contains(query)) {

                                        results.add(string);
                                }
                        }
                        return results;
                }

        // Method 5:
        // Method 5:
        public static String combineStrings(List<String> strings) {
                StringBuilder result = new StringBuilder();
                int count = 1;

 (String string : strings) {
```

```
                        result.append(string);


                        if (count < strings.size()) {
                                result.append(", ");
                        }

                        count++;
                }
                return result.toString();
        }


        // Method 4:
        public static List<String> swapFirstAndLast(List<String> list)
{
                String temp = list.get(0);
                list.set(0, list.get(list.size() - 1));
                \
                list.set(list.size() - 1, temp);


                return list;
        }



        // Method 3:

        public static String findShortestString(List<String> list) {
                String shortest = list.get(0);

                for (String string : list) {

                        if (string.length() < shortest.length()) {
                                shortest = string;
                        }
                }
                return shortest;
        }


}
```