```java
package week04;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;



public class Week04StringBuilderListSetMapLab {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		// 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
				StringBuilder sb = new StringBuilder();
				String dash = "-";
				sb.append("1");
				sb.append(dash);
				sb.append("2");
				sb.append(dash);
				sb.append("3");
				sb.append(dash);
				sb.append("4");
				sb.append(dash);
				sb.append("5");
				sb.append(dash);
				sb.append("6");
				sb.append(dash);
				sb.append("7");
				sb.append(dash);
				sb.append("8");
				sb.append(dash);
				sb.append("9");
				System.out.println(sb);

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

				ArrayList<String> myInstruments = new ArrayList<>();
				myInstruments.add("Ibanez Electric Guitar");
				myInstruments.add("Alesis Midi Controller");
				myInstruments.add("Tape Organ");
				myInstruments.add("Fender Acoustic Guitar");
				myInstruments.add("Frogslap");

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

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


      System.out.println(shortestInstrument(myInstruments));

                        // 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(firstLastSwitched(myInstruments));

                        // 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(instrumentList(myInstruments));

                        // 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)
                        String term = "Guitar";
                        System.out.println("Which guitars do I have?");
                        System.out.println(findGuitars(myInstruments, term));

                        // 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
                        ArrayList<Integer> testIntegers = new
ArrayList<Integer>();
                        testIntegers.add(1);
                        testIntegers.add(8);
                        testIntegers.add(6);
                        testIntegers.add(13);
                        testIntegers.add(4);
                        testIntegers.add(9);
                        testIntegers.add(21);
                        testIntegers.add(5);
                        testIntegers.add(19);
                        testIntegers.add(12);
```

```
                        System.out.println(fourLists(testIntegers));


                        // 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

                        ArrayList<String> names = new ArrayList<String>();
                        names.add("Ben");
                        names.add("Kai");
                        names.add("Althea");
                        names.add("Camas");
                        names.add("Piper");
                        names.add("Indy");

                        System.out.println(nameLength(names));

                        // 9. Create a set of strings and add 5 values
                        Set<String> pasta = new HashSet<String>();
                        pasta.add("mostaccioli");
                        pasta.add("radiatori");
                        pasta.add("manicotti");
                        pasta.add("orzo");
                        pasta.add("fettuccine");

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



                        // 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.
                        char firstLetter = 'm';
                        System.out.println(startsWithChar(pasta,
firstLetter));
                        System.out.println("-------------------------");


                        // 11. Write and test a method that takes a set of
strings
                        //                and returns a list of the same
strings
                        System.out.println(turnsToList(pasta));
                        System.out.println("-------------------------");



                        // 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
                        Set<Integer> bunchaNumbers = new HashSet<Integer>();
                        bunchaNumbers.add(1);
                        bunchaNumbers.add(2);
                        bunchaNumbers.add(3);
                        bunchaNumbers.add(4);
                        bunchaNumbers.add(5);
                        bunchaNumbers.add(6);
                        bunchaNumbers.add(7);
                        bunchaNumbers.add(8);
                        bunchaNumbers.add(9);
                        bunchaNumbers.add(10);

                        System.out.println(evensSet(bunchaNumbers));
                        System.out.println("-------------------------");


                        // 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
                        HashMap<String, String> wordsAndDefs = new
HashMap<String, String>();

                        wordsAndDefs.put("vestibular", "relating to a
vestibule");
                        wordsAndDefs.put("specific", "clearly defined or
identified");
                        wordsAndDefs.put("cantankerous", "bad-tempered,
argumentative, and uncooperative");

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


                        // 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)
                        String mapWord = "specific";
                        System.out.println(dictionaryLookup(wordsAndDefs,
mapWord));
                        System.out.println("-------------------------");

                        // 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
                        ArrayList<String> unitedStates = new
ArrayList<String>();
                        unitedStates.add("Alabama");
```

```
        unitedStates.add("Alaska");
        unitedStates.add("Arizona");
        unitedStates.add("Arkansas");
        unitedStates.add("California");
        unitedStates.add("Colorado");
        unitedStates.add("Conneticut");
        unitedStates.add("Delaware");
        unitedStates.add("Florida");
        unitedStates.add("Georgia");
        unitedStates.add("Hawai'i");
        unitedStates.add("Idaho");
        unitedStates.add("Illinois");
        unitedStates.add("Indiana");
        unitedStates.add("Iowa");
        unitedStates.add("Kansas");
        unitedStates.add("Kentucky");
        unitedStates.add("Louisiana");
        unitedStates.add("Maine");
        unitedStates.add("Maryland");
        unitedStates.add("Massachusetts");
        unitedStates.add("Michigan");
        unitedStates.add("Minnesota");
        unitedStates.add("Mississippi");
        unitedStates.add("Missouri");
        unitedStates.add("Montana");
        unitedStates.add("Nebraska");
        unitedStates.add("Nevada");
        unitedStates.add("New Hampshire");
        unitedStates.add("New Jersey");
        unitedStates.add("New Mexico");
        unitedStates.add("New York");
        unitedStates.add("North Carolina");
        unitedStates.add("North Dakota");
        unitedStates.add("Ohio");
        unitedStates.add("Oklahoma");
        unitedStates.add("Oregon");
        unitedStates.add("Pennsylvania");
        unitedStates.add("Rhode Island");
        unitedStates.add("South Carolina");
        unitedStates.add("South Dakota");
        unitedStates.add("Tennessee");
        unitedStates.add("Texas");
        unitedStates.add("Utah");
        unitedStates.add("Vermont");
        unitedStates.add("Virginia");
        unitedStates.add("Washington");
        unitedStates.add("West Virginia");
        unitedStates.add("Wisconsin");
        unitedStates.add("Wyoming");

        System.out.println(stateStartsWith(unitedStates));

    }


    // Method 15:
```

```
	public static HashMap<Character, Integer>
stateStartsWith(ArrayList<String> states) {
		System.out.println("Enter a letter to find number of
states that start with that letter:");
		Scanner sc = new Scanner(System.in);
		HashMap<Character, Integer> returnStates = new
HashMap<Character, Integer>();
		char letter = sc.next().charAt(0);
		int count = 0;
		String firstLetter = Character.toString(letter);

		for (String state : states) {
			if ( state.startsWith(firstLetter)) {
				count += 1;
				returnStates.put(letter, count);
			}
		}
		return returnStates;

	}


	// Method 14:
	public static String dictionaryLookup(HashMap<String,
String> map, String word) {
		String test = map.get(word);
				return test;

	}



	// Method 12:
	public static Set<Integer>
evensSet(Set<Integer>bunchaNumbers) {
		Set<Integer> newEvens = new HashSet<Integer>();
		for (int evens : bunchaNumbers) {
			if (evens % 2 == 0) {
				newEvens.add(evens);
			}
		}
		return newEvens;
	}


	// Method 11:
	public static ArrayList<String> turnsToList(Set<String>
pasta) {
		ArrayList<String> newList = new ArrayList<String>();
		for (String tempPasta : pasta) {
			newList.add(tempPasta);
		}
		return newList;
	}


	// Method 10:
```

```
	public static Set<String> startsWithChar(Set<String> pasta,
char first) {
		Set<String> startsWithM = new HashSet<String>();
		String firstLetter = Character.toString(first);
		for (String pastaString : pasta) {
			if (pastaString.startsWith(firstLetter)) {
				startsWithM.add(pastaString);
			}
		}
		return startsWithM;
	}


	// Method 8:
	public static ArrayList<Integer>
nameLength(ArrayList<String> names) {
		ArrayList<Integer> stringLength = new
ArrayList<Integer>();
		for (int i = 0; i < names.size(); i++) {
			stringLength.add(i, names.get(i).length());
		}
		return stringLength;
	}

	// Method 7:
	public static ArrayList<ArrayList<Integer>>
fourLists(ArrayList<Integer> testIntegers) {
		ArrayList<ArrayList<Integer>> arrayList = new
ArrayList<ArrayList<Integer>>();
		ArrayList<Integer> evens = new ArrayList<Integer>();
		ArrayList<Integer> threes = new ArrayList<Integer>();
		ArrayList<Integer> fives = new ArrayList<Integer>();
		ArrayList<Integer> noneOfTheAbove = new
ArrayList<Integer>();

		for (int i = 0; i < testIntegers.size(); i++) {
			if (testIntegers.get(i) % 2 == 0) {
				evens.add(testIntegers.get(i));
			} else if (testIntegers.get(i) % 3 == 0) {
				threes.add(testIntegers.get(i));
			} else if (testIntegers.get(i) % 5 == 0) {
				fives.add(testIntegers.get(i));
			} else {
				noneOfTheAbove.add(testIntegers.get(i));
			}
		}
		arrayList.add(evens);
		arrayList.add(threes);
		arrayList.add(fives);
		arrayList.add(noneOfTheAbove);
		return arrayList;



	}
```

```
		// Method 6:
		public static ArrayList<String>
findGuitars(ArrayList<String> myInstruments, String term) {
			ArrayList<String> guitars = new ArrayList<>();
			for (int i = 0; i < myInstruments.size(); i++) {
				if (myInstruments.get(i).contains(term)) {
					guitars.add(myInstruments.get(i));
				}
			}
			return guitars;
		}


		// Method 5:
		public static String instrumentList(ArrayList<String>
myInstruments) {
			String listOut = "";
			for (int i = 0; i < myInstruments.size(); i++) {
				if (i != myInstruments.size() - 1) {
					listOut += myInstruments.get(i) + ", ";
				} else {
					listOut += myInstruments.get(i);
				}
			}
			return listOut;
		}


		// Method 4:
		public static ArrayList<String>
firstLastSwitched(ArrayList<String> myInstruments) {
			String temp = "";
			temp = myInstruments.get(0);
			myInstruments.set(0,
myInstruments.get(myInstruments.size() - 1));
			myInstruments.set(myInstruments.size() - 1, temp);
			return myInstruments;
		}


		// Method 3:

		public static String shortestInstrument(ArrayList<String>
myInstruments) {
			String shortest = "";
			int length = 9999;
			for (int i = 0; i < myInstruments.size(); i++) {
				if (myInstruments.get(i).length() < length) {
					shortest = myInstruments.get(i);
					length = myInstruments.get(i).length();
				}
			}
			return shortest;
	}

}
```