EASTERN INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTING
AND INFORMATION TECHNOLOGY

Practice Assignment – Quarter 1, 2024-2025
Course Name: OOD
Course Code: CSE 203
Student's Full Name:
Student ID:

Lab 5

1. Word Set

Write an application that reads a line of input from the keyboard, and then displays each unique word that was entered, sorted in ascending order. You can do this by tokenizing the line of input and adding each token to an appropriate `Set` object.

2. `Instructor` Set

Modify the Instructor class by overriding the `hashCode` and `equals` methods. Then write a class that stores several `Instructor` objects in a `HashSet`. The class should be able to display all the instructors in the set, and allow the user to search for an instructor. Students are free to design the methods for displaying and searching. Demonstrate the class in an application.

3. `EmployeeMap` Class

Create an `Employee` class that stores an employee's ID number and name. Then create an `EmployeeMap` class that allows you to add `Employee` objects and look them up by their ID numbers. The `EmployeeMap` class should use a `Map` object to map ID numbers to `Employee` objects. Create an application to demonstrate the classes.

4. Dealing Cards

Write a class named **`Card`**, which can be used to represent a card from a deck of cards. The class should be able to store a card's suit and face value. A card's suit can be one of the following: Hearts, Diamonds, Spades, or Clubs. A card's face value can be Ace, Jack, Queen, King, or a value in the range of two through ten.
Next write a **`Deck`** class. This class constructor should create a list of 52 Card objects, each representing a valid card in a deck of cards. The class should have a shuffle method that randomly shuffles the Card objects in the list. It should also have a deal method that "deals" a card from the deck. It does this by removing the Card object at the beginning of the list and returning a reference to that object.

Next, write **`CardPlayer`** class. This class should keep a list of `Card` objects that have been dealt to it. This represents a hand of cards. A method named `getCard` should accept a reference to a `Card` object, which is added to the list. A method named `showCards` displays the `Card` objects in the list.
Demonstrate these classes in an application that creates a `Deck` object, shuffles the cards it contains, and deals five cards from the `Deck` to a `CardPlayer` object. The `CardPlayer` should then display its cards.

5. **Word Frequency Count**

Write a program that that allows the user to specify a text file, opens the file, and prints a two-column table consisting of all the words in the file together with the number of times that each word appears. Words are space-delimited and case-sensitive. The table should list the words in alphabetical order.

6. **`MyList` Class**

Write a generic class named `MyList`, with a type parameter T. The type parameter T should be constrained to an upper bound: the `Number` class. The class should have as a field an `ArrayList` of T. Write a public method named add, which accepts a parameter of type T. When an argument is passed to the method, it is added to the `ArrayList`. Write two other methods, largest and smallest, which return the largest and smallest values in the `ArrayList`.

7. **`MyList` Modification**

Modify the `MyList` class that you wrote for Programming Challenge 6 so the type parameter `T` should accept any type that implements the `Comparable` interface. Test the class in a program that creates one instance of `MyList` to store Integers, and another instance to store Strings.

8. **`TestScores` Class**

Write a class named `TestScores`. The class constructor should accept an array of test scores as its argument. The class should have a method that returns the average of the test scores. If any test score in the array is negative or greater than 100, the class should throw an `IllegalArgumentException`. Demonstrate the class in a program.

9. **`TestScores` Class Custom Exception**

Write an exception class named `InvalidTestScore`. Modify the `TestScores` class you wrote in Programming Challenge 8 so that it throws an `InvalidTestScore` exception if any of the test scores in the array are invalid.

10. **File Encryption Filter**

File encryption is the science of writing the contents of a file in a secret code. Your encryption program should work like a filter, reading the contents of one file, modifying the data into a code, and then writing the coded contents out to a second file. The second file will be a version of the first file, but written in a secret code. Although there are complex encryption techniques, you should come up with a simple one of your own. For example, you could read the first file one character at a time, and add 10 to the character code of each character before it is written to the second file.

11. File Decryption Filter

Write a program that decrypts the file produced by the program in Programming Challenge 10. The decryption program should read the contents of the coded file, restore the data to its original state, and write it to another file.

Submit: Must include:

- a pdf file containing your information (student id, name), and images of the class diagrams, codes, answers
- and all source code files (if any)

in a zipped (.zip or *.rar) file to moodle