The Copperbelt University
School of Information and Communications Technology
Task: Lab 4
Due Date: 16th June 2023

Question 1 (25 marks)

Using interfaces, as you learned in this class, you can specify similar behaviors for possibly disparate classes. Governments and companies worldwide are becoming increasingly concerned with carbon footprints (annual releases of carbon dioxide into the atmosphere) from buildings burning various types of fuels for heat, vehicles burning fuels for power, and the like. Many scientists blame these greenhouse gases for the phenomenon called global warming. Create three small classes unrelated by inheritance—classes Building, Car and Bicycle.

Give each class some unique appropriate attributes and behaviors that it does not have in common with other classes. Write an interface **CarbonFootprint** with a **getCarbonFootprint** method. Have each of your classes implement that interface, so that its **getCarbonFootprint** method calculates an appropriate carbon footprint for that class (check out a few websites that explain how to calculate carbon footprints).

Write an application that creates objects of each of the three classes, places references to those objects in **ArrayList<CarbonFootprint>**, then iterates through the ArrayList, polymorphically invoking each object's **getCarbonFootprint** method. For each object, print some identifying information and the object's carbon footprint.

Question 2 (20 marks)

(Guess-the-Number Game) Write an application that plays "guess the number" as follows:
Your application chooses the number to be guessed by selecting an integer at random in the range 1–1000.

> The application then displays the following in a label:
> I have a number between 1 and 1000. Can you guess my number? Please enter your first guess.

A JTextField should be used to input the guess. As each guess is input, the background color should change to either red or blue. Red indicates that the user is getting "warmer," and blue, "colder." A JLabel should display either "Too High" or "Too Low" to help the user zero in.

When the user gets the correct answer, "Correct!" should be displayed, and the JTextField used for input should be changed to be uneditable. A JButton should be provided to allow the user to play the game again. When the JButton is clicked, a new random number should be generated and the input JTextField changed to be editable.

Question 3 (20 marks)

a) In Unit 9, you studied an inheritance hierarchy in which class BasePlusCommissionEmployee inherited from class CommissionEmployee. However, not all types of employees are CommissionEmployees.
In this question, you'll create a more general Employee superclass that factors out the attributes and behaviors in class CommissionEmployee that are common to all Employees. The common attributes and behaviors for all Employees are firstName, lastName, socialSecurityNumber, getFirstName, getLastName, getSocialSecurityNumber and a portion of method toString. Create a new superclass Employee that contains these instance variables and methods and a constructor. Next, rewrite class CommissionEmployee from Section 9.4.5 (**Source:** Recommended Book) as a subclass of Employee. Class CommissionEmployee should contain only the instance variables and methods that are not declared in superclass Employee.

Class CommissionEmployee's constructor should invoke class Employee's constructor and CommissionEmployee's toString method should invoke Employee's toString method. Once you've completed these modifications, run the CommissionEmployeeTest and BasePlusCommissionEmployeeTest apps using these new classes to ensure that the apps still display the same results for a CommissionEmployee object and BasePlusCommissionEmployee object, respectively.

b) Other types of Employees might include SalariedEmployees who get paid a fixed weekly salary, PieceWorkers who get paid by the number of pieces they produce or HourlyEmployees who get paid an hourly wage with time-and-a-half—1.5 times the hourly wage—for hours worked over 40 hours.

Create class HourlyEmployee that inherits from class Employee in part (a) and has instance variable hours (a double) that represents the hours worked, instance variable wage (a double) that represents the wages per hour, a constructor that takes as arguments a first name, a last name, a social security number, an hourly wage and the number of hours worked, set and get methods for manipulating the hours and wage, an earnings method to calculate an HourlyEmployee's earnings based on the hours worked and a toString method that returns the HourlyEmployee's String representation. Method setWage should ensure that wage is nonnegative, and setHours should ensure that the value of hours is between 0 and 168 (the total number of hours in a week).

Use class HourlyEmployee in a test program that's similar to the one in Fig. 9.5 (**Source:** Recommended Book).