SCHOOL OF COMPUTER SCIENCES
UNIVERSITI SAINS MALAYSIA
Semester II Session 2023/2024

Tutorial CPT113
Inheritance and Composition

Learning Outcomes:
- Building inheritance and composition using guided programming
- Solving applied questions on inheritance and composition.

Guided Programming

Question 1

This question has three parts. You need to answer them sequentially.

Part 1:
Write a class called `Shape` where the variables can hold the following values: `height`, `width`, `length` which are declared as `private` and the methods declared as a `public` member of the class. Write the following function:

a) `print`
b) `setVal` to set the values of all the variables
c) `getHW` to access the values of height and width
d) default constructor
e) overloading constructor
f) destructor
g) write the main to test all functions

Part 2:
Create another class that is inherited from `Shape` called `Prism`. `Prism` has one `private` member to retain the `volume`. Prism has the following function:

a) `print` to print l, w, h & volume
b) `setDimension` to set the length, width and height
c) `calculateVol` to calculate the volume of the prism
d) default constructor
e) overloading constructor that accepts **four parameters**
f) destructor
g) write the main to test overloading Prism constructor and to calculate the volume and print it out.
P.S.: Only change length variable from Shape to protected. Others must be maintained.

Part 3:
Write a new class called `Cube`. `Cube` has two `private` members: `volume` and composition of class Shape. `Cube` has the following functions:

a) `print` to print l, w, h & volume
b) `setDimension` to set the length, width and height
c) `calculateVol` to calculate the volume of the Cube
d) default constructor
e) destructor
f) write the main to test the functions created.

Formula:

Prism volume = length * height * width
Cube volume = height * height * height

Question 2

This question has four parts. You need to answer them sequentially.

Part 1:
Write a class called `Circle` where the variables can hold the values: `radius`, `radian`, `degree` and `area` which is declared as `private` and the methods declared as a public member of the class. Declare `PI` as a `constant` member class variable. Write the following function:

a) `print`
b) `setVal` to set the values of radius and degree
c) `getVal` to access all the values of variable in Circle
d) `calcRadian`
e) `calcArea`
f) default constructor
g) overloading constructor
h) destructor
i) write the main to test all functions

Part 2:
Create another class that is inherited from `Circle` called `Cone`. `Cone` has three private members to retain the height, area and volume. Cone has the following function:

a) `print` to print radius, height, area and volume
b) `setDimension` to set the radius and height
c) `calculateArea` to calculate the surface area of the cone
d) `calculateVol` to calculate the volume of the Cone
e) default constructor
f) overloading constructor that accepts **two parameters**
g) destructor
h) write the main to test overloading Circle constructor and to calculate the area and volume and print it out.

P.S.: Change `radius` variable from `Circle` to `protected`. Others must be maintained. Add method(s) if necessary in the base class.

Part 3:

Write a new class called `Cylinder`. `Cylinder` has two private members: `volume` and composition of class `Circle`. `Circle` has the following functions:

a) `print` to print radius, height, area & volume
b) `setDimension` to set the radius and height
c) `calculateVol` to calculate the volume of the Cylinder
d) default constructor
e) destructor
f) write the main to test the functions created.

Part 4

Create a class name `Ellipse` inherited from `Circle` with another variable as a private member representing the shorter radius. Write the function to calculate the Ellipse `area`.

Formula:

Radian calculation = degree × pi /180
Circle area = pi * radius^2
Cone area = pi * radius * (radius + sqrt(height^2 + radius^2))
Cone volume = pi * radius^2 * height
Cylinder surface area = (2 * pi * radius^2)+ (2 * pi * radius * height)
Cylinder volume = pi * radius^2 * height
Ellipse area = pi * radius * shortRadius

Problem Solving

Question 3

Given the following `Time.h` contains a `Time` class.

Design a class called `MilTime` that is derived from the Time class. The `MilTime` class should convert time in military (24-hour) format to the standard time format used by the `Time` class. The class should have the following member variables:

`milHours`: Contains the hour in 24-hour format. For example, 1:00 pm would be stored as 1300 hours, and 4:30 pm would be stored as 1630 hours.

`milSeconds`: Contains the seconds in standard format.

The class should have the following member functions:

`Constructor`: The constructor should accept arguments for the hour and seconds, in military format. The time should then be converted to standard time and stored in the `hours`, `min`, and `sec` variables of the `Time` class.

`setTime`: Accepts arguments to be stored in the `milHours` and `milSeconds` variables. The time should then be converted to standard time and stored in the `hours`, `min`, and `sec` variables of the `Time` class.

`getHour` : Returns the hour in military format.

`getStandHr` : Returns the hour in standard format.

Demonstrate the class in a program that asks the user to enter the time in military format. The program should then display the time in both military and standard format.

Input Validation: The `MilTime` class should not accept hours greater than 2359, or less than 0. It should not accept seconds greater than 59 or less than 0.

Question 4

Design a class named `TimeClock` . The class should be derived from the `MilTime` class you designed in the previous question. The class should allow the programmer to pass two times to it: starting time and ending time. The class should have a member function that returns the amount of time elapsed between the two times. For example, if the starting time is 900 hours (9:00 am), and the ending time is 1300 hours (1:00 pm), the elapsed time is 4 hours.

Input Validation: The class should not accept hours greater than 2359 or less than 0.

Question 5

Design an `Essay` class that is derived from the `GradedActivity` class used during lecture (make `GradedActivity` class as the base class). The `Essay` class should determine the grade a student receives on an essay.

The student's essay score can be up to 100, and is determined in the following manner:

- Grammar: 30 points
- Spelling: 20 points
- Correct length: 20 points
- Content: 30 points

Demonstrate the class in a simple program.

Question 6

Design a class named `PersonData` with the following member variables:

- `lastName`
- `firstName`
- `address`
- `city`
- `state`
- `zip`
- `phone`

Write the appropriate accessor and mutator functions for these member variables. Next, design a class named `CustomerData`, which is derived from the `PersonData` class. The `CustomerData` class should have the following member variables:

- `customerNumber`
- `mailingList`

The `customerNumber` variable will be used to hold a unique integer for each customer.

The `mailingList` variable should be a `bool` . It will be set to true if the customer wishes to be on a mailing list, or `false` if the customer does not wish to be on a mailing list. Write appropriate accessor and mutator functions for these member variables.

Demonstrate an object of the `CustomerData` class in a simple program.

Question 7 (Final Exam Sem 2, 2023/2024)

Analyse the following scenario:

There is a class Person, storing information available for a person which includes their full name, their state, and their entry qualification level (to university). There is also a class Student which is publicly inherited from class Person, which consists of Student ID, and is composed of the class Course. Each student is allowed to register up to 5 courses only.

All method definitions must be non-inlined function.

You are also given the following brief description to some methods.

Student members:

`printCourse()`: will check if the student has courses in their list and if it is not empty, display the details of the student's course.

`addCourse(string, string, int)`: accept course name, course code and unit to be added into the student's course list.

`addResult()`: will check if the course code is available in the student list, add student grade, then update grade point value (`pointVal` variable).

`searchCourse(string)`: will accept course code and search if the code is available in the list of course belonging to a student. Return 1 if the course code is found.

`numCourse()`: will return the number of courses that a student currently has.

`printFullReport()`: print out the details of the student from person details, student and course list

`calcGPA()`: will calculate the GPA of the courses in the list.

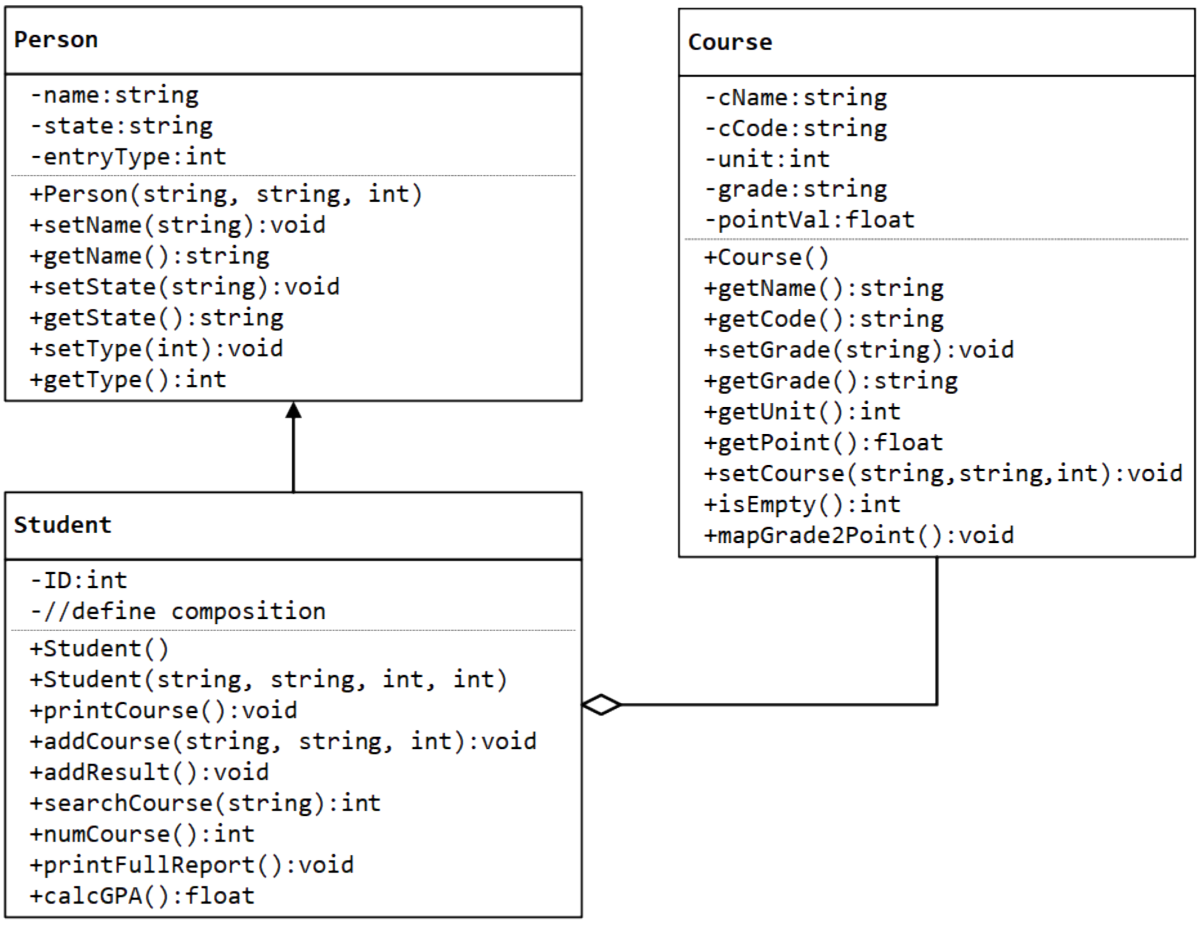
Course members:

`setCourse(string,string,int)`: accept 3 values for course name, course code and unit of that course.

`isEmpty()`: checking if any course has been added into the student's course. Will return 1 if there is no course and will return 0 if there is already course assigned to a student.

`mapGrade2Point()`: map the grade assigned to a course to their respective grade point value for example grade A will obtained 4.00 point value, grade A- will receive 3.67 point value.

The following UML diagram is provided to illustrate the structure of the classes and their relationships.



Show how should the composition of class `Course` be declared under the `Student` class. Refer the UML diagram given.

Write a method called `isEmpty()` that belongs to the `Course` class. If any additional methods are required (apart from the ones declared in the UML), their definitions need to be written as well.

Write a method called `addCourse()` that belongs to `Student` class. If any additional methods are required (apart from the ones declared in the UML), their definitions need to be written as well.

Write a method called `calcGPA()` that belong to `Student` class. This method first ensures that grades are assigned to all courses belonging to a student and will then calculate the GPA based on the list. If any additional methods are required (apart from the ones declared in the UML), their definitions need to be written as well.