Week 4: Constructor, Destructor, Default parameters, Static Member Variable

Learning Materials: Chapter 6

**** You may need to create additional functions to complete the tasks**

Task 1

Define a class named "Flight" with the following attributes and functionalities.

Private Members:

1. **flightNumber** (integer): Represents the flight number of the flight.
2. **destination** (string): Represents the destination of the flight.
3. **distance** (float): Represents the distance to the flight's destination in kilometers.
4. **maxFuelCapacity** (float): Represents the maximum fuel capacity of the flight in liters.

Public Members:

1. **Constructor** (Parameterized with default values):
    - Initializes the private members of the class.
    - Parameters:
        - **flightNum** (integer, default: 0): The flight number assigned to the flight.
        - **dest** (string, default: ""): The destination of the flight.
        - **dist** (float, default: 0.0): The distance to the flight's destination in kilometers.
        - **maxFuel** (float, default: 0.0): The maximum fuel capacity of the flight in liters.
2. **Destructor**:
    - Displays a message indicating the destruction of the flight object, including the flight number.
3. **CalFuel()**:
    - Calculates and returns the fuel required for the current distance value based on the following criteria:
        - If the distance is less than or equal to 1000 km, the fuel needed is 500 liters.
        - If the distance is more than 1000 km and less than or equal to 2000 km, the fuel needed is 1100 liters.
        - If the distance is more than 2000 km, the fuel needed is 2200 liters.
4. **ShowInfo()**:
    - Displays the flight information including flight number, destination, distance, and max fuel capacity.
    - Calls the **CalFuel()** function to calculate the required fuel and displays an appropriate message indicating whether the fuel capacity is fit for the flight distance.

Instructions:

1. Create an instance of the **Flight** class using the constructor, optionally providing values for flight number, destination, distance, and max fuel capacity.
2. Display the flight information using the **ShowInfo()** function.
3. The destructor will display a message indicating the destruction of the flight object.

Task 2

Define a class named "Student" with the following attributes and functionalities.

Private Members:

1. **firstName** (string): Represents the first name of the student.
2. **lastName** (string): Represents the last name of the student.
3. **studentID** (string): Represents the unique student ID.
4. **birthYear** (int): Represents the birth year of the student.
5. **course** (string): Stores the list of courses in which the student is enrolled.
6. **obtainedMark** (string): Stores the obtained marks for each enrolled course.
7. **totalStudents** (static int): Represents the total number of student objects created.

Public Members:

1. **Student(firstName, lastName, id, birthYear)**:
    - Initializes the private members of the class.
    - Increments the **totalStudents** count.
    - Calculates and returns the current age of the student based on the birth year.
2. **Destructor**:
    - Decrements the **totalStudents** count.
3. **enrollInCourse(courseName)**:
    - Enrolls the student in the specified course.
    - Initializes the obtained marks for the course to 0.0.
4. **generateEmail()**:
    - Generates and returns an email address for the student using their first name, last name, and a university domain.
5. **obtainedMarks(courseName, marks)**:
    - Assigns marks for the specified course.
6. **setGPAForEachCourse()**:
    - Calculates and returns the GPA for each course based on the obtained marks. You can follow a traditional grading system or define your own.
7. **displayCGPA()**:
    - Calculates and returns the Cumulative GPA (CGPA) based on the GPA of each course.
8. **willGraduate()**:
    - Prints whether the student will graduate or not with the current marks.
9. **applyForScholarship()**:
    - Checks if the CGPA is higher than 3.8 and allows students to apply for a scholarship.
10. **participateInInternship(company)**:
    - Checks if the CGPA is higher than 3.0 and if the student has taken a specific course.
    - If both conditions are met, the student can participate in an internship at a specific company.
11. **printAcademicRecord()**:
    - Prints the student's full name, ID, email, enrolled courses, graduation status, scholarship status, and internship status.

Instructions:

1. Create instances of the **Student** class using the constructor.
2. Use the provided member functions to manage student information, courses, marks, and various academic aspects.
3. The destructor will automatically decrement the **totalStudents** count when a student object is destroyed.

Task 3

Define a class named "Rectangle" with the following attributes and functionalities.

Private Attributes:

1. **length** (float): Represents the length of the rectangle.
2. **width** (float): Represents the width of the rectangle.

Public Member Functions:

1. **Rectangle (length = 1.0, width = 1.0)**:
    - Initializes the private members **length** and **width** with the provided values.
    - If no values are passed, default values of **length = 1.0** and **width = 1.0** are used.
2. **calculatePerimeter()**:
    - Calculates and returns the perimeter of the rectangle using the formula: **perimeter = 2 * (length + width)**.
3. **calculateArea()**:
    - Calculates and returns the area of the rectangle using the formula: **area = length * width**.
4. **calculateDiagonal()**:
    - Calculates and returns the diagonal of the rectangle using the Pythagorean theorem: **diagonal = sqrt (length^2 + width^2)**.
5. **calculateAngleWithLength()**:
    - Calculates and returns the angle between the diagonal and the length of the rectangle in degrees.
    - This calculation involves trigonometric functions based on the lengths of the sides.
6. **setDimensions(newLength, newWidth)**:
    - Sets the dimensions of the rectangle after verifying that **newLength** and **newWidth** are each floating-point number greater than or equal to 1.0 and less than 20.0.
7. **getDimensions()**:
    - Returns a pair containing the current length and width of the rectangle.

Instructions:

1. Create an instance of the **Rectangle** class using the constructor.
2. Use the provided member functions to calculate the perimeter, area, diagonal, and angle.
3. Use the **setDimensions** function to manage the dimensions of the rectangle.
4. Use the **getDimensions** function to retrieve the current dimensions.