LAB 6 (SECJ1023)

PROGRAMMING TECHNIQUE II

SECTION 04 & 05, SEM 2, 2024/2025

1. Given class declarations in the following code segment. Explain or draw the relationship between the class **`Person`**, **`TextBook`** and **`Course`**.

```
 //Program 6.1
class Person
{
   private:
      string name;
};

class TextBook
{
   private:
      Person author;
};
class Course
{
   private:
      TextBook book;
      Person instructor;
};
```

2. Consider the following Program 6.2. and answer questions (a), (b) and (c).

```
 //Program 6.2
#include <iostream>
using namespace std;

class Officer {
private :
    string name;
    int id;

public :
    Officer (string oname = "", int oid = 0) {
       name = oname; id = oid;}

    void set(string a, int b) {
       name = a; id = b;   }

    void print() {
       cout << "Officer    : " << name << " " << id << endl; }
};

class Offender {
private :
    string name, ic, address;

public :
    Offender(string oname="", string id="", string add=""){
       name = oname; address = add; ic = id; }

    void set(string a, string b, string c) {
       name = a;  ic = b;  address = c;}

    void print() {
       cout << "Offender :" << name << " " << ic << " "
            << address << endl; }
};

class Summon {
private :
    int code;
    Officer *officer;
    Offender offender;

public :
    Summon (int cod=0, string name="", string id="",
       string add = "" ){
       code = cod; offender.set(name,id,add); }

    void addOfficer(string a, int b) {
       officer = new Officer;  officer->set(a,b); }

    void removeOfficer() {
       delete officer;  officer = NULL; }

    void print() {
       cout<<endl<<"Summon Code : "<<code<<endl;
       offender.print();
       officer->print(); }
};

int main()
{
  Officer  a("ALI", 75381);
  Offender b("KASSIM","841203015311",
             "NO 1 JALAN PS 13 TAMAN PERMATA SINAR");
  Summon c(12,"SELAMAT","992112035621",
           "NO 3 JALAN PC 31 TAMAN PERMATA CAHAYA");

  c.addOfficer ("AHMAD", 87611);
  a.print();
  b.print();
  c.print();
  c.removeOfficer();

  return 0;
}
```

a. Describe the two approaches to implement the concept of class aggregrations and identify the line number of code in Program 6.2 which implement these approaches.

Name of the aggregation approach	**Description of the approach**	**The line number that implements the aggregation approach**

b. Based on Program 6.2, draw the UML diagram showing all the classes and the relationship between them.

c. What is the output of Program 6.2?

3. Given Program 6.3 as shown below. Answer questions (a) and (b) following it.

```
 //Program 6.3
#include <string>
#include <iostream>
using namespace std;

class Name
{
  private:
    string firstname, lastname;

  public:
    Name(string fname, string lname)
    {
      firstname = fname;
      lastname = lname;
    }

    string getFullName()
    {
      return firstname + " " + lastname;
    }
};

class Lecturer
{
  private:
    Name name;
    string staffId;

  public:
    Lecturer(string fname, string lname, string sId):
    name(fname, lname)
    {
      staffId = sId;
    }

    string getLecturer()
    {
      return name.getFullName() + "\nLecturer id  : " +
             staffId;
    }
};

class Department
{
  private:
    Lecturer *lecturerDepart;

  public:
    Department(Lecturer *lectDepart)
    {
      lecturerDepart = lectDepart;
    }

    void printDepartment()
    {
      cout << "Lecturer name: " <<
           lecturerDepart->getLecturer() << endl;
    }
};

int main()
{
  Lecturer *lect = new Lecturer("Abdullah", "Hamid", "124");
  Department department(lect);
  department.printDepartment();

  return 0;
}
```

a. Based on Program 6.3 given above, draw the UML class diagram that shows the relationship between classes.

b. What is the output of Program 6.3?

c. Write the class definitions that contain aggregation relationship for the following classes as depicted in Figure 6.1.

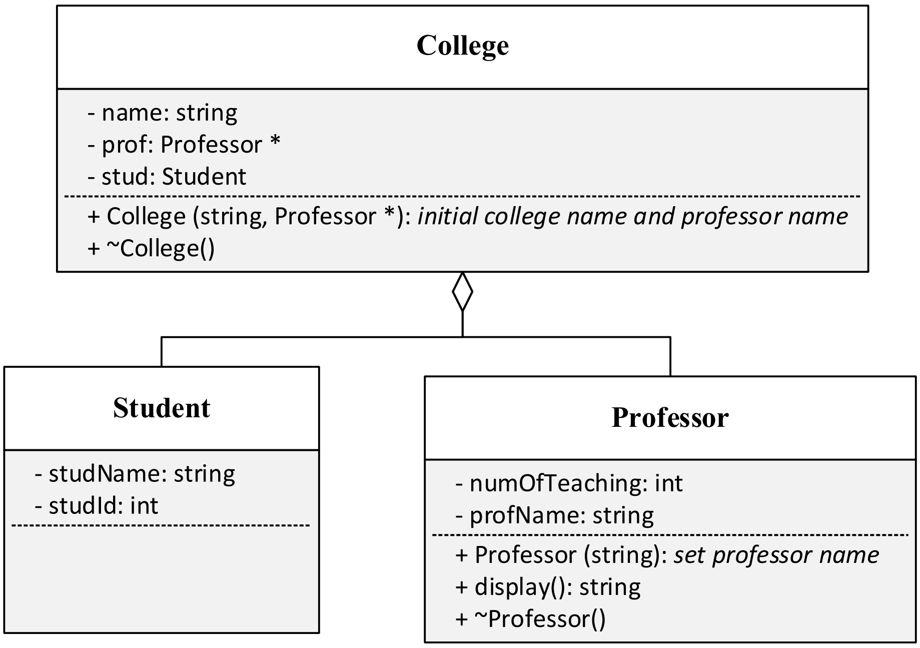

Figure 6.1: Aggregation relationship

4. Given Program 6.4 as shown below. Answer questions (a) and (b) following it.

```
 //Program 6.4
#include <iostream>
using namespace std;

class classA
{
    int val1;
    string val2;

  public:
    classA(int val1, string val2)
    { this->val1 = val1;
      this->val2 = val2; }

    int getValue() { return val1 * 100; }
    string getStringA() { return val2; }
};

class classC
{
    string value1;
    string value2;

  public:
    classC(string value1, string value2)
    { this->value1 = value1;
      this->value2 = value2; }

    string getStringC() { return value1 + ' ' + value2; }
};
```

```

class classB
{
    string valA;
    int valB;
    classA *valC;
    classC valD;

  public:
    classB(string valA, int valB, classA *C): valD("Good",
     "Luck")
    { this->valA = valA;
      this->valB = valB;
      valC = C; }

    void print()
    { cout << "In class A: " << valC->getStringA()
           << " " << valC->getValue() << endl
           << "In class C: " << valD.getStringC() << endl
           << "In class B: " << valA << " " << valB * 100; }
};

int main()
{
  classA *obj1 = new classA(50, "My target");
  classB obj2("For You", 10, obj1);
  obj2.print();
  return 0;
}
```

a. Based on Program 6.4 above, draw the UML class diagram that shows the relationship among classes.

b. What is the output of this program?

5. Consider the Program 6.5 and answer questions (a) and (b).

```
//Program 6.5
#include <iostream>
using namespace std;

class Instructor
{
  private:
    string instructorName;
    string officeNum;

  public:
    Instructor(string n, string o)
    { set(n, o); }

    void set (string n, string o)
    { instructorName = n;
      officeNum = o; }

    void print() const
    { cout << "\n Instructor name: " << instructorName;
      cout << "\nOffice number: " << officeNum << endl; }
};

class TextBook
{
    private:
       string title;
       string author;

    public:
       TextBook(string t, string a)
       { set(t, a); }

       void set(string t, string a)
       { title = t;
         author = a; }

        void print() const
        { cout << " \nTitle: " << title << endl;
          cout << "Author: " << author << endl; }
};

class Exam
{
  private:
    string examName;
    string date;

  public:
    Exam()
    { set("", ""); }

    Exam(string n, string d)
    { set(n, d); }

    void set (string n, string d)
    { examName = n;
      date = d; }

    void print() const
    { cout << "\nExam name: " << examName << endl;
      cout << "Date: " << date << endl; }
};

class Course
{
  private:
    string courseName;
    Instructor *instructor;
    TextBook *textbook;
    Exam exam;

  public:
    Course(string n, string d, string c, Instructor *I,
           TextBook *T): exam(n, d)
    { courseName = c;
      instructor = I;
      textbook = T; }

    void print() const
    { cout << "Course name: " << courseName;
      instructor->print();
      textbook->print();
      exam.print(); }
};

class Department
{
  private:
    string depName;
    Instructor *instructor;

  public:
    Department(string n, Instructor *I)
    { depName = n;
      instructor = I; }

    void print() const
    { cout << "\nDepartment name: " << depName;
      instructor->set("Amir Hamzah", "N28-301");
      instructor->print(); }
};

int main()
{
  Instructor *myInstructor = new Instructor("Noraminah
                             Hassan", "N28A-512");
  TextBook *myText = new TextBook("Introduction to C++",
                     "Daniel Liang");
  Department myDepart("Software Engineering", myInstructor);
  Exam myExam("Final Exam", "05 January 2018");
  Course myCourse("Test 1", "07 November 2017", "Programming
                  Technique II", myInstructor, myText);

  myCourse.print();
  myDepart.print();
  myText->set("Starting Out with C++", "Gaddis");
  myInstructor->print();
  myText->print();
  myExam.print();

  return 0;
}
```

a. Based on Program 6.5, draw the UML diagram showing all the classes and the relationship between them.

b. What is the output of this program 6.5?

6. Using Object-oriented Programming approach, develop a C++ program for a shipping company to manage their inventory of ships. The company keeps the following information about each ship:

a. The name of the ship

b. The year that the ship was built

c. The address that the ship was registered at. It includes the country where the ship was registered and the registrar office that keeps the registration record.

The UML class diagram for the inventory system is given as in Figure 6.2. In class **`Address`**, the function set is a mutator used for specifying the member variables, **`registrar`** and **`country`**, with values obtained from the function's arguments, while the functions **`getRegistrar`** and **`getCountry`** are accessors.

As for the class **`Ship`**, the function read is a mutator that specifies all the member variables, **`name`**, **`yearMade`** and **`address`**, with values obtained from the keyboard. The function print on the other hand, is used to print the member variables.

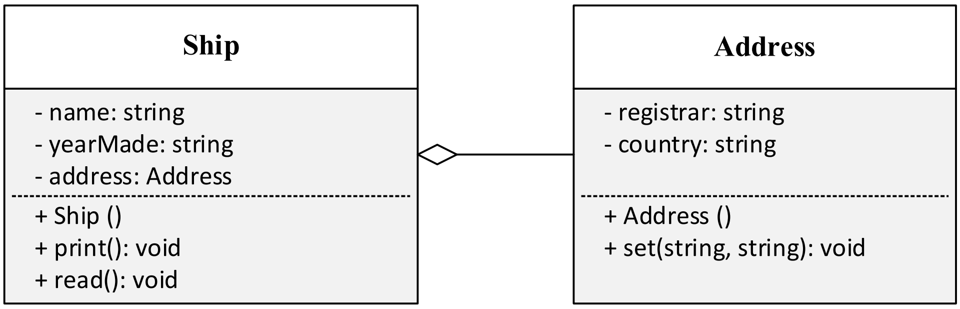

Figure 6.2: Class diagram for the ship inventory system

The implementation of your program must be based on the class diagram and specifications given above. Also, your program should use a dynamically allocated array to store the list of ships. Furthermore, the program should provide a menu-driven interaction for the user to use the program. It should allow the user to add the record for a ship one at a time. It should also provide an operation to display the number of ships and the information of each ship. Figure 6.3 illustrates an example run of the program. Note that the **bold texts represent** user inputs.

```
======== MENU ========
1. Add a ship
2. Display ships
3. Exit

Choose an operation  => 1
```

```
<<< Enter the information of the ship >>>

Ship Name: Super Gemilang
Year Built: 2010
The address the ship was registered:
Registrar Office: Malaysian Shipping Council
Country: Malaysia

Press any key to continue . . .

======== MENU ========
1. Add a ship
2. Display ships
3. Exit

Choose an operation  => 1

<<< Enter the information of the ship >>>

Ship Name: Orion Cruise
Year Built: 2000
The address the ship was registered:
Registrar Office: Shipping Authority Board
Country: Singapore

Press any key to continue . . .

======== MENU ========
1. Add a ship
2. Display ships
3. Exit

Choose an operation  => 2

<<< Inventory of ships >>>

Total ship: 2

==== Ship List ====

Ship Name: Super Gemilang
Year Built: 2010
Registered at:
  Malaysian Shipping Council, Malaysia

Ship Name: Orion Cruise
Year Built: 2000
Registered at:
  Shipping Authority Board, Singapore

Press any key to continue . . .

======== MENU ========
1. Add a ship
2. Display ships
3. Exit

Choose an operation  => 3
```

Figure 6.3: Example run

7. A faculty in a private university wishes to computerise its student records system. The record of each student includes his or her name and the program he or she enrolls. The faculty offers several programs at postgraduate level such as “Master of Business Administration”, “PhD of Social Science”, and many more. Regardless of the program or degree, each student is appointed with a lecturer as his or her academic advisor. The role of the advisor is to guide the student on academic matters. As for the postgraduate programs either master or PhD degrees, they are conducted in fully research-based. Each postgraduate student has to have a research project and a lecturer to supervise his or her project.

Based on the given problem, answer the following questions:

a. Draw the UML class diagram for the above problem. Your design has to include the classes and their attributes and methods accordingly, as well as relationships between the classes. Each class has to provide a constructor, mutators, and assessors.

b. Then, write the C++ code to implement the design. Your implementation should apply object-oriented programming concepts including data hiding, composition and/ or aggregation.

c. Next, utilize the classes to store a list of postgraduate students. You need to use dynamic arrays for the list and fill it in with data read from an input file. Figure 6.4 shows the example of input file containing the list of postgraduate students. The first line in each file indicates the number of students the file contains. Following that is the record of a student in which each attribute is arranged in a line. The student records are separated by blank lines. Finally, print all the students from the arrays into another text file. Figure 6.5 shows an example of the output file.

```
50                                     } The number of students

Bakar Bin Malik                        } Student’s name and program
Master of Social Science
Dr. Rohaya Qamarool                    } Academic advisor
Prof. Dr. Kamil Ahmad                  } Project supervisor
The Effects of Social Media            } Research project

Ruby Gabriella
PhD of Business
Prof. Dr. Kamil Ahmad
Prof. Dr. Saidi Abdullah
Adaptive Human Capital Management

(due to limited spaces, the remaining student records are not shown in this figure)
```

Figure 6.4: An example of input file named “pgstudents.txt”, for the list of postgraduate student records

Note that, the texts in *italic* are given to describe the fields accordingly. Also, only two records from the input file out of 50 are shown here due to limited spaces.

```
THE LIST OF POSTGRADUATE STUDENTS
No Name            Supervisor            Project

 1  Bakar Bin Malik   Prof. Dr. Kamil Ahmad   The Effects of Social Media
 2  Ruby Gabriella    Dr. Saidi Abdullah      An Adaptive Human Capital Management
.. ...............  .....................  ....................................
                                          (students no 3 to 49 go here)
.. ...............  .....................  ....................................
50  Zul Bin Hashim    Prof. Dr. Kamil Ahmad   Artificial intelligence for robots
```

Figure 6.5: An example of the output file containing the lists of postgraduate students

Note that only some students are shown Figure 6.5 due to limited spaces.

8. Consider the class diagram in Figure 6.6 which shows the data model for a car rental company. Note that the company has set the rule that each customer can only rent one car at a time. Based on the class diagram, write a C++ program which performs the following tasks:

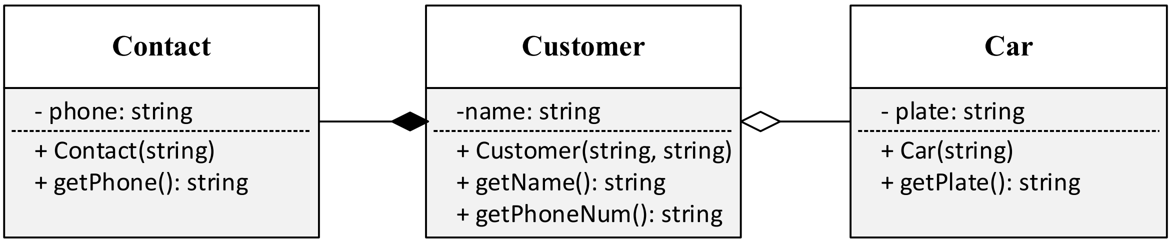

Figure 6.6: Class diagram for a car rental service

a. Implement all the three classes with the given attributes and operations. Note that, the purpose of each operation is as the name implies.

b. Test the classes by creating an object of **`Car`** and an array of customers with the following data:

Customer's Name	Phone Number	Rented Car Plate
Ahmad Kamal	015-75769800	JSQ245
Siti Nurdiana Abdullah	014-8889900	

Note that, the column **"Rented Car Plate"** for the second customer is empty because she does not rent any car at the moment.

c. Print the array of customers onto the screen. The screen output should look like as in Figure 6.7.

```
Customer's Name: Ahmad Kamal
Phone Number: 015-75769800
Rented Car  : JSQ245

Customer's Name: Siti Nurdiana Abdullah
Phone Number: 014-8889900
Rented Car  :
```

Figure 6.7: Screen output

9. Write a complete C++ program for the Person and Cat classes based on the Figure 6.8 and the information provided.

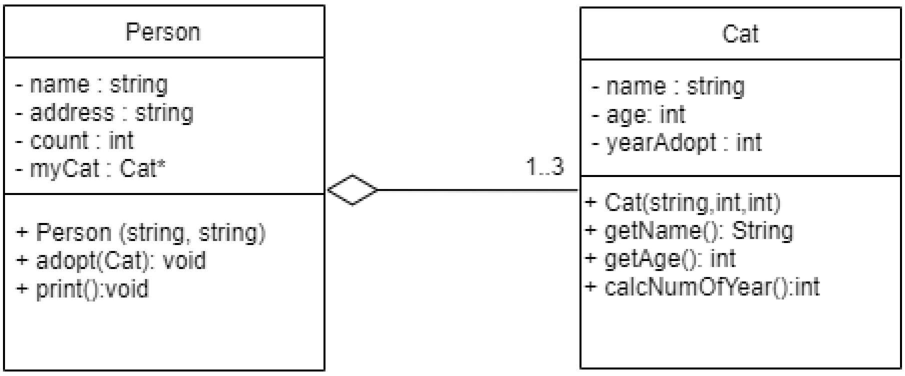

Figure 6.8: Class diagram

a. The Person class consists of the following members:

- A constructor that accepts the person's name and address as arguments. These values should be assigned to the object's name and address member variables.
- adopt function: This function is used to add objects from the Cat class to myCat array. The objects added to array refers to the cats adopted by person.
- print function: This function will display the person information along with his adopted cats and number of years he has adopted the cat(s).

b. The Cat class consists of the following members:

- A constructor that will initialize all the member attributes to the values received as arguments.
- getName function: This function will return a value of name member attributes.
- getAge function: This function will return a value of age of the cat adopted.
- calcNumOfYear function: This function will calculate number of years one cat being adopted.