School Computing, Faculty of Engineering
Universiti Teknologi Malaysia

SCSJ1023 Programming Technique II
Semester 1, 2018/2019

Exercise 3
Inheritance

Duration : 90 minutes

Objectives
This exercise aims at implementing the following concept:

- Inheritance

Exercise Materials
- Program templates are provided for this exercise. Please download from the elearning and extract the ZIP file to your local drive.
- You have the choice to use Microsoft **VS Code**, **Dev C++** or any other IDEs to write the code for this exercise.
- If you choose VS Code, use debug **"Console program"** to run the program.
- If you use Dev C++, you do not have to work in a folder. Instead, you open directly the source file.

Deliverable Item
- Only the **source code** file is needed for the submission (i.e., **exercise3.cpp**).
- You must submit your source code at elearning.

Plagiarism Policy
- Discussions among the students are still possible during the exercise session.
- However, all works must be done individually.
- Any kind of plagiarism (e.g., copy and paste code by any mean) would lead to disqualification of submissions for both parties (i.e., students that copy others' code and students that give their code to others).

Late Submission Policy
- **5% deduction for every 5 minutes late**.
- For example: if the duration of the exercise is 90 minutes, and your submission is received on the 91st minute, you are only eligible to earn 95% at maximum of the total marks.

Case Study

You will be writing a program to record student information of a university. There are two types of students, undergraduate and postgraduate students. The program needs to record the name and matriculation number for each student. Besides, the program also needs to record the cgpa for each undergraduate student, and the project information for each postgraduate student. Regarding the project of the postgraduate students, each project consists of a title and area or discipline.

Tasks

The exercise is divided into three parts:

- Part 1 is about design. You will be doing this part on a piece of paper and it needs to be handed in.
- Part 2 consists of guided tasks, in which the questions or tasks are accompanied with the code segment for the solution. This part will not be assessed. However, this part is a pre-requisite for the next part.
- Part 3 consists of tasks that you need to do it yourself.

Part 1: Design

You are supposed to spend about 15 to 20 minutes to complete this part. This part will be graded. ***Do this part on a piece of paper and hand it in before proceeding to Part 2****.*

1. Analyze the given case study and draw the UML **class diagram** consisting of classes **Student**, **Project**, **Undegraduate**, and **Postgraduate**, as well as their relationships.

   Tips: the class diagram should use **inheritance** and **composition** relationships.

Part 2: Guided Coding Tasks

You are supposed to spend about 20 to 30 minutes to complete this part. This part will not be graded. However, it is a pre-requisite for the next part.

2. Full implementations for the classes `Student` and `Project` have been given in the template program **exercise3.cpp**. Review the code of these classes.

3. Complete the implementation for the class `Undergraduate.` Here, you are supposed to implement inheritance. The solution is given in the following code segment. Copy only the highlighted lines from the code segment below. Then review the code.

```
class Undergraduate : public Student
{
  protected:
    double cgpa;

  public:
    Undergraduate( string _name = "",
                   string _matric = "",
                   double _cgpa = 0.0  ) : Student(_name, _matric), cgpa(_cgpa)
    {}
    double getCGPA() const { return cgpa; }
    void setCGPA(double c) { cgpa = c; }

    void input()
    {
        Student::input();
        cout << "Enter CGPA => ";
        cin >> cgpa;
    }

    void print() const
    {
        Student::print();
        cout << "CGPA   : " << cgpa << endl;
    }
};
```

Part 3: Graded Coding Tasks

You are supposed to spend about 30 to 40 minutes to complete this part. This part will be graded.

4. Complete the implementation for the class `Postgraduate`.

   Tips: Part 2 should give the idea to complete this task.
   Besides inheritance, you also need to implement composition for the class (i.e. for the student's project)

Task 5 to 8 below are related to the main function:

5. Create two arrays to hold lists of undergraduate and postgraduate students, respectively.

6. Read data for an undergraduate student and add it to the relevant array.

7. Read data for a postgraduate student and add it to the relevant array.

8. Diplay the lists of undergraduate and postgraduate students.

Figure 1 ((a) to (g)) shows expected result that the program should produce. The bold texts are user inputs.

```
=============[ Menu ]=============
1. Add an undergraduate student
2. Add a postgraduate student
3. Display all students
9. Exit

Choose an operation [1,2,3 or 9] => 1

Adding an undergraduate student:

Enter name => Aminah Abdullah
Enter matric => A16CS5022
Enter CGPA => 3.85
```

(a) The user has chosen operation 1 to add a new undergraduate student

```
=============[ Menu ]=============
1. Add an undergraduate student
2. Add a postgraduate student
3. Display all students
9. Exit

Choose an operation [1,2,3 or 9] => 1

Adding an undergraduate student:

Enter name => Abdul Jalil Rahman
Enter matric => B17CS3012
Enter CGPA => 3.7
```

(b). The user has chosen operation 1 to add a new undergraduate student

```
=============[ Menu ]=============
1. Add an undergraduate student
2. Add a postgraduate student
3. Display all students
9. Exit

Choose an operation [1,2,3 or 9] => 2

Adding a postgraduate student:

Enter name => Hassan Shahir
Enter matric => PC150523
Enter project title => Machine learning with abnormal data
Enter project area => Artificial Intelligence
```

(c). The user has chosen operation 2 to add a postgraduate student

```
=============[ Menu ]=============
1. Add an undergraduate student
2. Add a postgraduate student
3. Display all students
9. Exit

Choose an operation [1,2,3 or 9] => 2

Adding a postgraduate student:

Enter name => Siti Nurliyana Kamaruddin
Enter matric => MA185246
Enter project title => Cost-effective house building
Enter project area => Building and construction
```

(d). The user has chosen operation 2 to add a postgraduate student

```
=============[ Menu ]=============
1. Add an undergraduate student
2. Add a postgraduate student
3. Display all students
9. Exit

Choose an operation [1,2,3 or 9] => 1

Adding an undergraduate student:

Enter name => Zainuddin Abdul Jalil
Enter matric => B18CS4586
Enter CGPA => 3
```

(e). The user has chosen operation 1 to add a new undergraduate student

```
=============[ Menu ]=============
1. Add an undergraduate student
2. Add a postgraduate student
3. Display all students
9. Exit

Choose an operation [1,2,3 or 9] => 3


The list of undergraduate students:

No.   : 1
Name  : Aminah Abdullah
Matric: A16CS5022
```

```
CGPA  : 3.85

No.   : 2
Name  : Abdul Jalil Rahman
Matric: B17CS3012
CGPA  : 3.7

No.   : 3
Name  : Zainuddin Abdul Jalil
Matric: B18CS4586
CGPA  : 3


The list of postgraduate students:

No.   : 1
Name  : Hassan Shahir
Matric: PC150523
Project title  : Machine learning with abnormal data
Project area  : Artificial Intelligence

No.   : 2
Name  : Siti Nurliyana Kamaruddin
Matric: MA185246
Project title  : Cost-effective house building
Project area  : Building and construction
```

(f). The user has chosen operation 3 to list the students

```
=============[ Menu ]=============
1. Add an undergraduate student
2. Add a postgraduate student
3. Display all students
9. Exit

Choose an operation [1,2,3 or 9] => 9
```

(g). The user has chosen operation 9 to end the program

Figure 1