E1800175_FINAL EXAM

by I Kadek Yau Dwi Mega Saputra

Submission date: 12-Jan-2021 12:56PM (UTC+0800)
Submission ID: 1486155400
File name: BIT205_E1800175.docx (85.8K)
Word count: 2045
Character count: 11996

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

HELP University
university of achievers

STIKOM BALI

SEMESTER 1, 2020

TAKE HOME EXAMINATION

Subject Code : **BIT205**	
Subject Name : **OBJECT ORIENTED PROGRAMMING IN C++**	
This examination carries 50% of the total assessment for this subject.	
Examiner(s)	**Moderator(s)**
SEETHA LETCHUMI	**MR KOK CHYE HOCK**
Duration: 4 HOURS **Turnitin Class Code: 23614527** **Enrolment Key: C++2020**	
Start Time: **Day:** WEDNESDAY **Date: 13 MAY 2020** **Release Time: 9.30AM**	**Due Date and Time:** **Day: WEDNESDAY** **Date: 13 MAY 2020** **Due Time: 1.30PM**

Declaration

- I have read and understood the Academic Integrity Policy that explains on **plagiarism**, and I testify that, unless otherwise acknowledged, the work submitted herein is entirely my own.
- I declare that no part of this examination has been written for me/ by any other person(s)
- I authorize the University to test any work submitted by me using text comparison software, for instances of plagiarism. I understand this will involve the University or its contractors copying my/our work and storing it on a database to be used in future to test work submitted by others.

Note:1) The attachment of this statement on any electronically submitted examinations will be deemed to have the same authority as a signed statement.

Student Name: I Kadek Yau Dwi Mega Saputra **Student ID: E1800175**

Instructions:

1. Download the exam paper from the e-learning course/subject folder at the stipulated start time.

2. Read and follow the instructions carefully and take note of the submission timeline and the required format.

3. Answer ALL the exam questions in Microsoft Word using the Cover Sheet attached as the first page.

4. You are not allowed to discuss the exam questions or answers with anyone, including your lecturers, friends, classmates or any other parties.

5. Label your answers clearly.

6. Save your Word document at regular intervals to avoid losing any data.

7. Name your Word document with your StudentID followed by subject code (e.g.B1223432BIT205.doc)

8. It is your responsibility to keep track of the time.

9. Once you have completed your exam:

A) submit your Word document to Turnitin.

B) submit a **copy** of the Word document to LMS.

Both submissions in (A) and (B) must be the same. However, only the copy in Turnitin will be marked.

10. You are only allowed to do ONE submission respectively to Turnitin and LMS.

11. The duration of the exam is 4 hours. Therefore, you will need to submit your answers to Turnitin within this time frame. You are advised to submit earlier than the actual end time to avoid high traffic on the submission sites.

12. Any submissions after the stipulated exam end time will be considered as a NON-SUBMISSION.

13. Once the answer is submitted for grading, NO requests for amendments or submission of additional documents will be permitted or accepted.

14. In case of any technical problems e.g., internet connection or computer malfunction, you must immediately contact your lecturer via the LMS with screenshots of the issues.

15. In the event that an investigation is required due to technical issues, plagiarism or misconduct, you will have to sit for a Face-to-Face examination at the next available examination period.

Important notes for Open Book Exams:

1. You must attempt this exam on your own. Showing it or discussing it with anyone else is not permitted.
2. You will be assessed on your ability to synthesise, interpret, collect, and systematically analyse information that is readily available. However, the work submitted must be original.
3. You may use any publicly available materials, including lecture notes, books, the internet, etc. These sources of information must be referenced.
4. You are required to read and adhere to the Academic Integrity Policy prior to the examination date.

Expected Learning Outcomes Assessed

CLO1: Analyze problems and interpret technical specifications to create and program appropriate algorithmic solutions that include the use of control structures, parameters and return values. (C3, A5)

CLO2: Construct program using reference variables and the management of dynamically linked structure. (C6)

CLO3: Design and implement object oriented solutions to programming problems using C++. (C6, P7)

Questions 1 until Question 4 are based on Fig. 1. Member and Trainer are derived from the class User.

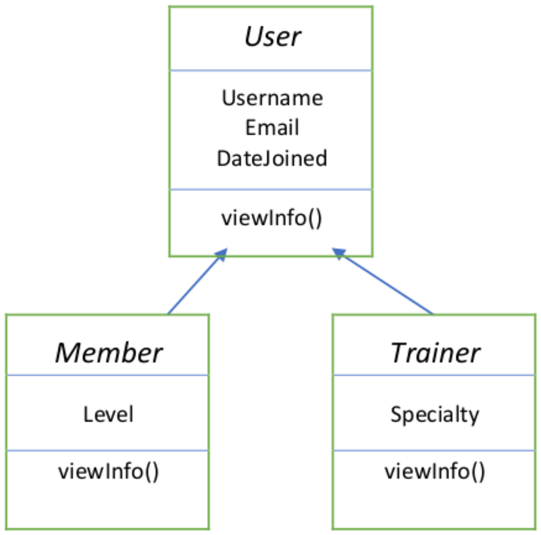

Figure 1: Class Diagram

Question 1

Write the complete DEFINITION (attributes and methods) of the abstract class named **User** that includes the following attributes:

- Username
- Email
- DateJoined

And the following methods:

- A parameter constructor
- A suite of getters and setters. A getter with public access is provided for each of the private data elements. A setter with protected access is provided for each of the private data elements.
- ***viewInfo*** method that print all the attributes of the class User
- Overload the friend operator **==** which accepts *User* objects and returns **true** if the *email* addresses of the objects are equal

Question 2

Write the complete DEFINITION (attributes and methods) of the class named *Member* that is publicly derived from the class *User* that includes the following attributes:

- level - (beginner, intermediate, expert)

and the following methods

- parameter constructor
- getters and setters for level
- overload the method, ***viewInfo*** to print out all the details of the Trainer inclusive of the details inherited by the object.

Question 3

Write the complete DEFINTION (attributes and methods) of the class named Trainer that is publicly derived from the class User that includes the following attributes:

- specialty

and the following methods

- parameter constructor
- getters and setters for specialty
- overload the method, ***viewInfo*** to print out all the details of the Trainer inclusive of the details inherited by the object.

Question 4

In the main function, you are required to use STL (either a list or vector) to create a collection of Users objects.

Your C++ program will have the following functionalities:

1. **Add a new user** - A user, either a member or a *trainer* can be added to the collection of users. Validate level to store the values: beginner, intermediate, expert for *Member* objects.
2. **Edit an existing user** – Based on existing *email* address of a user, all other information (except email address) of a user can be updated.
3. **Delete an existing user –** Based on an existing *email* address of a user, the user object is to be deleted.

4. **View User** – Based on an existing *email* address of a user, the user object details are viewed.
5. **List All Users**– A list of all users with their information is viewed. At the end there will also be a total count for each type of users i.e. a count of members and a count of trainers.
6. **Menu** to perform the tasks from 1 – 5.

Requirements:

A. Program will demonstrate the use of
   - STL (either a vector or list) for the collection of users.
   - 'getters' and 'setters' to access the class objects
   - public inheritance
   - iterator to access the vector
   - polymorphism via ***viewInfo*** method.
   - operator overloading of == as friend functions

B. An executable C++ program without any syntax errors

SUBMISSION REQUIREMENT

A. Your examination has to be submitted to TurnitIn, with the following all contained in a single file:
   1. All your cpp source files, printed in Word document format
      Submit your Turnitin Report (http://www.turnitin.com)

         - Class Name : **STIKOM BIT205 2020**
         - Class ID : **23614527**
         - Enrollment Key : **C++2020**

   - Please refer to the Academic Integrity Policy (https://lms.help.edu.my/helpelearning/pluginfile.php/90637/mod_resource/content/1/Academic%20Integrity%20Policy%20%28FCDT%29.pdf) for penalties to be applied in the event of misconduct

B. You are also required to be submit to **LMS** https://lms.help.edu.my/helpelearning/mod/assign/view.php?id=64687 **(a zipped file consisting of):**

- All your source files (inclusive of the exe file) of your project.
- MS Word of the Turnitin submission

CLO	Item		Marks	Awarded Marks
CLO1	Coding Style	Variable Names/Indentation	2	
		Functions/Parameter Passing	3	
		Efficiency of Code	3	
CLO2, CLO3	Requirement	**User (abstract)**	2	
		1. attributes	1	
		2. constructor	2	
		3. getters/setters	3	
		4. viewInfo()	5	
		5. friend operator ==	5	
		Member (public inheritance)	1	
		1. attributes	1	
		2. constructor	3	
		3. getters/setters	2	
		4. viewInfo()	3	
		Trainer (public inheritance)	1	
		1. attributes	1	
		2. constructor	3	
		3. getters/setters	2	
		4. viewInfo()	3	
		Usage of STL - collection of users	3	
		Use of Iterator to access STL	5	
CLO2, CLO3	Program Execution	Adding Trainer	6	
		Adding Member	6	
		Delete Trainer/Member	6	
		Edit Trainer/Member	6	
		Display Member/Trainer	6	
		Display All + Trainer Count + Member Count	6	
		Validation	5	
		Menu	5	
		TOTAL	100	

MAIN.cpp

```
//Name  : I Kadek Yau Dwi Mega Saputra
//Nim   : E1800175
//Compal: CodeBlock


#include <iostream>
#include <trainJourney.h>
#include <freightJourney.h>
#include <passengerJourney.h>

#include <vector>

using namespace std;

int trainJourney::ai = 1111;

/// Methods required for question 4
int menu();
void addtrainJourney(vector<trainJourney*> &listJourney);
void viewpassengerJourneys(
        vector<trainJourney*> &listJourney,
        const string& startTown,
        const string& endTown
        );
void summaryReport(vector<trainJourney*> &listJourney);


int main() {
    vector<trainJourney*> listJourney;
    int chosenMenu;
    do {
        chosenMenu = menu();
        switch (chosenMenu) {
            case 0: {
                cout << "Quitting.." << endl;
                break;
            }
            case 1: {
                addtrainJourney(listJourney);
                break;
            }
            case 2: {

                string st, et;
                cout << "Start town >>> ";
                cin >> ws;
                getline(cin, st);
                cout << "End town >>> ";
                cin >> ws;
                getline(cin, et);
                viewpassengerJourneys(listJourney, st, et);
                break;
            }
            case 3: {
                summaryReport(listJourney);
                break;
            }
            default:
                cout << "Invalid input"; break;
        }
        cout << "\n-------------------------------------\n";
    } while (chosenMenu != 0);
    return 0;
}

int menu() {
    int selMenu;
    cout << "1. Add a new train journey\n"
            "2. View passenger train journey\n"
            "3. Summary report\n"
            "0. quit\n"
            ">>>>>>>> ";
    cin >> selMenu;
    return selMenu;
}

void addtrainJourney(vector<trainJourney*> &listJourney) {
    string st, et;
    int jTime;
    trainJourney *journey;

    int chosenJourneyType;
    do {
        cout << "1. Freight\n"
                "2. Passenger\n"
                ">>> ";
        cin >> chosenJourneyType;
        if (chosenJourneyType != 1 && chosenJourneyType != 2)
            cout << "Invalid journey type" << endl;
        else break;
    } while (true);

    cout << "Enter origin town: ";
    cin >> ws;
    getline(cin, st);

    cout << "Enter destination town: ";
    cin >> ws;
    getline(cin, et);

    cout << "Time it takes: ";
    cin >> jTime;

    if (chosenJourneyType == 1) {
        int capacity, carriages;
        string hazardousOpt;
        bool canCarryHazarous;
        cout << "Capacity: ";
        cin >> capacity;
        cout << "Carriages: ";
        cin >> carriages;
        cout << "Can carry hazardous materials ? [y to allow]";
        cin >> ws;
        getline(cin, hazardousOpt);
        canCarryHazarous = (hazardousOpt == "y" || hazardousOpt == "Y");
        journey = new freightJourney(st,et, jTime, capacity, carriages,
                canCarryHazarous);
    } else {
        int noFirst, noSecond, noEco;
        string restaurantOpt;
        bool hasRestaurantOnBoard;
        cout << "First class capacity: ";
        cin >> noFirst;
        cout << "Second class capacity: ";
        cin >> noSecond;
        cout << "Economy class capacity: ";
        cin >> noEco;
        cout << "Has a restaurant on board? [Y/n]";
        cin >> ws;
        getline(cin, restaurantOpt);
        hasRestaurantOnBoard = (restaurantOpt == "y" || restaurantOpt == "Y");
        journey = new passengerJourney(st, et, jTime, noFirst, noSecond,
                noEco, hasRestaurantOnBoard);
    }
    listJourney.push_back(journey);
    cout << "New train journey added! Detail: \n";
    journey->viewJourney();
}

void viewpassengerJourneys(
        vector<trainJourney*> &listJourney,
        const string& startTown,
        const string& endTown
) {
}

void summaryReport(vector<trainJourney*> &listJourney) {

}
```

trainJourney.h

```
#ifndef TRAINJOURNEY_H
#define TRAINJOURNEY_H
#include <string>

using namespace std;

/// QUESTION 1
class trainJourney {
    int trainID;
    string startTown;
    string endTown;
    int journeyTime;

public:
    static int ai;

    trainJourney(
            const string &startTown,
            const string &endTown,
            int journeyTime
            ):
            startTown(startTown),
            endTown(endTown),
            journeyTime(journeyTime) { trainID = ai++; }

    int getTrainId() const {
        return trainID;
    }

    const string &getStartTown() const {
        return startTown;
    }

    const string &getEndTown() const {
        return endTown;
    }

    int getJourneyTime() const {
        return journeyTime;
    }

    virtual void viewJourney() = 0;

    friend
    bool operator==(const trainJourney &ltj, const trainJourney &rtj) {
        return ltj.trainID == rtj.trainID &&
               ltj.startTown == rtj.startTown &&
               ltj.endTown == rtj.endTown &&
               ltj.journeyTime == rtj.journeyTime;
    }

protected:
    void setTrainId(int trainId) {
        trainID = trainId;
    }

    void setStartTown(const string &startTown) {
        trainJourney::startTown = startTown;
    }

    void setEndTown(const string &endTown) {
        trainJourney::endTown = endTown;
    }

    void setJourneyTime(int journeyTime) {
        trainJourney::journeyTime = journeyTime;
    }

    void printAttributes() {
        cout << "ID: " << trainID
        << " Origin: " << startTown
        << " Destination: " << endTown
        << " takes " << journeyTime << " minutes.";
    }
};



#endif // TRAINJOURNEY_H
```

freightJourney.h

```
#ifndef FREIGHTJOURNEY_H
#define FREIGHTJOURNEY_H
#include <trainJourney.h>
#include <string>

/// QUESTION 2
class freightJourney: public trainJourney {
    int capacity;
    int carriages;
    bool hazardous;
public:
    freightJourney(
            const string &startTown,
            const string &endTown,
            int journeyTime,
            int capacity,
            int carriages,
            bool hazardous
            ): trainJourney(
                    startTown,
                    endTown,
                    journeyTime
                    ),
                    capacity(capacity),
                    carriages(carriages),
                    hazardous(hazardous) {}

    int getCapacity() const {
        return capacity;
    }

    void setCapacity(int capacity) {
        freightJourney::capacity = capacity;
    }

    int getCarriages() const {
        return carriages;
    }

    void setCarriages(int carriages) {
        freightJourney::carriages = carriages;
    }

    bool isHazardous() const {
        return hazardous;
    }

    void setHazardous(bool hazardous) {
        freightJourney::hazardous = hazardous;
    }

    void viewJourney() override {
        cout << "(Freight Journey) "
        << "hazardous materials"
        << (hazardous ? " " : " not ")
        << "allowed. ";
        printAttributes();
        cout << "Capacity: " << capacity
        << ". Carriages: " << carriages
        << endl;
    }
};
#endif // FREIGHTJOURNEY_H
```

passengerJourney.h

```
#ifndef PASSENGERJOURNEY_H
#define PASSENGERJOURNEY_H
#include <trainJourney.h>
#include <string>


/// QUESTION 3
class passengerJourney: public trainJourney {
    int noFirstClass, noSecondClass, noEcoClass;
    bool catering;
public:
    passengerJourney(
            const string &startTown,
            const string &endTown,
            int journeyTime,
            int noFirstClass,
            int noSecondClass,
            int noEcoClass,
            bool catering
            ): trainJourney(
            startTown,
            endTown,
            journeyTime
            ), noFirstClass(noFirstClass),
               noSecondClass(noSecondClass),
               noEcoClass(noEcoClass),
               catering(catering) {}

    int getNoFirstClass() const {
        return noFirstClass;
    }

    void setNoFirstClass(int noFirstClass) {
        passengerJourney::noFirstClass = noFirstClass;
    }

    int getNoSecondClass() const {
        return noSecondClass;
    }

    void setNoSecondClass(int noSecondClass) {
        passengerJourney::noSecondClass = noSecondClass;
    }

    int getNoEcoClass() const {
        return noEcoClass;
    }

    void setNoEcoClass(int noEcoClass) {
        passengerJourney::noEcoClass = noEcoClass;
    }

    bool isCatering() const {
        return catering;
    }

    void setCatering(bool catering) {
        passengerJourney::catering = catering;
    }

    void viewJourney() override {
        cout << "(Passenger Journey) "
             << "has"
             << (catering ? " " : " no ")
             << "restaurant on board. ";
        printAttributes();
        cout << "[First: "
        << noEcoClass
        << ", Second: "
        << noSecondClass
        << ", Eco: "
        << noEcoClass
        << "]"
        << endl;
    }
};

#endif // PASSENGERJOURNEY_H
```