Topic : Online Educational Games

Group no : KDY_07

Campus : Kandy

Submission Date : 19th of May 2022

We declare that this is our own work and this Assignment does not incorporate without acknowledgment any material previously submitted by anyone else in SLIIT or any other university/Institute. And we declare that each one of us equally contributed to the completion of this Assignment.

Registration No	Name	Contact Number
IT21244452	D.G.N.U.S.Wijayasiri	071 129 9896
IT21277054	M.S.Shiraz	076 564 3133
IT21271182	R.M.U.V.Rathnayake	076 051 0582
IT21227554	A.K.S.Chandrajith	071 266 8133
IT21298080	B.M.V.B.B.Gurusinghe	077 659 7283

1) Description of the requirements

User ID

- User ID taken to verify the number of friend that the user has.

Student Academic Status

- If the user is a student, the user should verify his user to get academic rewards.
- Also depending on their Academic status User should verify their institute or their respective teacher.

School Student's related Teachers ID

- School students receive a teacher ID from their teacher which should be verified to get a special gamer code which give special access to that student.

Game Name

- Game Name is required to specify the game that the user wants to purchase.
- Game Name is also required to verify which game the user wants to refund.

Friend Name

- Friend ID is required to determine which friend / friends do the user have and to claim the discount accordingly.

Number of Months of Subscription

- Number of months is needed to specify how long the user wants the game to be accessible for the respective user.
- Number of months is also used to determine how much time is left if the user requires a refund.

Number Purchases Done by a User

- Number Purchases is needed to determine the number of purchases done by the user and give a discount if the amount surpasses a certain number of purchases.

2) Identified classes

1. User
2. Student
3. School Student
4. Institute Student
5. Teacher
6. Game
7. Friend
8. Subscription
9. Leaderboard
10. Payment History
11. Refund

3) CRC cards

User	
Responsibility	Collaborators
Get User Details	
Display User Details	

Student	
Responsibility	Collaborators
Get Student Details	
Display Student Details	

School Student	
Responsibility	Collaborators
Get School Student Details	
Display School Student Details	
Get game code from teacher	Teacher

Institute Student	
Responsibility	Collaborators
Get Student Details	
Display Student Details	
Check Validity for Student Offer	
Display Student Discount	

Teacher	
Responsibility	Collaborators
Get Teacher Details	
Display Teacher Details	
Store Payment History	Payment History
Request Refund Details	Refund

Game	
Responsibility	Collaborators
Store Game Details	
Display Game Details	

Friend	
Responsibility	Collaborators
Get Friend Discount	
Display Friend Discount	

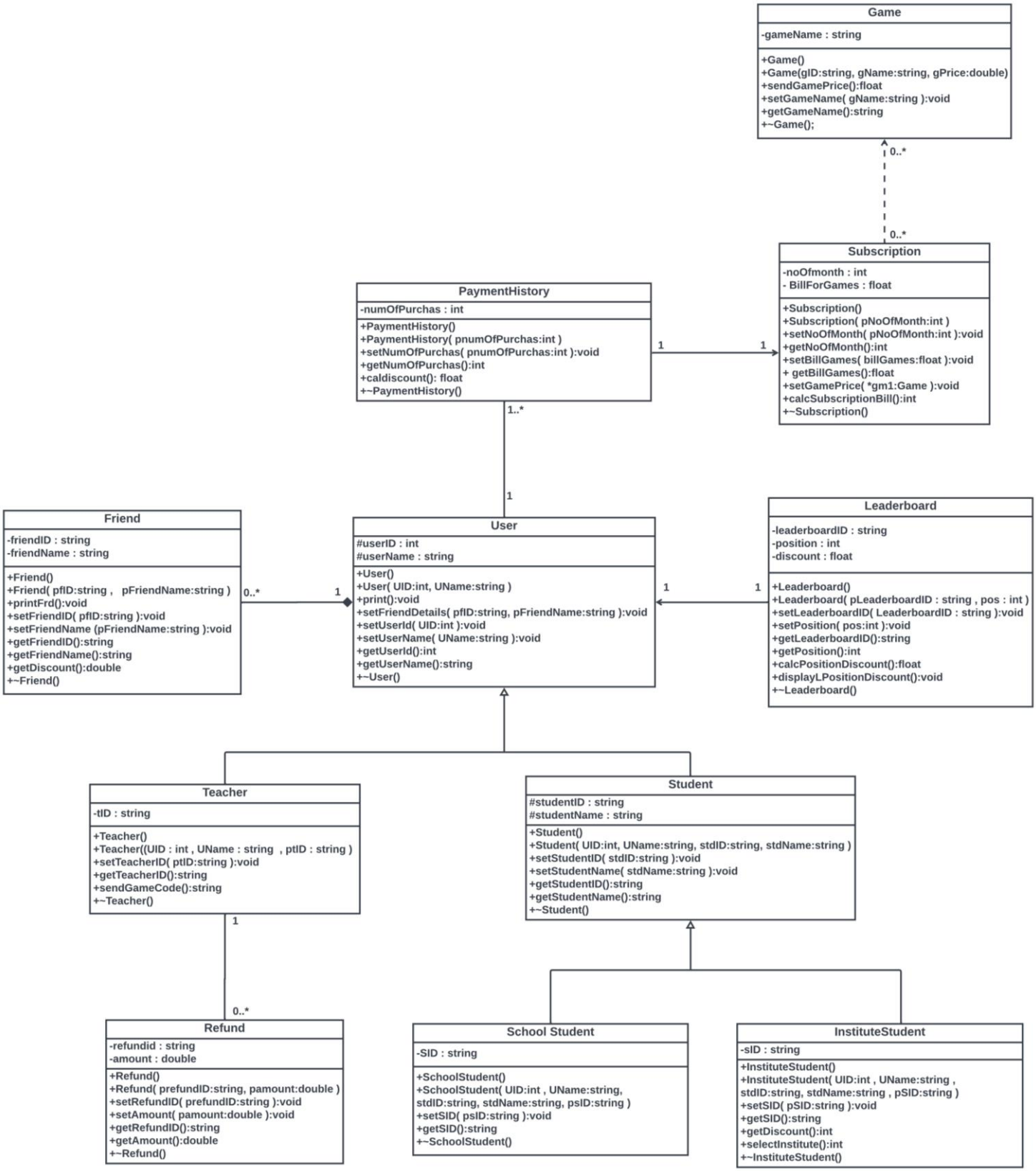
Subscription	
Responsibility	Collaborators
Get Subscription Details	Game
Display Subscription Details	
Calculate Subscription Discount	Game

Leader board	
Responsibility	Collaborators
Get User Position	User
Validate User Position	
Validate Viability for Leaderboard Discount	
Calculate Leaderboard Discount	Subscription

Payment History	
Responsibility	Collaborators
Display Purchase Details	Game
Display Number of Purchases	
Calculate a discount depending on the number of purchases	Game

Refund	
Responsibility	Collaborators
Validate Validity for Refund	Payment History
Calculate Refund Amount	Payment History
Display Refund Amount	

4) Class diagram



5) Individual contribution

Registration No	Name	Contributed classes
IT21244452	D.G.N.U.S.Wijayasiri	• Subscription • Leader board
IT21277054	M.S.Shiraz	• User • Game
IT21271182	R.M.U.V.Rathnayake	• Student • Institute Student
IT21227554	A.K.S.Chandrajith	• Friend • Refund
IT21298080	B.M.V.B.B.Gurusinghe	• Payment History • Teacher • School Student

6) Online education games – Declaration and Implementation

Main.cpp

```
#include <iostream>
#include <cstring>

#include "Friend.h"
#include "Game.h"
#include "Leaderboard.h"
#include "PaymentHistory.h"
#include "Refund.h"
#include "Student.h"
#include "Subscription.h"
#include "Teacher.h"
#include "User.h"
#include "InstituteStudent.h"
#include "SchoolStudent.h"

using namespace std;

int main() {

  //create objects

  //IT21244452 _ D.G.N.U.S.Wijayasiri
  Subscription sb1(30.00 , 2);
  Leaderboard* lead1 = new Leaderboard();

  //IT21277054 _ M.S.Shiraz
      User Us1 (123 , "Shamry Shiraz");
      Game* gm1 = new Game();

  //IT21271182 _ R.M.U.V.Rathnayake
      Student stu1(234 , "Vihang123" , "S234", "Vihangi Rathnayake");
      InstituteStudent* intstd1 = new InstituteStudent();

  //IT21227554 _ A.K.S.Chandrajith
      Friend frd1("f123", "Umayangana Wijayasiri");
      Refund* ref1 = new Refund();

  //IT21298080 _ B.M.V.B.B.Gurusinghe
      PaymentHistory payHis1(30);
      Teacher* tech1 =  new Teacher();
      SchoolStudent schstd1( 123 , "Vishwa123" , "S123" , "Vishwa
Gurusinghe" , "Ss123") ;

  //create variables
      int uid;
      string uName,fid,fName,gname, insName, tID;
      string stuTy , sclIns , gamecode;
      char temp ;
      int Smonth , institute , game , noOfFriends ;
      float insDis , billForSubscription;
      string lid , fID ;
      int lposition;
      double lDisk1;
```

```
  //Implementation of the program


      cout << "\n\t Welcome to PIXXEL!\tツ\n\n" << endl;

  //get user details

      cout << " ---  Enter your details  --- \n" << endl;
      cout<<"Enter User ID : ";
      cin>>uid;

      cout<<"Enter User Name : ";
        cin.ignore();
      getline(cin ,uName);

      cout<<"Enter Friend ID : ";
      cin>>fid;

      cout<<"Enter Friend Name : ";
      cin.ignore();
      getline(cin ,fName);

      User* u1 = new User(uid,uName);
      u1->setFriendDetails(fid,fName);

      u1->print();



  cout << "-----------------------------------------"<< endl;

  //End of user details



  //selection of a game
  //IT21277054 _ M.S.Shiraz

      cout << "\n ---  Select a Game  --- \n" << endl;

      cout <<"1 - Mario Maths Runner \n2 - English with Sonic \n3 - Kirby's
World Ma\t\n\nSelect one : ";
    cin >> game;
  while(true){
    if(game == 1 ){
      gname = "Mario Maths Runner";
      break;
    }
    else if(game == 2 ){
      gname = "English with Sonic";
      break;

    }
    else if(game == 3 ){
      gname = "Kirby's World Ma";
      break;
    }
    else{
      cout<< "\nInvalid Enter again : ";
```

```
    cin >> game;
    continue;
  }

}

    Game* g1 = new Game(gname);
    g1->sendGamePrice();

    cout<<"Price of the selected game : $"<<g1->sendGamePrice()<<endl;

cout << "-----------------------------------------"<< endl;

//End game



//select the subscription
//IT21244452 _ D.G.N.U.S.Wijayasiri

    cout << "\n ---  Subscription  --- \n" << endl;

  cout << "Enter the No of months : ";
    cin >> Smonth;

  float gprice = g1->sendGamePrice();


    Subscription sub1(45.00 , 3);
    billForSubscription = sub1.calcSubscriptionBill();
    cout << "Bill for Subscription : " << billForSubscription << endl;

  cout << "-----------------------------------------"<< endl;

//End of the subscription




// get details of friends
//IT21227554 _ A.K.S.Chandrajith

    cout << "\n ---  Enter friend details  --- \n" << endl;

    cout << "Enter number of Friends : ";
    cin >> noOfFriends ;
  cout << endl;
    Friend * f[noOfFriends];

    for ( int i = 0 ; i < noOfFriends ; i++ ){
    cout << "Enter Friend ID : ";
    cin >> fID;

    cout << "Enter Friend Name : ";
    cin >> fName;
    cout << endl << endl;

    f[i] = new Friend(fID , fName);
    }
```

```
        cout << "You have "<< noOfFriends ;

        if(noOfFriends > 0 ) {

              cout << " Friends. Their names : \n";

              for ( int i = 0 ; i < noOfFriends ; i++ ){
              cout << "\t" << f[i]->getFriendName() << endl;
              }
        }

        cout << "\n Based on your number of friends ";

        if (noOfFriends == 0 ){
        cout << "you get 0% discount ! \n"<< endl;
        }
        else if (noOfFriends >= 5){
        cout << "you get "<< noOfFriends*1 << "% discount ! \n"<< endl;
        }
        else {
        cout << "you get "<< noOfFriends*1 << "% discount ! \n"<< endl;
        }



      cout << "-----------------------------------------"<< endl;

    //End of Friend



    //Selection of the user type
    //IT21271182 _ R.M.U.V.Rathnayake

        cout << "\n ---  Select your user type  --- \n" << endl;

        cout<< "Are you an academic student?(yes/no)   " ;
        cin.ignore();
        getline(cin ,stuTy);

        if (stuTy == "yes" ){

              cout<< "Are you a School student or Institute student
(school/institute) : " ;
              getline(cin ,sclIns );

              if (sclIns == "school" ){
                    cout << "Enter your teacher's ID : " ;
                    getline(cin ,tID);

                    //get the releveant game ID of the teacher
                    Teacher* teach1 = new Teacher(uid , uName ,tID);

                    gamecode = teach1 -> sendGameCode();

                    cout << "Your game code is : " << gamecode << endl;
              }
              else if (sclIns == "institute" ){

                    //selection of the institute
```

```
                    InstituteStudent* instStd1;
                    insDis = instStd1 -> selectInstitute();

                    if ( insDis == 100 ){
                          cout << "Institute discount = 100% " << endl;
                    }
                    else{
                          cout<< "No institute discount !" << endl;
                    }
              }
        }
        else{
              cout<< "Not continuing as a student! " << endl;
        }

      cout << "-----------------------------------------"<< endl;

    //End student type



    //Leaderboard
    //IT21244452 _ D.G.N.U.S.Wijayasiri

        cout << "\n ---  Select leaderboad details  --- \n" << endl;
        cout << "Enter Leaderboard ID :";
        cin >> lid;

        cout << "Enter the position of Leaderboard :";
        cin >> lposition;

        Leaderboard *l1 =new Leaderboard(lid , lposition);
        l1-> setLeaderboardID(lid);
        l1-> setPosition(lposition);

        l1->getLeaderboardID();
        l1->getPosition();

        cout << "Position Discount : $" << l1->calcPositionDiscount() << endl;


      cout << "-----------------------------------------"<< endl;

    //End of leaderboad


    //calculate the refund
    //IT21244452 _ D.G.N.U.S.Wijayasiri

        Refund r1("f123" , 5);
        cout << "Refund amount : $" << r1.getAmount() << endl;



    //History of the payments
    //IT21298080 _ B.M.V.B.B.Gurusinghe

      cout << "\n ---  Select the number of game (Every 10 purchases gives a 1$
discount)  --- \n" << endl;
```

```
    int nofp;

    cout<<"Enter Number of Purchases :";
    cin>> nofp;

    PaymentHistory* ph = new PaymentHistory(nofp);
    cout<<"Discount Due to Purchasing many Games :"<<ph->caldiscount
()<<"%"<<endl;

    cout << "----------------------------------------"<< endl;

  //end payment history


  //delete dynamic objects
      delete lead1;
      delete gm1;
      delete intstd1;
      delete ref1;
      delete tech1;
      for ( int i = 0 ; i < noOfFriends ; i++ ){
            delete f[i];
      }


}     //end of the program
```

Friend.h - IT21227554 : A.K.S.Chandrajith

```
//IT21227554 _ A.K.S.Chandrajith

#pragma once
#include <iostream>
#include <cstring>
#include <string>
using namespace std;

class Friend{
  private://declaring private attributes
    string fID;
    string fName;

  public://declaring public functions
    Friend();//default constructor
    Friend(string pfID,string pFriendName);//overloaded constructor
    void printFrd();
    void setFriendID(string pfID);
    void setFriendName(string pFriendName);
    string getFriendID();
    string getFriendName();
    double getDiscount();
    ~Friend();//destructor
  };
```

Friend.cpp - IT21227554 : A.K.S.Chandrajith

```cpp
//IT21227554 _ A.K.S.Chandrajith

#pragma once
#include <iostream>
#include <cstring>
#include <string>
#include "Friend.h"
using namespace std;


Friend::Friend(){//implementing the default constructor
	fID = "";
	fName = "";

}

Friend::Friend(string pfID , string pFriendName){//implementing the
overloaded constructor
  fID = pfID;
  fName = pFriendName;

}

void Friend::printFrd(){
  cout << "Friend ID : "<<fID<<"\tName : "<<fName;//printing friend name
}

void Friend::setFriendID(string pfID){
  fID = pfID;//assigning values to private attributes
}

void Friend::setFriendName(string pFriendName){
   fName = pFriendName;//assigning values to private attributes
}

string Friend::getFriendID(){
  return fID;//getting values from private attributes
}

string Friend::getFriendName(){
  return fName ;//getting values from private attributes
}


Friend::~Friend(){//implementing the destructor
  cout << "Destructor runs to Friend class"<< endl;
}
```

Game.h - IT21277054 : M.S.Shiraz

```
//IT21277054 _ M.S.Shiraz

#pragma once
#include <iostream>
#include <string>

using namespace std;

class Game{
  private://declaring private attributes
    string gameName;

  public://declaring public functions
    Game();//default constructor
    Game(string gName);//overloaded constructor
    float sendGamePrice();
    void setGameName(string gName);

    string getGameName();

    ~Game();//destructor

};
```

Game.cpp - IT21277054 : M.S.Shiraz

```
//IT21277054 _ M.S.Shiraz

#include "Game.h"
#include <iostream>
#include <string>
#include <bits/stdc++.h>
using namespace std;

Game::Game(){//implementing the default constructor
    gameName = "";
}

Game::Game(string gName){//implementing the overloaded constructor
    gameName = gName;
}

float Game::sendGamePrice(){
  map<string,float> games {
  {"Mario Maths Runner",39.99},{"English with Sonic",29.99},{"Kirby's World
Map",19.99}//mapping as assosiative array to declare prices to games
  };

  if(gameName == "Mario Maths Runner"){
      return games["Mario Maths Runner"];//returning the game price
  }
  else if(gameName == "English with Sonic"){
      return games["English with Sonic"];//returning the game price
  }
  else if(gameName == "Kirby's World Map"){
      return games["Kirby's World Map"];//returning the game price
  }
  else{
      cout<<"Invalid game name"<<endl;//returning that the value that was
entered was invalid
  }

  return 0;
}


void Game::setGameName(string gName){
   gameName = gName;//assigning values to private attributes
}


string Game::getGameName(){
    return gameName;//getting values from private attributes
}


Game::~Game(){//implementing the destructor
  cout << "Destructor runs to Game class" << endl;//letting the user know
that the destructor was initiated
}
```

InstituteStudent.h - IT21271182 : R.M.U.V.Rathnayake

```
//IT21271182 _ R.M.U.V.Rathnayake

#pragma once
#include "Student.h"
#include <iostream>
#include <string>
using namespace std;

class InstituteStudent : public Student
{
  private://declaring private attributes
       string sID;

  public://declaring public functions
       InstituteStudent();//default constructor
       InstituteStudent(int UID , string UName  ,string stdID,string stdName
, string pSID );//overloaded constructor
       void setSID( string pSID );
       string getSID();
       int getDiscount();
       int selectInstitute();
       ~InstituteStudent();//destructor

};
```

InstituteStudent.cpp - IT21271182 : R.M.U.V.Rathnayake

```
//IT21271182 _ R.M.U.V.Rathnayake

#include "InstituteStudent.h"
#include <iostream>
#include <string>
using namespace std;

InstituteStudent::InstituteStudent(){//implementing the default constructor
    sID = "";
}

InstituteStudent::InstituteStudent(int UID , string UName  ,string stdID,
string stdName , string pSID) : Student (UID , UName ,stdID , stdName)
{//implementing the overloaded constructor
    sID = pSID;
}

void InstituteStudent::setSID( string pSID ){ //assigning values to private
attribute called "SID"
      sID = pSID;
}

string InstituteStudent::getSID(){//getting values from private attributes
    return sID;
}

int InstituteStudent::getDiscount(){//method to return the discount value

      return 100;
}

int InstituteStudent::selectInstitute(){//method to get discounf for a
specific institute
      string insName;
      int discount;

      cout<< "Enter your institute name : " ;
      getline(cin ,insName);

      if ( insName == "SLIIT" || insName == "NSBM" || insName == "ICBT" ||
insName=="NIBM"){
            discount =getDiscount();
      }
      return discount;
}



InstituteStudent::~InstituteStudent(){//implementing the destructor
      cout << "Destructor runs to Institute Student class" << endl;//letting
the user know that the destructor was initiated
}
```

Leaderboard.h - IT21244452 : D.G.N.U.S.Wijayasiri

```
//IT21244452 _ D.G.N.U.S.Wijayasiri

#pragma once
#include <iostream>
#include <string>
#include "User.h"
using namespace std;


class Leaderboard{
	private://declaring private attributes
		string LeaderboardID;
		int position;
		float discount;
		User *u1;

	public://declaring public attributes
		Leaderboard();//default constructor
		Leaderboard(string pLeaderboardID,int pos);//overloaded
constructor
		void setLeaderboardID(string LeaderboardID);
		void setPosition(int pos);
		string getLeaderboardID();
		int getPosition();
		float calcPositionDiscount();
		void displayLPositionDiscount();
		~Leaderboard();//destructor
};
```

Leaderboard.cpp - IT21244452 : D.G.N.U.S.Wijayasiri

```cpp
//IT21244452 _ D.G.N.U.S.Wijayasiri

#pragma once
#include <iostream>
#include <string>
#include "Leaderboard.h"
#include "User.h"

using namespace std;

Leaderboard::Leaderboard(){//implementing the default constructor
   LeaderboardID = "";
}

Leaderboard::Leaderboard(string pLeaderboardID,int pos){//implementing the
overloaded constructor{
    LeaderboardID = pLeaderboardID;
    position = pos;
}

void Leaderboard::setLeaderboardID(string pLeaderboardID){// assign a value to
leaderboardID{
       LeaderboardID = pLeaderboardID ;
}

string Leaderboard::getLeaderboardID(){// take value from leaderboardID{
      return LeaderboardID;
}
void Leaderboard::setPosition(int pos){// assign a value to position
  position = pos;
}
int Leaderboard::getPosition(){// take value from position
  return position;
}

float Leaderboard::calcPositionDiscount(){// calculating discount according to
the leaderboard position
  if (position <= 3){
     return 40 * 0.05;
  }
  else if (position <= 10){
    return 40 * 0.03;
  }
  else if (position <= 20){
    return 40 * 0.02;
  }
}


void Leaderboard::displayLPositionDiscount(){// display discount
    cout << "Leader Board Discount : " << discount << endl;
}


Leaderboard::~Leaderboard(){ // destructor to the leaderboard class
    cout<<"Destructor runs to Leaderboard class"<<endl;//letting the user know
that the destructor was initiated
}
```

PaymentHistory.h - IT21298080 : B.M.V.B.B.Gurusinghe

```
//IT21298080 _ B.M.V.B.B.Gurusinghe

#ifndef NUMOFPURCHAS // Guards to prevent redeclaration
#define NUMOFPURCHAS

#pragma once
#include <iostream>
#define SIZE 10

class User;
class Subscription;

class PaymentHistory
{
	private://declaring private attributes
		int numOfPurchas;
		Subscription *subscrip;
		User *use1[SIZE];


	public://declaring public attributes
		PaymentHistory();//default constructor
		PaymentHistory( int pnumOfPurchas);  //overloaded constructor
		void setNumOfPurchas(int pnumOfPurchas);
		int getNumOfPurchas();
		float caldiscount();
		~PaymentHistory();//destructor
};

#endif
```

PaymentHistory.cpp - IT21298080 : B.M.V.B.B.Gurusinghe

```
//IT21298080 _ B.M.V.B.B.Gurusinghe

#pragma once
#include "PaymentHistory.h"
#include"Subscription.h"
#include<iostream>
using namespace std;


PaymentHistory::PaymentHistory(){//implementing the default constructor
     numOfPurchas = 0;
}

PaymentHistory::PaymentHistory(int pnumOfPurchas ){//implementing the
overloaded constructor
     numOfPurchas = pnumOfPurchas;
}

float PaymentHistory::caldiscount(){

    int newdis = 10,disamount = 0;

    for(int i = 0; numOfPurchas >= i;i++){

    if(i == newdis){
      disamount += 1;
      newdis += 10;
    }
  }

    return disamount;

}


void PaymentHistory::setNumOfPurchas(int NumofPurchas){//assigning values to
private attributes
    numOfPurchas = numOfPurchas;
}


int PaymentHistory::getNumOfPurchas(){//getting values from private
attributes
    return numOfPurchas;
}

PaymentHistory::~PaymentHistory(){//implementing the destructor
    cout << "Destructor runs to PaymentHistory class" << endl;//letting the
user know that the destructor was initiated
}
```

Refund.h - IT21227554 : A.K.S.Chandrajith

```
//IT21227554 _ A.K.S.Chandrajith

#ifndef REFUND// Guards to prevent redeclaration
#define REFUND

#pragma once
#include<iostream>
// #include"Teacher.h"

#define SIZE 10

class Teacher;

using namespace std;

class Refund{
      private://declaring private attributes
            string refundID;
            double amount;
    Teacher *t1[SIZE];
      public://declaring public attributes
            Refund();//default constructor
            Refund( string prefundID , double pamount);//overloaded
constructor
            void setRefundID( string prefundID);
            void setAmount( double pamount );
            string getRefundID();
            double getAmount();
            ~Refund();//destructor
};

#endif
```

Refund.cpp - IT21227554 : A.K.S.Chandrajith

```
//IT21227554 _ A.K.S.Chandrajith

#pragma once
#include <iostream>
#include <cstring>
#include "Refund.h"
#include"Teacher.h"
#include <string>

using namespace std;

Refund::Refund()//implementing the default constructor
{
	refundID = "";
	amount = 0;
}

Refund::Refund( string prefundID , double pamount )//implementing the
overloaded constructor
{
	refundID = prefundID;
	amount = pamount;
}

void Refund::setRefundID( string prefundID )//assigning values to private
attributes
{
	refundID = prefundID;
}

void Refund::setAmount( double pamount )//assigning values to private
attributes
{
	amount = pamount;
}

string Refund::getRefundID()//getting values from private attributes
{
	return refundID;
}

double Refund::getAmount()//getting values from private attributes
{
	return amount;
}

Refund::~Refund()//implementing the destructor
{
  cout << "Destructor runs to Refund class" << endl;//letting the user know
that the destructor was initiated
}
```

SchoolStudent.h - IT21298080 : B.M.V.B.B.Gurusinghe

```
//IT21298080 _ B.M.V.B.B.Gurusinghe

#include <iostream>
using namespace std;
#include "Student.h"

class SchoolStudent : public Student{//inheritance
private://declaring private attributes
  string sID;

public://declaring public attributes
  SchoolStudent();//default constructor
  SchoolStudent (int UID , string UName  ,string stdID,string stdName ,
string psID);//overloaded constructor
  void setSID(string psID);
  string getSID();
  ~SchoolStudent();//destructor
};
```

SchoolStudent.cpp - IT21298080 : B.M.V.B.B.Gurusinghe

```
//IT21298080 _ B.M.V.B.B.Gurusinghe

#include <iostream>
#include <cstring>
#include "SchoolStudent.h"
using namespace std;

SchoolStudent::SchoolStudent()//implementing the default constructor
{
	sID = "";
}

SchoolStudent::SchoolStudent (int UID , string UName  ,string stdID,string
stdName , string psID) : Student (UID , UName ,stdID, stdName)//implementing
the overloaded constructor
{
	sID = psID;
}

void SchoolStudent::setSID(string psID)//assigning values to private
attributes
{
	sID = psID;
}

string SchoolStudent::getSID()//getting values from private attributes
{
	return sID;
}

SchoolStudent::~SchoolStudent()//implementing the destructor
{
	cout << "Destructor runs to School Student class" << endl;//letting
the user know that the destructor was initiated

}
```

School.h - IT21271182 : R.M.U.V.Rathnayake

```
//IT21271182 _ R.M.U.V.Rathnayake

#pragma once
#include <iostream>
#include "User.h"

class Student : public User{
  protected:
    string studentID;
    string studentName;

  public://declaring public attributes
    Student();//default constructor
    Student (int UID , string UName ,string stdID,string
stdName);//overloaded constructor
    void setStudentID(string stdID);
    void setStudentName(string stdName);

    string getStudentID();
    string getStudentName();

    ~Student();//destructor


};
```

School.cpp - IT21271182 : R.M.U.V.Rathnayake

```
//IT21271182 _ R.M.U.V.Rathnayake

#include <iostream>
#include "Student.h"
#include <cstring>
using namespace std;

Student::Student(){//implementing the default constructor
  studentID = "";
  studentName = "";
}

Student::Student(int UID , string UName ,string stdID,string stdName) :
User(UID , UName){//implementing the overloaded constructor
  studentID = stdID;
  studentName = stdName;
}


void Student::setStudentID(string stdID){//assigning values to private
attribute
  studentID = stdID;

}

void Student::setStudentName(string stdName){//assigning values to private
attribute
  studentName = stdName;
}

string Student::getStudentID(){//getting values from private attribute
  return studentID;
}

string Student::getStudentName(){//getting values from private attribute
  return studentName;
}

Student::~Student(){//implementing the destructor

   cout << "Destructor runs to Student class" << endl;//letting the user know
that the destructor was initiated
}
```

Subscription.h - IT21244452 : D.G.N.U.S.Wijayasiri

```
//IT21244452 _ D.G.N.U.S.Wijayasiri

#include <iostream>
#include "Game.h"

class Subscription{
  private://declaring private attributes
    int noOfMonth ;
    float BillForGames;

  public://declaring public attributes
    Subscription();//default constructor
    Subscription(float bill ,int pNoOfMonth);//overloaded constructor
    void setNoOfMonth(int pNoOfMonth);
    int getNoOfMonth();
    void setBillGames(float billGames);
    float getBillGames();
    void setGamePrice(Game *gm1);
    int calcSubscriptionBill();
    ~Subscription();//destructor
};
```

Subscription.cpp - IT21244452 : D.G.N.U.S.Wijayasiri

```
//IT21244452 _ D.G.N.U.S.Wijayasiri

#include <iostream>
#include <cstring>
#include "Subscription.h"
#include "Game.h"
#include "PaymentHistory.h"
using namespace std;


Subscription::Subscription(){//implementing the default constructor
  BillForGames = 0;
  noOfMonth = 0;
}

Subscription::Subscription(float bill , int pNoOfMonth){//implementing the
overloaded constructor
    BillForGames = bill;
    noOfMonth = pNoOfMonth;
}


void Subscription::setNoOfMonth(int pNoOfMonth){//assigning values to private
attributes
  noOfMonth = pNoOfMonth;
}

void Subscription::setGamePrice(Game *gm1){

}

void Subscription::setBillGames(float billGames)
{
  BillForGames = billGames;
}

int Subscription::getNoOfMonth(){//getting values from private attributes
  return noOfMonth;
}

float Subscription::getBillGames()
{
  return BillForGames;
}



int Subscription::calcSubscriptionBill() //Calculation
{
  return BillForGames * noOfMonth;
}

Subscription::~Subscription(){//implementing the destructor
  cout << "Destructor runs to Subscription class" << endl;//letting the user
know that the destructor was initiated
}
```

Teacher.h - IT21298080 : B.M.V.B.B.Gurusinghe

```
//IT21298080 _ B.M.V.B.B.Gurusinghe

#ifndef TEACHER
#define TEACHER

#pragma once
#include <iostream>
#include <bits/stdc++.h>
#include "User.h"
// #include "Refund.h"

class Refund;

class Teacher : public User{
	private://declaring private attributes
		string tID;
	Refund *ref;

	public://declaring public attributes
		Teacher();//default constructor
		Teacher(int UID , string UName , string ptID);//overloaded
constructor
	string sendGameCode();
		void setTeacherID(string ptID);
		string getTeacherID();
		~Teacher();//destructor
};

#endif
```

Teacher.cpp - IT21298080 : B.M.V.B.B.Gurusinghe

```
//IT21298080 _ B.M.V.B.B.Gurusinghe

#pragma once
#include <iostream>
#include <cstring>
#include <string>
#include "Teacher.h"
using namespace std;

Teacher::Teacher(){//implementing the default constructor
	tID = " ";
}

Teacher::Teacher(int UID , string UName , string ptID) : User(UID , UName){
//implementing the overloaded constructor
	tID =  ptID;
}

string Teacher::sendGameCode(){
    map<string,string>gameCode{{"T001","AB16"},
{"T002","BC25"},{"T003","CD34"}};//mapping as assosiative array to declare
code to game

	if(tID == "T001" ){
		return gameCode["T001"];//returning the TeacherID
	}
	else if(tID == "T002"){
	return gameCode["T002"];//returning the TeacherID
	}
	else if(tID == "T003"){
		return gameCode["T003"];//returning the TeacherID
	}
	else{
		return "Invaid TeacherID";//returning that the value that was
entered was invalid
	}
}

void Teacher::setTeacherID(string ptID){//assigning values to private
attributes
	tID = ptID;
}

string Teacher::getTeacherID(){//getting values from private attributes
	return tID;
}

Teacher::~Teacher(){//implementing the destructor
	cout<<"Destructor runs to Teacher class"<<endl;//letting the user know
that the destructor was initiated
}
```

User.h - IT21277054 : M.S.Shiraz

```
//IT21277054 _ M.S.Shiraz

#pragma once
#include <iostream>
#include <string>
#include "Friend.h"
// #include "PaymentHistory.h"
using namespace std;

class PaymentHistory;

class User{
	protected:
		int userID;
		string userName;
	  PaymentHistory *payHis1;

    private://declaring private attributes
	  Friend *frd1;

	  public://declaring public attributes
	  User();//default constructor
		User(int UID , string UName);
	  void print();
	  void setFriendDetails(string pfID,string pFriendName);//overloaded
constructor
		void setUserId(int UID);
		void setUserName(string UName);
		int getUserId();
		string getUserName();
		~User();//destructor

};
```

User.cpp - IT21277054 : M.S.Shiraz

```cpp
//IT21277054 _ M.S.Shiraz

#pragma once
#include <iostream>
#include <string>
#include "User.h"
#include "Friend.h"
#include "Refund.h"
using namespace std;

User::User(){//implementing the default constructor
  userID = 0;
  userName = "";
}

User::User(int UID , string UName){//implementing the overloaded constructor
  userID = UID;
  userName = UName;
  frd1 = new Friend;
}

void User::setFriendDetails(string pfID,string pFriendName){//Implementing
attributes in friends(composition)
  frd1->setFriendID(pfID);
  frd1->setFriendName(pFriendName);
}

void User::print(){
  cout<< "\nYour ID : " <<userID<< "\tName : " <<userName<<endl;//printing
UserID and User name
  frd1->printFrd();
  cout<<endl;
}

void User::setUserId(int UID){//assigning values to private attributes
      userID=UID;
}

void User::setUserName(string UName){//assigning values to private attributes
      userName = UName;
}

int User::getUserId(){//getting values from private attributes
      return userID;
}

string User::getUserName(){//getting values from private attributes
      return userName;
}

User::~User(){//implementing the destructor
      cout << "Destructor runs to User class" << endl;
}
```