**Assignment 03**

**Name: Muhammad Musa Roll no: l1f23bsai0039 SEC:AC-2**

**OOP Section AC 2**

**Marks: 10+10+5=25**

**-All students of OOP Section AC 2 are requested to submit the solution on portal before/on**

**14th, Jan, 2025.**

**-No submission will be accepted after due date. This is individual assignment.**

**-on front page must mention your full name, registration and section.**

**Note:**

**Past the solution of code in one .doc or .pdf . Don’t not send .cpp files.**

**Task 1:**

Student class inherits Person class. Declare one object of each class in **main** by calling their setter and getters. Also define the appropriate constructor and destructors.

***Note:***

- ID, is automatically assigned to newly created object by calling default Constructor

-abstract at least three attributes/data members of each class.

-each data member of class has its own setter and getter

-all getter must be constant member function

**code:**

#include <iostream>

#include <string>

using namespace std;

class Person {

string address;

string name;

char gender;

string father\_name;

int age;

public:

Person() : name("unknown"), age(0), address("unknown"), gender('U'), father\_name("unknown") {}

Person(string name, int age, string address, char gender, string father\_name)

: name(name), age(age), address(address), gender(gender), father\_name(father\_name) {

setname(name);

setage(age);

setaddress(address);

setfather\_name(father\_name);

}

void setname(string name) {

if (name.length() >= 3) {

this->name = name;

}

else {

cout << "INVALID INPUT. ENTER VALID INPUT (LENGTH SHOULD BE GREATER THAN OR EQUAL TO 3): ";

cin >> name;

setname(name);

}

}

void setfather\_name(string father\_name) {

if (father\_name.length() >= 3) {

this->father\_name = father\_name;

}

else {

cout << "INVALID INPUT. ENTER VALID INPUT (LENGTH SHOULD BE GREATER THAN OR EQUAL TO 3): ";

cin >> father\_name;

setfather\_name(father\_name);

}

}

void setage(int age) {

if (age >= 18 && age <= 50) {

this->age = age;

}

else {

cout << "INVALID INPUT. ENTER VALID INPUT (AGE MUST BE BETWEEN 18 AND 50): ";

cin >> age;

setage(age);

}

}

void setaddress(string address) {

if (address.length() >= 8) {

this->address = address;

}

else {

cout << "INVALID INPUT. ENTER VALID INPUT (ADDRESS LENGTH SHOULD BE GREATER THAN OR EQUAL TO 8): ";

cin >> address;

setaddress(address);

}

}

void setgender(char gender) {

if (gender == 'm' || gender == 'M' || gender == 'f' || gender == 'F') {

this->gender = gender;

}

else {

cout << "INVALID INPUT. ENTER VALID INPUT (GENDER MUST BE 'M' OR 'F'): ";

cin >> gender;

setgender(gender);

}

}

string getname() const { return name; }

int getage() const { return age; }

string getaddress() const { return address; }

string getfather\_name() const { return father\_name; }

char getgender() const { return gender; }

void displayA() const {

cout << "NAME: " << getname() << endl;

cout << "AGE: " << getage() << endl;

cout << "ADDRESS: " << getaddress() << endl;

cout << "FATHER'S NAME: " << getfather\_name() << endl;

cout << "GENDER: " << getgender() << endl;

}

~Person() {

cout << "DESTRUCTOR CALLED FOR PERSON CLASS." << endl;

}

};

class Student : public Person {

int studentid;

double gpa;

static int k;

public:

Student() : Person(), gpa(0.0), studentid(++k) {}

Student(string name, int age, string address, double gpa)

: Person(name, age, address, 'U', "unknown"), gpa(gpa), studentid(++k) {}

void setid(int id) {

studentid = id;

}

void setgpa(double gpa) {

if (gpa >= 0.0 && gpa <= 4.0) {

this->gpa = gpa;

}

else {

cout << "INVALID INPUT. ENTER VALID INPUT (GPA MUST BE BETWEEN 0.0 AND 4.0): ";

cin >> gpa;

setgpa(gpa);

}

}

int getid() const { return studentid; }

double getgpa() const { return gpa; }

void displayB() const {

displayA();

cout << "STUDENT ID: " << getid() << endl;

cout << "GPA: " << getgpa() << endl;

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

}

~Student() {

cout << "DESTRUCTOR CALLED FOR STUDENT CLASS." << endl;

}

};

int Student::k = 0;

int main() {

Person p;

cout << "DEFAULT VALUES OF PERSON CLASS:" << endl;

p.displayA();

Student s;

cout << "DEFAULT VALUES OF STUDENT CLASS:" << endl;

s.displayB();

string name, address, father\_name;

int age;

char gender;

cout << "ENTER NAME: ";

getline(cin, name);

p.setname(name);

cout << "ENTER AGE: ";

cin >> age;

p.setage(age);

cin.ignore();

cout << "ENTER ADDRESS: ";

getline(cin, address);

p.setaddress(address);

cout << "ENTER FATHER'S NAME: ";

getline(cin, father\_name);

p.setfather\_name(father\_name);

cout << "ENTER GENDER (M/F): ";

cin >> gender;

p.setgender(gender);

cin.ignore();

cout << "PARAMETERIZED VALUES OF PERSON CLASS:" << endl;

p.displayA();

double gpa;

cout << "ENTER NAME OF STUDENT: ";

getline(cin, name);

s.setname(name);

cout << "ENTER AGE OF STUDENT: ";

cin >> age;

s.setage(age);

cin.ignore();

cout << "ENTER ADDRESS OF STUDENT: ";

getline(cin, address);

s.setaddress(address);

cout << "ENTER GPA: ";

cin >> gpa;

s.setgpa(gpa);

cout << "PARAMETERIZED VALUES OF STUDENT CLASS:" << endl;

s.displayB();

return 0;

}

**Task 2:**

Declare an array of size 100 of class student (***according to task 1***) and call their setter /getters of class. Also define a function named void showStudentDeatils(), which shows the details of those students who are female and have age between 18 and 20 years.

**Note:**

-The setter/getters of child class should call the setter/getter of parent class inside its body.

- all setters must be normal member function while all getters must be inline

**code:**

#include <iostream>

#include <string>

using namespace std;

class Person {

private:

string name;

int age;

string address;

static int personIDCounter;

int id;

public:

Person() : id(++personIDCounter) {}

Person(const string& n, int a, const string& addr) : name(n), age(a), address(addr), id(++personIDCounter) {}

~Person() {}

inline int getId() const { return id; }

inline string getName() const { return name; }

void setName(const string& n) { name = n; }

inline int getAge() const { return age; }

void setAge(int a) {

if (a >= 0) {

age = a;

}

else {

cout << "INVALID INPUT. AGE MUST BE NON-NEGATIVE. Please enter again: ";

cin >> a;

setAge(a);

}

}

inline string getAddress() const { return address; }

void setAddress(const string& addr) { address = addr; }

};

int Person::personIDCounter = 0;

class Student : public Person {

private:

string program;

int year;

string studentID;

char gender;

static int studentIDCounter;

public:

Student() : studentID("S" + to\_string(++studentIDCounter)) {}

Student(const string& n, int a, const string& addr, const string& prog, int y, char g)

: Person(n, a, addr), program(prog), year(y), studentID("S" + to\_string(++studentIDCounter)), gender(g) {}

~Student() {}

inline string getProgram() const { return program; }

void setProgram(const string& prog) { program = prog; }

inline int getYear() const { return year; }

void setYear(int y) {

if (y > 0 && y>=1980 && y<=2025) {

year = y;

}

else {

cout << "INVALID INPUT. YEAR MUST BE POSITIVE. Please enter again: ";

cin >> y;

setYear(y);

}

}

inline string getStudentID() const { return studentID; }

char getGender() const { return gender; }

void setGender(char g) {

if (g == 'M' || g == 'm' || g == 'F' || g == 'f') {

gender = g;

}

else {

cout << "INVALID INPUT. GENDER MUST BE 'M' OR 'F'. Please enter again: ";

cin >> g;

setGender(g);

}

}

void setName(const string& n) { Person::setName(n); }

void setAge(int a) { Person::setAge(a); }

void setAddress(const string& addr) { Person::setAddress(addr); }

};

int Student::studentIDCounter = 0;

void showStudentDetails(Student arr[], int size) {

for (int i = 0; i < size; i++) {

if (arr[i].getGender() == 'F' && arr[i].getAge() >= 18 && arr[i].getAge() <= 20) {

cout << "Name: " << arr[i].getName() << ", Age: " << arr[i].getAge()

<< ", Address: " << arr[i].getAddress() << ", Program: " << arr[i].getProgram()

<< ", Year: " << arr[i].getYear() << ", ID: " << arr[i].getStudentID() << endl;

}

}

}

int main() {

Student students[100];

int numberOfStudents;

cout << "Enter the number of students: ";

cin >> numberOfStudents;

cin.ignore();

for (int i = 0; i < numberOfStudents; i++) {

string name, address, program;

int age, year;

char gender;

cout << "Enter details for student " << (i + 1) << ":\n";

cout << "Name: ";

getline(cin, name);

students[i].setName(name);

cout << "Age: ";

cin >> age;

students[i].setAge(age);

cin.ignore();

cout << "Address: ";

getline(cin, address);

students[i].setAddress(address);

cout << "Program: ";

getline(cin, program);

students[i].setProgram(program);

cout << "Year: ";

cin >> year;

students[i].setYear(year);

cin.ignore();

cout << "Gender (M/F): ";

cin >> gender;

students[i].setGender(gender);

cin.ignore();

cout << endl;

}

showStudentDetails(students, numberOfStudents);

return 0;

}

**Task 3:**

An Employee class has data members such as ID, Name,Gender,Salary. Write a program in C++ to add the salaries of two Employees by overloading the binary operator ( + ).

**code:**

#include <iostream>

#include <string>

using namespace std;

class Employee {

private:

int id;

string name;

string gender;

double salary;

public:

Employee() : id(0), salary(0.0) {}

Employee(int i, const string& n, const string& g, double s) : id(i), name(n), gender(g), salary(s) {}

double getSalary() const { return salary; }

void setSalary(double s) { salary = s; }

Employee operator+(const Employee& e) {

Employee temp;

temp.salary = this->salary + e.salary;

return temp;

}

void display() const {

cout << "ID: " << id << ", Name: " << name << ", Gender: " << gender << ", Salary: " << salary << endl;

}

};

int main() {

Employee e1(1, "jahan ara", "Female", 5000.0);

Employee e2(2, "umar", "Male", 7000.0);

Employee total = e1 + e2;

cout << "Employee 1: ";

e1.display();

cout << "Employee 2: ";

e2.display();

cout << "Total Salary: " << total.getSalary() << endl;

return 0;

}