**INAssignment 4 Inheritance**

1. Create a base class Employee(Name, Id). Inherit a class Manager(Department,Salary) from Employee. Inherit a class SalesManager(Incentive) from Manager. Use appropriate parameterized constructors for each class.

#include <iostream>

#include <string>

using namespace std;

class Employee {

 public:

  Employee(string name, int id) : name\_(name), id\_(id) {}

  string getName() const { return name\_; }

  int getId() const { return id\_; }

 private:

  string name\_;

  int id\_;

};

class Manager : public Employee {

 public:

  Manager(string name, int id, string department, int salary)

      : Employee(name, id), department\_(department), salary\_(salary) {}

  string getDepartment() const { return department\_; }

  int getSalary() const { return salary\_; }

 private:

  string department\_;

  int salary\_;

};

class SalesManager : public Manager {

 public:

  SalesManager(string name, int id, string department, int salary, int incentive)

      : Manager(name, id, department, salary), incentive\_(incentive) {}

  int getIncentive() const { return incentive\_; }

 private:

  int incentive\_;

};

int main() {

  Employee e("Alice", 12345);

  Manager m("Bob", 54321, "Marketing", 100000);

  SalesManager sm("Charlie", 67890, "Sales", 110000, 5000);

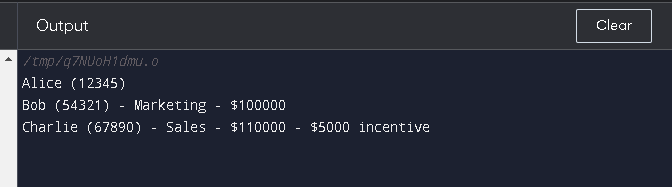
  cout << e.getName() << " (" << e.getId() << ")" << endl;

  cout << m.getName() << " (" << m.getId() << ") - " << m.getDepartment() << " - $" << m.getSalary() << endl;

  cout << sm.getName() << " (" << sm.getId() << ") - " << sm.getDepartment() << " - $" << sm.getSalary() << " - $" << sm.getIncentive() << " incentive" << endl;

  return 0;

}



1. Derive a class Result(Percentage,Grade) from two base classes namely Student(Rollno, Name) and Marks(Comp, Maths, Elec, Stats). Calculate the percentage using a member function calc\_per(), display the grade using disp\_grade() member function. Display three toppers out of n students.

#include <iostream>

#include <string>

class Student

{

  public:

    Student(int rollno, std::string name) : rollno\_(rollno), name\_(name) {}

    int rollno() const { return rollno\_; }

    std::string name() const { return name\_; }

  private:

    int rollno\_;

    std::string name\_;

};

class Marks

{

  public:

    Marks(int comp, int maths, int elec, int stats) : comp\_(comp), maths\_(maths), elec\_(elec), stats\_(stats) {}

    int comp() const { return comp\_; }

    int maths() const { return maths\_; }

    int elec() const { return elec\_; }

    int stats() const { return stats\_; }

  private:

    int comp\_;

    int maths\_;

    int elec\_;

    int stats\_;

};

class Result : public Student, public Marks

{

  public:

    Result(int rollno, std::string name, int comp, int maths, int elec, int stats)

        : Student(rollno, name), Marks(comp, maths, elec, stats) {}

    double calc\_per()

    {

        return (comp() + maths() + elec() + stats()) / 4.0;

    }

    std::string disp\_grade()

    {

        double percentage = calc\_per();

        if (percentage >= 90)

            return "A+";

        else if (percentage >= 80)

            return "A";

        else if (percentage >= 70)

            return "B+";

        else if (percentage >= 60)

            return "B";

        else if (percentage >= 50)

            return "C+";

        else if (percentage >= 40)

            return "C";

        else

            return "D";

    }

};

int main()

{

    // Create an array of Results for n students

    const int n = 5;

    Result students[n] = {

        Result(1, "Alice", 90, 85, 80, 95),

        Result(2, "Bob", 70, 75, 80, 85),

        Result(3, "Charlie", 60, 65, 70, 75),

        Result(4, "Dave", 50, 55, 60, 65),

        Result(5, "Eve", 40, 45, 50, 55)};

    // Find the top three students by sorting the array

    for (int i = 0; i < n - 1; i++)

    {

        for (int j = 0; j < n - i - 1; j++)

        {

            if (students[j].calc\_per() < students[j + 1].calc\_per())

            {

                std::swap(students[j], students[j + 1]);

            }

        }

    }

     // Print the top three students

  std::cout << "Top three students:" << std::endl;

  for (int i = 0; i < 3; i++) {

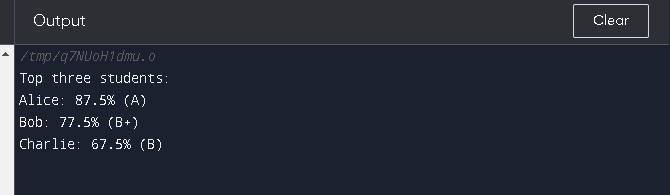
    std::cout << students[i].name() << ": " << students[i].calc\_per() << "%"

         << " (" << students[i].disp\_grade() << ")" <<std:: endl;

  }

  return 0;

}



3.Create a class Person(Name, Aadhar\_no). Derive classes Doctor(Speciality, place\_of\_hosp ) and Teacher(Subject, college\_name). Use appropriate constructors and destructors.

#include <iostream>

#include <string>

using namespace std;

class Person {

 public:

  Person(string name, string aadhar\_no) : name\_(name), aadhar\_no\_(aadhar\_no) {}

  virtual ~Person() {}

  string GetName() const { return name\_; }

  string GetAadharNo() const { return aadhar\_no\_; }

 private:

  string name\_;

  string aadhar\_no\_;

};

class Doctor : public Person {

 public:

  Doctor(string name, string aadhar\_no, string speciality, string place\_of\_hosp)

      : Person(name, aadhar\_no), speciality\_(speciality), place\_of\_hosp\_(place\_of\_hosp) {}

  ~Doctor() {}

  string GetSpeciality() const { return speciality\_; }

  string GetPlaceOfHosp() const { return place\_of\_hosp\_; }

 private:

  string speciality\_;

  string place\_of\_hosp\_;

};

class Teacher : public Person {

 public:

  Teacher(string name, string aadhar\_no, string subject, string college\_name)

      : Person(name, aadhar\_no), subject\_(subject), college\_name\_(college\_name) {}

  ~Teacher() {}

  string GetSubject() const { return subject\_; }

  string GetCollegeName() const { return college\_name\_; }

 private:

  string subject\_;

  string college\_name\_;

};

int main() {

  Person p("John", "1234567890");

  cout << "Name: " << p.GetName() << ", Aadhar No: " << p.GetAadharNo() << endl;

  Doctor d("Jane", "0987654321", "Surgery", "St. Mary's Hospital");

  cout << "Name: " << d.GetName() << ", Aadhar No: " << d.GetAadharNo()

       << ", Speciality: " << d.GetSpeciality()

       << ", Place of Hospital: " << d.GetPlaceOfHosp() << endl;

  Teacher t("Mike", "1231231230", "Physics", "MIT");

  cout << "Name: " << t.GetName() << ", Aadhar No: " << t.GetAadharNo()

       << ", Subject: " << t.GetSubject() << ", College: " << t.GetCollegeName()

       << endl;

  return 0;

}

