|  |  |
| --- | --- |
| **Name** | ADITI RAO |
| **UID no.** | 202220003 |
| **Experiment No.** | 5 |

|  |  |
| --- | --- |
| **AIM:** | **Program on Inheritance: Implement a Program to demonstrate single, multilevel Inheritance** |
| **Program 1** | |
| **PROBLEM STATEMENT:** | Create a class called Employee with data members emp\_id and name. Write a parameterized constructor and getter methods for the data members of the Employee class. Next, create two classes FullTimeEmployee and PartTimeEmployee derived from the Employee class using public mode of inheritance. FullTime Employee receives salary monthly and has data members basic salary, dearness allowance and HRA. Total monthly salary is equal to sum of all the above data members. Write a calculateSalary() method for computing a full time employee’s monthly salary. PartTime Employees are paid on an hourly basis with the rate of payment per hour and the total number of hours worked stored as data members. Write a calculateWages() method for computing the daily wages of a given Part Time Employee assuming that the employee has worked for a period of 30 days. Write appropriate constructors for the derived classes. Also write a menu-driven main() function which simulates the above scenario. |
| **ALGORITHM:** | CLASS Employee  PROTECTED:  DECLARE emp\_id as integer  DECLARE name as string  PUBLIC:  FUNCTION Employee(int id, string nm)  emp\_id = id  name = nm  END FUNCTION  FUNCTION getEmpId()  RETURN emp\_id  END FUNCTION  FUNCTION getName()  RETURN name  END FUNCTION  END CLASS  CLASS FullTimeEmployee INHERITS Employee  PRIVATE:  DECLARE basic\_salary as double  DECLARE dearness\_allowance as double  DECLARE hra as double  PUBLIC:  FUNCTION FullTimeEmployee(int id, string nm, double bs, double da, double hr)  CALL Employee(id, nm)  basic\_salary = bs  dearness\_allowance = da  hra = hr  END FUNCTION  FUNCTION calculateSalary()  RETURN basic\_salary + dearness\_allowance + hra  END FUNCTION  END CLASS  CLASS PartTimeEmployee INHERITS Employee  PRIVATE:  DECLARE rate\_per\_hour as double  DECLARE total\_hours as integer  PUBLIC:  FUNCTION PartTimeEmployee(int id, string nm, double rph, int th)  CALL Employee(id, nm)  rate\_per\_hour = rph  total\_hours = th  END FUNCTION  FUNCTION calculateWages()  RETURN rate\_per\_hour \* total\_hours \* 30  END FUNCTION  END CLASS  FUNCTION main()  DECLARE id as integer  DECLARE total\_hours as integer  DECLARE bs as double  DECLARE da as double  DECLARE hr as double  DECLARE rate\_per\_hour as double  DECLARE employee\_name as string  PRINT "Enter Full Time Employee details:"  PRINT "Enter employee ID: "  READ id  IGNORE\_NEWLINE  PRINT "Enter employee name: "  READ employee\_name  PRINT "Enter basic salary: "  READ bs  PRINT "Enter dearness allowance: "  READ da  PRINT "Enter house rent allowance: "  READ hr  CALL FullTimeEmployee(id, employee\_name, bs, da, hr)  PRINT "Enter Part Time Employee details:"  PRINT "Enter employee ID: "  READ id  IGNORE\_NEWLINE  PRINT "Enter employee name: "  READ employee\_name  PRINT "Enter rate per hour: "  READ rate\_per\_hour  PRINT "Enter total hours worked in a day: "  READ total\_hours  CALL PartTimeEmployee(id, employee\_name, rate\_per\_hour, total\_hours)  PRINT "Full Time Employee details:"  PRINT "ID: " + CALL fte.getEmpId()  PRINT "Name: " + CALL fte.getName()  PRINT "Salary: " + CALL fte.calculateSalary()  PRINT "Part Time Employee details:"  PRINT "ID: " + CALL pte.getEmpId()  PRINT "Name: " + CALL pte.getName()  PRINT "Wages: " + CALL pte.calculateWages()  RETURN 0  ENDMAIN |
| **PROGRAM:** | #include <iostream>  #include <string.h>  using namespace std;  class Employee  {  protected: // Protected Data Members because we want it to be accessible in the derived classes  int emp\_id;  string name;  public:  Employee(int id, string nm) // Parametrized Constructor  {  emp\_id = id;  name = nm;  }  int getEmpId() // Getter Function to get Employee ID  {  return emp\_id;  }  string getName() // Getter Function to get Employee Name  {  return name;  }  };  class FullTimeEmployee : public Employee // Inheriting Employee Class in FullTimeEmployee Class publicly  {  private: // Private Data Members because we don't want it to be directly accessible outside of the class  double basic\_salary;  double dearness\_allowance;  double hra;  public:  FullTimeEmployee(int id, string nm, double bs, double da, double hr) : Employee(id, nm) //Constructor for subclass FullTimeEmployee of Employee  {  basic\_salary = bs;  dearness\_allowance = da;  hra = hr;  }  double calculateSalary()  {  return basic\_salary + dearness\_allowance + hra;  }  };  class PartTimeEmployee : public Employee  {  private:  double rate\_per\_hour;  int total\_hours;  public:  PartTimeEmployee(int id, string nm, double rph, int th) : Employee(id, nm) //Constructor for subclass PartTimeEmployee of Employee  {  rate\_per\_hour = rph;  total\_hours = th;  }  double calculateWages()  {  return rate\_per\_hour \* total\_hours \* 30;  }  };  int main()  {  // Taking input from user for Full Time Employee  int id, total\_hours;  double bs, da, hr, rate\_per\_hour;  string employee\_name;  cout << "Enter Full Time Employee details:" << endl;  cout << "Enter employee ID: ";  cin >> id;  cin.ignore(); // To ignore the newline character left in the buffer by the previous input statement  cout << "Enter employee name: ";  getline(cin, employee\_name); // To take input with spaces in between  cout << "Enter basic salary: ";  cin >> bs;  cout << "Enter dearness allowance: ";  cin >> da;  cout << "Enter house rent allowance: ";  cin >> hr;  FullTimeEmployee fte(id, employee\_name, bs, da, hr);  // Taking input from user for Part Time Employee  cout << "Enter Part Time Employee details:" << endl;  cout << "Enter employee ID: ";  cin >> id;  cin.ignore(); // To ignore the newline character left in the buffer by the previous input statement  cout << "Enter employee name: ";  getline(cin, employee\_name);  cout << "Enter rate per hour: ";  cin >> rate\_per\_hour;  cout << "Enter total hours worked in a day: ";  cin >> total\_hours;  PartTimeEmployee pte(id, employee\_name, rate\_per\_hour, total\_hours);  //Printing Full Time Employee details  cout << "Full Time Employee details:" << endl;  cout << "ID: " << fte.getEmpId() << endl;  cout << "Name: " << fte.getName() << endl;  cout << "Salary: " << fte.calculateSalary() << endl;  //Printing Part Time Employee details  cout << "Part Time Employee details:" << endl;  cout << "ID: " << pte.getEmpId() << endl;  cout << "Name: " << pte.getName() << endl;  cout << "Wages: " << pte.calculateWages() << endl;  return 0;  } |
| **RESULT:** |  |
| **Program 2** | |
| **PROBLEM STATEMENT:** | Mr John is given the contract to colour the house . Design a c++ program that implements multi- level inheritance in such a way that   * + There must be 3 classes i.e. Exterior ,Paint ,Bill.   + The Class Bill inherits the Paint class which in turn inherits the Class Exterior   + Class Exterior contains a function to find the area of the house   + Class Paint contains a function to select the paint type and brand   + Class Bill calculates the final cost of the painting by getting the house details and paint details.   Write a main() function to create appropriate objects and simulate the above scenario. |
| **ALGORITHM:** | CLASS Exterior:  PROTECTED area    PUBLIC  FUNCTION setArea(a)  area = a  FUNCTION getArea()  return area  CLASS Paint:  INHERIT Exterior  PROTECTED paintType, brand  PUBLIC  FUNCTION setPaintType(p)  paintType = p  FUNCTION setBrand(b)  brand = b  FUNCTION selectPaint()  print "Selecting paint type: " + paintType  print "Selecting brand: " + brand  CLASS Bill:  INHERIT Paint  PRIVATE costPerSqFt, totalCost  PUBLIC  FUNCTION setCostPerSqFt(c)  costPerSqFt = c  FUNCTION calculateCost()  totalCost = area \* costPerSqFt  print "The final cost for painting the house is $" + totalCost  FUNCTION main()  instantiate Bill as b  b.setArea(1000)  b.setPaintType("Luxury Emulsion")  b.setBrand("Nerolac")  b.setCostPerSqFt(3.50)  b.selectPaint()  b.calculateCost()  ENDMAIN |
| **PROGRAM:** | #include <iostream>  using namespace std;  class Exterior // Base Class  {  protected:  float area;  public:  void setArea(float a)  {  area = a;  }  float getArea()  {  return area;  }  };  class Paint : public Exterior //Inheriting Exterior Class in Paint Class publicly. Paint class sets the paint type and brand  {  protected:  string paintType;  string brand;  public:  void setPaintType(string p)  {  paintType = p;  }  void setBrand(string b)  {  brand = b;  }  void selectPaint()  {  cout << "Selecting paint type: " << paintType << endl;  cout << "Selecting brand: " << brand << endl;  }  };  class Bill : public Paint //Inheriting Paint Class in Bill Class publicly. Bill class calculates the cost of painting the house  {  private:  float costPerSqFt;  float totalCost;  public:  void setCostPerSqFt(float c)  {  costPerSqFt = c;  }  void calculateCost()  {  totalCost = area \* costPerSqFt;  cout << "The final cost for painting the house is $" << totalCost << endl;  }  };  int main()  {  Bill b;  b.setArea(1000);  b.setPaintType("Luxury Emulsion");  b.setBrand("Nerolac");  b.setCostPerSqFt(3.50);  b.selectPaint();  b.calculateCost();  return 0;  } |
| **RESULT:** |  |
| **Program 3** | |
| **PROBLEM STATEMENT:** | a. Design three classes: STUDENT, EXAM and RESULT. The STUDENT class has data members such as rollno, name. EXAM is created by inheriting STUDENT. EXAM class adds data members representing the marks scored in six subjects. Derive RESULT from EXAM and has its own data members such as totalmarks. Write a program to model this relationship. |
| **ALGORITHM:** | CLASS STUDENT  PROTECTED:  INTEGER rollno  STRING name  PUBLIC:  FUNCTION get\_data()  PRINT "Enter Roll No: "  READ rollno  PRINT "Enter Name: "  READ name  ENDFUNCTION  ENDCLASS  CLASS EXAM : PUBLIC STUDENT  PROTECTED:  INTEGER marks[6]  PUBLIC:  FUNCTION set\_marks()  PRINT "Enter Marks of 6 Subjects:"  FOR i IN RANGE 0 TO 5  PRINT "Subject " + i+1 + ": "  READ marks[i]  ENDFOR  ENDFUNCTION  ENDCLASS  CLASS RESULT : PUBLIC EXAM  PRIVATE:  INTEGER totalmarks  PUBLIC:  FUNCTION calculate\_total\_marks()  SET totalmarks TO 0  FOR i IN RANGE 0 TO 5  ADD marks[i] TO totalmarks  ENDFOR  ENDFUNCTION  FUNCTION display\_result()  PRINT "Roll No: " + rollno  PRINT "Name: " + name  FOR i IN RANGE 0 TO 5  PRINT "Subject " + i+1 + " Marks: " + marks[i]  ENDFOR  PRINT "Total Marks: " + totalmarks  ENDFUNCTION  ENDCLASS  FUNCTION main()  DECLARE AN OBJECT r OF CLASS RESULT  CALL r.get\_data()  CALL r.set\_marks()  CALL r.calculate\_total\_marks()  CALL r.display\_result()  RETURN 0  ENDMAIN |
| **PROGRAM:** | #include <iostream>  #include <string>  using namespace std;  class STUDENT // Base Class STUDENT  {  protected:  int rollno;  string name;  public:  void get\_data() // Function to get student data  {  cout << "Enter Roll No: ";  cin >> rollno;  cin.ignore();  cout << "Enter Name: ";  getline(cin, name);  }  };  class EXAM : public STUDENT // Derived Class EXAM publicly inherited from Base Class STUDENT  {  protected:  int marks[6];  public:  void set\_marks() // Function to set marks  {  cout << "Enter Marks of 6 Subjects: " << endl;  for (int i = 0; i < 6; i++)  {  cout << "Subject " << i + 1 << ": ";  cin >> marks[i];  }  }  };  class RESULT : public EXAM // Derived Class RESULT publicly inherited from Derived Class EXAM  {  private: // Private Data Members because we don't want it to be directly accessible outside of the class  int totalmarks;  public:  void calculate\_total\_marks() // Function to calculate total marks  {  totalmarks = 0;  for (int i = 0; i < 6; i++)  {  totalmarks += marks[i];  }  }  void display\_result() // Function to display result  {  cout << "Roll No: " << rollno << endl;  cout << "Name: " << name << endl;  for (int i = 0; i < 6; i++) {  cout << "Subject " << i + 1 << " Marks: " << marks[i] << endl;  }  cout << "Total Marks: " << totalmarks << endl;  }  };  int main()  {  RESULT r;  r.get\_data();  r.set\_marks();  r.calculate\_total\_marks();  r.display\_result();  return 0;  } |
| **RESULT:** |  |
| **PROBLEM STATEMENT** | b.Taking person as base class: Attributes: Name,gender,age Function : get\_data()  Worker derived from person: Attributes: base\_salary,shift(in hours) Function: Set\_data()  Manger derived from worker: Attributes: dept,no of people(subordinates or people working under him/her),bonus Function:display\_payment()  Display the total payment. Total payment calculated as : TP=(base\_salary\*shift)+(no o people\* bonus) |
| **ALGORITHM** | CLASS PERSON  PROTECTED:  STRING name  STRING gender  INTEGER age  PUBLIC:  FUNCTION get\_data()  PRINT "Enter Name: "  READ name  PRINT "Enter Gender: "  READ gender  PRINT "Enter Age: "  READ age  END FUNCTION  END CLASS  CLASS WORKER : PUBLIC PERSON  PROTECTED:  FLOAT base\_salary  INTEGER shift  PUBLIC:  FUNCTION set\_data()  PRINT "Enter Base Salary: "  READ base\_salary  PRINT "Enter Shift: "  READ shift  END FUNCTION  END CLASS  CLASS MANAGER : PUBLIC WORKER  PRIVATE:  STRING dept  INTEGER num\_people  FLOAT bonus  PUBLIC:  FUNCTION display\_payment()  PRINT "Enter Department: "  READ dept  PRINT "nter Number of People: "  READ num\_people  PRINT "Enter Bonus: "  READ bonus  SET total\_payment TO (base\_salary \* shift) + (num\_people \* bonus)  PRINT "Total Payment: Rs." + total\_payment  END FUNCTION  END CLASS  FUNCTION main()  DECLARE AN OBJECT m OF CLASS MANAGER  CALL m.get\_data()  CALL m.set\_data()  CALL m.display\_payment()  RETURN 0  END MAIN |
| **PROGRAM** | #include <iostream>  #include <string>  using namespace std;  class Person  {  protected:  string name;  string gender;  int age;  public:  void get\_data()  {  cout << "Enter Name: ";  getline(cin, name);  cout << "Enter Gender: ";  getline(cin, gender);  cout << "Enter Age: ";  cin >> age;  cin.ignore();  }  };  class Worker : public Person  {  protected:  float base\_salary;  int shift;  public:  void set\_data()  {  cout << "Enter Base Salary: ";  cin >> base\_salary;  cout << "Enter Shift (in hours): ";  cin >> shift;  cin.ignore();  }  };  class Manager : public Worker  {  private:  string dept;  int num\_people;  float bonus;  public:  void display\_payment()  {  cout << "Enter Department: ";  cin.ignore();  getline(cin, dept);  cout << "Enter Number of People: ";  cin >> num\_people;  cout << "Enter Bonus: ";  cin >> bonus;  float total\_payment = (base\_salary \* shift) + (num\_people \* bonus);  cout << "Total Payment: Rs." << total\_payment << endl;  }  };  int main() {  Manager m;  m.get\_data();  m.set\_data();  m.display\_payment();  return 0;  } |
| **RESULT** |  |
| **PROBLEM STATEMENT 3** | Consider that you want to buy a mobile. Design a program that implements multi-level inheritance with the following specifications.  1. There must be 3 classes i.e., Mobile, Model, Bill.  2. The class Bill inherits the class Model which in turn inherits the class Mobile.  3. Class Mobile contains a function that takes the brand of the mobile.  4. Class Model contains a function to select the specifications and model.  5. Class Bill calculates the final cost of the mobile by getting the brand details and model details.  Using the above specifications as a base, write a program which inputs the details of a mobile and prints its final bill based on the mobile brand and model |
| **ALGORITHM** | CLASS Mobile  PROTECTED:  STRING brand  PUBLIC:  FUNCTION setBrand(string b)  SET brand TO b  END FUNCTION  END CLASS  CLASS Model : PUBLIC Mobile  PROTECTED:  STRING model  INTEGER price  PUBLIC:  FUNCTION setModel(string m, int p)  SET model TO m  SET price TO p  END FUNCTION  FUNCTION getPrice()  RETURN price  END FUNCTION  END CLASS  CLASS Bill : PUBLIC Model  PUBLIC:  FUNCTION printBill()  PRINT "Brand: " + brand  PRINT "Model: " + model  PRINT "Price: " + getPrice()  END FUNCTION  END CLASS  FUNCTION main()  DECLARE AN OBJECT b OF CLASS Bill  DECLARE STRING brand, model  DECLARE INTEGER price  PRINT "Enter brand: "  READ brand  PRINT "Enter model: "  READ model  PRINT "Enter price: "  READ price  CALL b.setBrand(brand)  CALL b.setModel(model, price)  CALL b.printBill()  ENDMAIN |
| **PROGRAM:** | #include <iostream>  #include <string>  using namespace std;  class Mobile // Base class Mobile  {  protected:  string brand;  public:  void setBrand(string b)  {  brand = b;  }  };  class Model : public Mobile // Derived class Model publicly inherited from Base class Mobile  {  protected:  string model;  int price;  public:  void setModel(string m, int p)  {  model = m;  price = p;  }  int getPrice()  {  return price;  }  };  class Bill : public Model // Derived class Bill publicly inherited from Derived class Model  {  public:  void printBill()  {  cout << "Brand: " << brand << endl;  cout << "Model: " << model << endl;  cout << "Price: " << getPrice() << endl;  }  };  int main()  {  Bill b;  string brand, model;  int price;  cout << "Enter brand: ";  cin >> brand;  cout << "Enter model: ";  cin >> model;  cout << "Enter price: ";  cin >> price;  b.setBrand(brand);  b.setModel(model, price);  b.printBill();  return 0;  } |
| **RESULT:** |  |