

Assignment 04(Roll no : 107)

1.Title: Smart Inventory System with Dynamic Memory and Inheritance Problem Statement:

Design a program to manage an inventory system for a store. Each item in the store belongs to a specific category (like Electronics or Groceries), but the data must be stored and managed without using virtual functions. You must handle object relationships, memory allocation, and cleanup properly. Objectives: Implement: • Encapsulation (private/protected members)

Parameterized Constructors & Destructors Inheritance (Base → Derived classes) Dynamic allocation using pointers (new / delete) Pointer-to-object relationships (no virtual keyword)

Requirements:

1. Base Class: Item ○ Private members: ■ string name ■ int id ○ ○ ■ float price Protected member: ■ int quantity Public functions: ■ Parameterized constructor to initialize all members. ■ void display() — prints item details. ■ float getTotalValue() — returns price * quantity. ■ Destructor — prints when the item object is destroyed.
2. Derived Class 1: Electronics ○ ○ ○ ○ Additional data members: ■ int warrantyYears ■ float powerUsage Constructor should call base class constructor using initializer list. void displayDetails() — prints both base and derived details. Destructor prints a message for cleanup.
3. Derived Class 2: Grocery ○ Additional data members: ■ string expiryDate ■ float weight ○ ○ Constructor and destructor similar to Electronics. Function void displayDetails() to show all info.
4. Main Function Logic: ○ ○ ○ ○ ○ Ask user how many total items are in inventory. Dynamically create an array of pointers to Electronics and Grocery objects. For each item, ask the user for category and input details. Display all item details and total inventory value. Properly delete all dynamically allocated memory at the end.


```
97 int main() {
98     int totalItems;
99     cout << "Enter total number of items: ";
100     cin >> totalItems;
101
102     Item* items[100];
103     int itemCount = 0;
104     float totalInventoryValue = 0.0;
105
106     for (int i = 0; i < totalItems; i++) {
107         int choice;
108         cout << "Enter category of item " << (i + 1) << " : ";
109         cin >> choice;
110
111         if (choice == 1) {
112             string name;
113             int id, qty, warranty;
114             float price, power;
115             cout << "Enter Name, ID, Price, Quantity, WarrantyYears, PowerUsage: ";
116             cin >> name >> id >> price >> qty >> warranty >> power;
117
118             items[itemCount++] = new Electronics(name, id, price, qty, warranty, power);
119         }
120         else if (choice == 2) {
121             string name, expiry;
122             int id, qty;
123             float price, weight;
124             cout << "Enter Name, ID, Price, Quantity, ExpiryDate, Weight: ";
125             cin >> name >> id >> price >> qty >> expiry >> weight;
126
127             items[itemCount++] = new Grocery(name, id, price, qty, expiry, weight);
128         }
129         else {
130             cout << "Invalid choice. Skipping item." << endl;
131         }
132     }
133
134     cout << "===== INVENTORY DETAILS =====<pre></pre>"
```

```
Enter total number of items: 2
Enter category of item 1:
1. Electronics
2. Grocery
Choice: 1
Enter Name, ID, Price, Quantity, WarrantyYears, PowerUsage:
AC
101
47500
2
2
15

Enter category of item 2:
1. Electronics
2. Grocery
Choice: 2
Enter Name, ID, Price, Quantity, ExpiryDate, Weight:
Rice
102
105
10
12-2025
10

===== INVENTORY DETAILS =====

--- Electronics Item ---
Item Name: AC
Item ID: 101
Price: 47500
Quantity: 2
Warranty (years): 2
Power Usage (Watts): 15
Total Value: 95000

--- Electronics Item ---
Item Name: Rice
Item ID: 102
Price: 105
Quantity: 10
Warranty (years): 350183168
Power Usage (Watts): 6.02558e-43
Total Value: 1050

Total Inventory Value: 96050
Destroying Item: AC
Destroying Item: Rice

All items deleted successfully.

C:\Users\vipul\source\repos\Basic\Debug\Basic.exe (process 18584) exited with code 0 (0x0).
```

2.Title: Employee Payroll Management System (with Dynamic Bonus Calculation) Problem

Statement: Design a C++ program to manage employees of a company. Each employee has common details (name, ID, base salary), but different roles (e.g., Manager, Developer) that determine their bonus. You must use classes, inheritance, encapsulation, constructors, destructors, and pointers to:

- Store and display employee information.
- Dynamically allocate memory for employees.
- Compute their total salary (base + bonus).

Ensure proper cleanup of allocated memory. Requirements:

1. Base Class: Employee
 - Private Data Members: ■ string name ■ int id ◦ ■ float baseSalary
 - Protected Member: ■ float bonus
 - Public Functions: ■ Parameterized Constructor to initialize name, id, salary.
 - virtual void calculateBonus() → base version sets bonus = 0.
 - virtual void display() → prints employee details.
 - Virtual Destructor (for safe cleanup).
2. Derived Class: Manager (inherits from Employee)
 - Overrides calculateBonus() → bonus = 40% of baseSalary.
 - Overrides display() → shows “Manager” and total salary.
3. Derived Class: Developer (inherits from Employee)
 - Overrides calculateBonus() → bonus = 25% of baseSalary.
 - Overrides display() → shows “Developer” and total salary.
4. Main Function Logic:
5. Ask user how many employees to create. Dynamically create an array of Employee* pointers (using new). Let the user choose the type (Manager or Developer) for each. Use runtime polymorphism (Employee* e = new Manager(...)) to store objects. Call calculateBonus() and display() for each employee. Finally, delete all dynamically allocated objects safely.

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  class Employee {
6  private:
7      string name;
8      int id;
9      float baseSalary;
10
11 protected:
12     float bonus;
13     float getBaseSalary() const { return baseSalary; }
14
15 public:
16     Employee(string n, int i, float s) : name(n), id(i), baseSalary(s), bonus(0) {}
17
18     virtual void calculateBonus() { bonus = 0; }
19
20     virtual void display() {
21         cout << "Employee Name: " << name << endl;
22         cout << "Employee ID: " << id << endl;
23         cout << "Base Salary: " << baseSalary << endl;
24         cout << "Bonus: " << bonus << endl;
25         cout << "Total Salary: " << (baseSalary + bonus) << endl;
26     }
27
28     virtual ~Employee() {
29         cout << "Destroying Employee: " << name << endl;
30     }
31 };
32
33 class Manager : public Employee {
34 public:
35     Manager(string n, int i, float s) : Employee(n, i, s) {}
36
37     void calculateBonus() override {
38         bonus = 0.4 * getBaseSalary();
39     }
40
41     void display() override {
42         cout << "Manager Details: " << endl;
43     }
44 };

```

```
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search Basic
Process: [16920] Basic.exe Lifecycle Events Thread PB Stack Frame Application Insights GitHub Copilot

EmployeeSalaryCalculation.cpp InventoryManagement.cpp Library.cpp ComplexNumber.cpp Employee.cpp Bank.cpp Student.cpp Rectangle.cpp Basic.cpp
Basic
43
44
45 void display() override {
46     cout << "\n--- Manager Details ---" << endl;
47     Employee::display();
48 }
49
50 ~Manager() {
51     cout << "Cleaning up Manager...\n";
52 }
53
54
55 class Developer : public Employee {
56 public:
57     Developer(string n, int i, float s)
58         : Employee(n, i, s) {
59     }
60
61 void calculateBonus() override {
62     bonus = 0.25f * getBaseSalary();
63 }
64
65 void display() override {
66     cout << "\n--- Developer Details ---" << endl;
67     Employee::display();
68 }
69
70 ~Developer() {
71     cout << "Cleaning up Developer...\n";
72 }
73
74
75 int main() {
76     int totalEmployees;
77     cout << "Enter total number of employees: ";
78     cin >> totalEmployees;
79
80     Employee* emplist[100];
81     int empCount = 0;
82
83     for (int i = 0; i < totalEmployees; i++) {
84         int choice;
85         cout << "\nEnter type of employee * << (i + 1) << " << endl;
86         cout << "1. Manager\n2. Developer\nChoice: ";
87         cin >> choice;
88     }
89
90     string name;
91     int id;
92     float salary;
93     cout << "Enter Name, ID, Base Salary: ";
94     cin >> name >> id >> salary;
95
96     if (choice == 1)
97         emplist[empCount++] = new Manager(name, id, salary);
98     else if (choice == 2)
99         emplist[empCount++] = new Developer(name, id, salary);
100     else
101         cout << "Invalid choice! Skipping.\n";
102
103     cout << "\n===== EMPLOYEE DETAILS =====\n";
104     for (int i = 0; i < empCount; i++) {
105         emplist[i] -> calculateBonus();
106         emplist[i] -> display();
107     }
108
109     for (int i = 0; i < empCount; i++) {
110         delete emplist[i];
111     }
112
113     cout << "\nAll employees deleted successfully.\n";
114     return 0;
115 }
116
117
118
83% No issues found Lnt 42 Ch: 41 SPC CRLF
```

```
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search Basic
Process: [16920] Basic.exe Lifecycle Events Thread PB Stack Frame Application Insights GitHub Copilot

EmployeeSalaryCalculation.cpp InventoryManagement.cpp Library.cpp ComplexNumber.cpp Employee.cpp Bank.cpp Student.cpp Rectangle.cpp Basic.cpp
Basic
76
77 int totalEmployees;
78 cout << "Enter total number of employees: ";
79 cin >> totalEmployees;
80
81 Employee* emplist[100];
82 int empCount = 0;
83
84 for (int i = 0; i < totalEmployees; i++) {
85     int choice;
86     cout << "\nEnter type of employee * << (i + 1) << " << endl;
87     cout << "1. Manager\n2. Developer\nChoice: ";
88     cin >> choice;
89
90     string name;
91     int id;
92     float salary;
93     cout << "Enter Name, ID, Base Salary: ";
94     cin >> name >> id >> salary;
95
96     if (choice == 1)
97         emplist[empCount++] = new Manager(name, id, salary);
98     else if (choice == 2)
99         emplist[empCount++] = new Developer(name, id, salary);
100     else
101         cout << "Invalid choice! Skipping.\n";
102
103     cout << "\n===== EMPLOYEE DETAILS =====\n";
104     for (int i = 0; i < empCount; i++) {
105         emplist[i] -> calculateBonus();
106         emplist[i] -> display();
107     }
108
109     for (int i = 0; i < empCount; i++) {
110         delete emplist[i];
111     }
112
113     cout << "\nAll employees deleted successfully.\n";
114     return 0;
115 }
116
117
118
83% No issues found Lnt 42 Ch: 41 SPC CRLF
```

```
Microsoft Visual Studio Debu x + v
Enter total number of employees: 2
C:\Users\vipul\source\repos\Basic\Debug\Basic.exe (process 16920) exited with code -1 (0xffffffff).
Enter type of employee 1:
1. Manager
2. Developer
Choice: 1
Enter Name, ID, Base Salary: Vipul
111
65000
Enter type of employee 2:
1. Manager
2. Developer
Choice: 2
Enter Name, ID, Base Salary: Sagar
22
75000
===== EMPLOYEE DETAILS =====
--- Manager Details ---
Employee Name: Vipul
Employee ID: 111
Base Salary: 65000
Bonus: 26000
Total Salary: 91000
--- Developer Details ---
Employee Name: Sagar
Employee ID: 22
Base Salary: 75000
Bonus: 18750
Total Salary: 93750
Cleaning up Manager...
Destroying Employee: Vipul
Cleaning up Developer...
Destroying Employee: Sagar
All employees deleted successfully.
C:\Users\vipul\source\repos\Basic\Debug\Basic.exe (process 13900) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

3.Title: Menu-Driven Employee Management System usingClasses, Objects, Inheritance, and Dynamic Memory in C++ Problem Statement Design a Menu-Driven Employee Management System for a company that manages two types of employees:

1. FullTimeEmployee
2. PartTimeEmployee You must: ● Use inheritance to derive these two classes from a base class Employee. ● Use encapsulation for data hiding (private/protected members). ● Create objects dynamically using pointers. ● Display and manage data using a menudriven interface. Class Design Base Class: Employee Private Members: ● string name ● int empID Protected Member: ● float salary Public Functions: ● Parameterized constructor (for name and empID) ● void displayBasic() → shows name and ID ● float getSalary() → returns salary ● Destructor → prints destruction message Derived Class: FullTimeEmployee Additional Members: ● float basicPay, float bonus Constructor: ● Uses initializer list to call base constructor and initialize basicPay and bonus Member Function: ● void calculateSalary() → salary = basicPay + bonus ● void displayDetails() → display all employee info ● Destructor → prints cleanup message Derived Class: PartTimeEmployee Additional Members: ● int hoursWorked ● float hourlyRate Constructor: ● Calls base class constructor and initializes new members Member Function: ● void calculateSalary() → salary = hoursWorked * hourlyRate ● void displayDetails() ● Destructor → prints cleanup message Menu Options in main()

1.Add Full-Time Employee 2.Add Part-Time Employee 3.Display All Employees 4.Search Employee by ID 5.Delete Employee (by ID) 6.Exit Program.

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  class Employee {
6  private:
7      string name;
8      int empID;
9  protected:
10     float salary;
11  public:
12     Employee(string n, int id) : name(n), empID(id), salary(0) {}
13     void displayBasic() {
14         cout << "Name: " << name << ", ID: " << empID << endl;
15     }
16     float getSalary() { return salary; }
17     int getID() { return empID; }
18     virtual void displayDetails() = 0;
19     virtual ~Employee() { cout << "Destroying Employee: " << name << endl; }
20 };
21
22 class FullTimeEmployee : public Employee {
23 private:
24     float basicPay;
25     float bonus;
26 public:
27     FullTimeEmployee(string n, int id, float bPay, float b) :
28         Employee(n, id), basicPay(bPay), bonus(b) {}
29
30     void calculateSalary() { salary = basicPay + bonus; }
31     void displayDetails() override {
32         cout << "Full-Time Employee ----> << endl;
33         displayBasic();
34         cout << "Basic Pay: " << basicPay << ", Bonus: " << bonus << endl;
35         cout << "Total Salary: " << salary << endl;
36     }
37     ~FullTimeEmployee() { cout << "Cleaning up Full-Time Employee...\n"; }
38 };
39
40 class PartTimeEmployee : public Employee {
41 private:
42     int hoursWorked;
43     float hourlyRate;
44 public:
45     PartTimeEmployee(string n, int id, int hWorked, float hRate) :
46         Employee(n, id), hoursWorked(hWorked), hourlyRate(hRate) {}
47
48     void calculateSalary() { salary = hoursWorked * hourlyRate; }
49     void displayDetails() override {
50         cout << "Part-Time Employee ----> << endl;
51     }
52     ~PartTimeEmployee() { cout << "Cleaning up Part-Time Employee...\n"; }
53 };
54
55 int main() {
56     Employee* empList[100];
57     int empCount = 0;
58     int choice;
59
60     do {
61         cout << "\nMenu:\n1.Add Full-Time Employee\n2.Add Part-Time Employee\n3.Display All Employees\n4.Search Employee by ID\n5.Delete Employee by ID\n6.Exit\nChoice: ";
62         cin >> choice;
63
64         if (choice == 1) {
65             string name;
66             int id;
67             float basic, bonus;
68             cout << "Enter Name, ID, Basic Pay, Bonus: ";
69             cin >> name >> id >> basic >> bonus;
70             FullTimeEmployee fte = new FullTimeEmployee(name, id, basic, bonus);
71             fte->calculateSalary();
72             empList[empCount++] = fte;
73         }
74         else if (choice == 2) {
75             string name;
76             int id, hours;
77             float rate;
78             cout << "Enter Name, ID, Hours Worked, Hourly Rate: ";
79             cin >> name >> id >> hours >> rate;
80             PartTimeEmployee pte = new PartTimeEmployee(name, id, hours, rate);
81             pte->calculateSalary();
82             empList[empCount++] = pte;
83         }
84         else if (choice == 3) {
85             cout << "\n--- All Employees ----> << endl;
86             for (int i = 0; i < empCount; i++)
87                 empList[i]->displayDetails();
88         }
89         else if (choice == 4) {
90             int searchID;
91             cout << "Enter Employee ID to search: ";
92             cin >> searchID;
93             bool found = false;
94             for (int i = 0; i < empCount; i++) {
95                 if (empList[i]->getID() == searchID) {

```

```

95                 if (empList[i]->getID() == searchID) {

```


