**1. Shape, Rectangle, and Triangle Classes**

#include <iostream>

using namespace std;

class Shape {

protected:

double height, width;

public:

Shape(double h = 0, double w = 0) : height(h), width(w) {}

virtual double area() = 0;

virtual double perimeter() = 0;

};

class Rectangle : public Shape {

public:

Rectangle(double h, double w) : Shape(h, w) {}

double area() override {

return height \* width;

}

double perimeter() override {

return 2 \* (height + width);

}

};

class Triangle : public Shape {

public:

Triangle(double h, double w) : Shape(h, w) {}

double area() override {

return 0.5 \* height \* width;

}

double perimeter() override {

return height + width + sqrt(height \* height + width \* width);

}

};

int main() {

Rectangle rect(5, 10);

Triangle tri(5, 10);

cout << "Rectangle Area: " << rect.area() << endl;

cout << "Rectangle Perimeter: " << rect.perimeter() << endl;

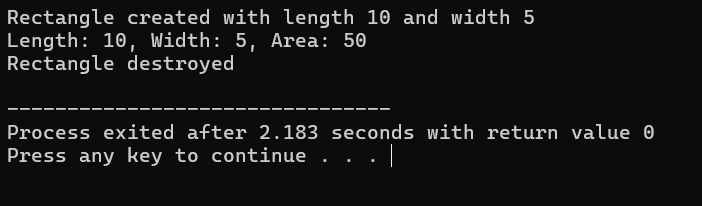
cout << "Triangle Area: " << tri.area() << endl;

cout << "Triangle Perimeter: " << tri.perimeter() << endl;

return 0;

}

**Output**

****

**2. Vehicle, Car, and Truck Classes**

#include <iostream>

using namespace std;

class Vehicle {

protected:

string make, model;

int year;

public:

Vehicle(string m, string mod, int y) : make(m), model(mod), year(y) {}

virtual void display() = 0; // Pure virtual function

};

class Car : public Vehicle {

private:

int seating\_capacity;

string fuel\_type;

public:

Car(string m, string mod, int y, int sc, string ft) : Vehicle(m, mod, y), seating\_capacity(sc), fuel\_type(ft) {}

void display() override {

cout << "Car - Make: " << make << ", Model: " << model << ", Year: " << year

<< ", Seating Capacity: " << seating\_capacity << ", Fuel Type: " << fuel\_type << endl;

}

};

class Truck : public Vehicle {

private:

double payload\_capacity, towing\_capacity;

public:

Truck(string m, string mod, int y, double pc, double tc) : Vehicle(m, mod, y), payload\_capacity(pc), towing\_capacity(tc) {}

void display() override {

cout << "Truck - Make: " << make << ", Model: " << model << ", Year: " << year

<< ", Payload Capacity: " << payload\_capacity << " tons, Towing Capacity: " << towing\_capacity << " tons" << endl;

}

};

int main() {

Car car("Toyota", "Camry", 2020, 5, "Gasoline");

Truck truck("Ford", "F-150", 2019, 1.5, 3.5);

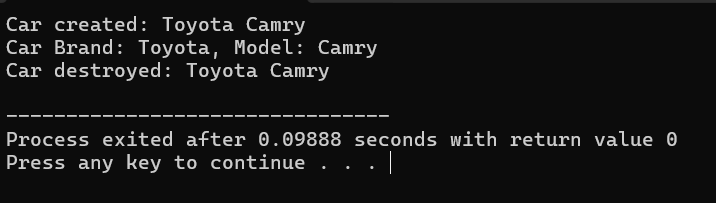
car.display();

truck.display();

return 0;

}

**Output**

****

**3. Animal, Cat, and Dog Classes**

#include <iostream>

using namespace std;

class Animal {

protected:

string name, species;

int age;

public:

Animal(string n, string s, int a) : name(n), species(s), age(a) {}

virtual void display() = 0; // Pure virtual function

};

class Cat : public Animal {

private:

string color, breed;

public:

Cat(string n, string s, int a, string c, string b) : Animal(n, s, a), color(c), breed(b) {}

void display() override {

cout << "Cat - Name: " << name << ", Species: " << species << ", Age: " << age

<< ", Color: " << color << ", Breed: " << breed << endl;

}

};

class Dog : public Animal {

private:

double weight;

string breed;

public:

Dog(string n, string s, int a, double w, string b) : Animal(n, s, a), weight(w), breed(b) {}

void display() override {

cout << "Dog - Name: " << name << ", Species: " << species << ", Age: " << age

<< ", Weight: " << weight << " kg, Breed: " << breed << endl;

}

};

int main() {

Cat cat("Whiskers", "Felis", 3, "Black", "Siamese");

Dog dog("Buddy", "Canis", 5, 20.5, "Labrador");

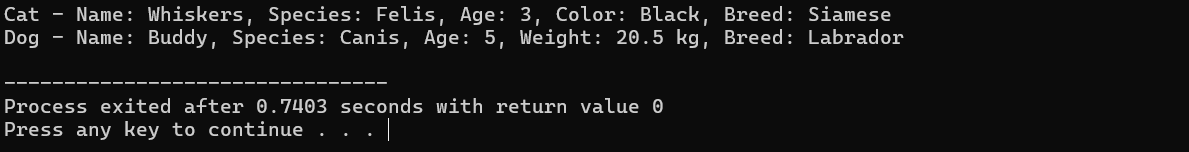
cat.display();

dog.display();

return 0;

}

**Output**

****

**4. Employee, Manager, and Engineer Classes**

#include <iostream>

using namespace std;

class Employee {

protected:

string name;

int id;

double salary;

public:

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

virtual void display() = 0; // Pure virtual function

};

class Manager : public Employee {

private:

string department;

double bonus;

public:

Manager(string n, int i, double s, string d, double b) : Employee(n, i, s), department(d), bonus(b) {}

void display() override {

cout << "Manager - Name: " << name << ", ID: " << id << ", Salary: $" << salary

<< ", Department: " << department << ", Bonus: $" << bonus << endl;

}

};

class Engineer : public Employee {

private:

string specialty;

int hours;

public:

Engineer(string n, int i, double s, string sp, int h) : Employee(n, i, s), specialty(sp), hours(h) {}

void display() override {

cout << "Engineer - Name: " << name << ", ID: " << id << ", Salary: $" << salary

<< ", Specialty: " << specialty << ", Hours: " << hours << " hours/week" << endl;

}

};

int main() {

Manager mgr("Alice", 101, 80000, "HR", 5000);

Engineer eng("Bob", 102, 75000, "Software", 40);

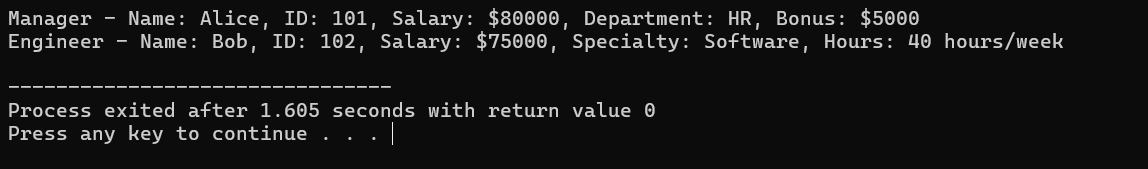
mgr.display();

eng.display();

return 0;

}

**Output**

****

**5. Person, Student, and Teacher Classes**

#include <iostream>

using namespace std;

class Person {

protected:

string name;

int age;

string gender;

public:

Person(string n, int a, string g) : name(n), age(a), gender(g) {}

virtual void display() = 0; // Pure virtual function

};

class Student : public Person {

private:

int roll\_number;

string student\_class;

public:

Student(string n, int a, string g, int rn, string sc) : Person(n, a, g), roll\_number(rn), student\_class(sc) {}

void display() override {

cout << "Student - Name: " << name << ", Age: " << age << ", Gender: " << gender

<< ", Roll Number: " << roll\_number << ", Class: " << student\_class << endl;

}

};

class Teacher : public Person {

private:

string subject;

double salary;

public:

Teacher(string n, int a, string g, string s, double sal) : Person(n, a, g), subject(s), salary(sal) {}

void display() override {

cout << "Teacher - Name: " << name << ", Age: " << age << ", Gender: " << gender

<< ", Subject: " << subject << ", Salary: $" << salary << endl;

}

};

int main() {

Student student("John", 16, "Male", 101, "10th Grade");

Teacher teacher("Mrs. Smith", 45, "Female", "Mathematics", 55000);

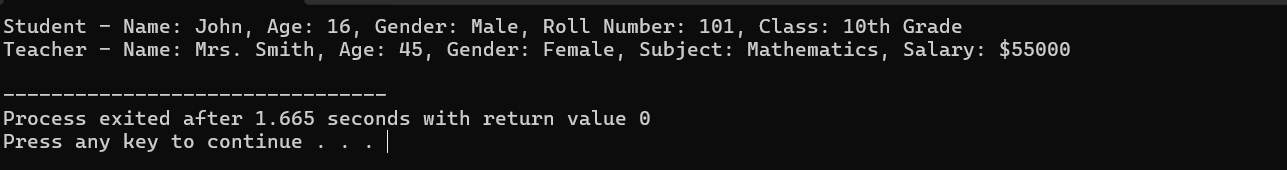
student.display();

teacher.display();

return 0;

}

**Output**

****