# LAB SESSION 7

**Question 1: Basic Inheritance**

**Create a base class Vehicle with a method move() that prints "Vehicle is moving". Derive a class Car from Vehicle and override the move() method to print "Car is moving".**

**Code:**

```cpp
#include <iostream>
using namespace std;

int startlab7()
{
    cout << "Name: Saad Ali Khan(SE-23083)" << endl;
    cout << "Start of Lab 07" << endl;
    return 0;
}

class Vehicle
{
public:
    virtual void move() const
    {
        cout << "Vehicle is moving" << endl;
    }
};

class Car : public Vehicle
{
public:
    void move() const override
    {
        cout << "Car is moving" << endl;
    }
};

int l7q1()
{
    Vehicle v;
    Car c;

    v.move();
    c.move();
```

```
    return 0;
}

int main()
{
    startlab7();
    l7q1();
    return 0;
}
```

**Output:**

```
Name: Saad Ali Khan(SE-23083)
Start of Lab 07
Vehicle is moving
Car is moving
PS D:\SE\oops_labs>
```

**Question 2: Constructor in Derived Class**

**Create a base class Person with attributes name and age. Create a derived class Student with an additional attribute studentID. Initialize these attributes using constructors.**

**Code:**

```
#include <iostream>
using namespace std;

int startlab7()
{
    cout << "Name: Saad Ali Khan(SE-23083)" << endl;
    cout << "Lab 07" << endl;
    return 0;
}
class Person
{
protected:
    string name;
    int age;

public:
    Person(const string &n, int a) : name(n), age(a) {}
};
```

```cpp
class Student : public Person
{
private:
    int studentID;

public:
    Student(const string &n, int a, int id) : Person(n, a), studentID(id) {}

    void display() const
    {
        cout << "Name: " << name << ", Age: " << age << ", Student ID: " <<
studentID << endl;
    }
};

int l7q2()
{
    Student s("Alice", 20, 12345);
    s.display();
    return 0;
}

int main()
{
    startlab7();
    l7q2();
    return 0;
}
```

**Output:**

```
Name: Saad Ali Khan(SE-23083)
Lab 07
Name: Alice, Age: 20, Student ID: 12345
PS D:\SE\oops_labs>
```

**Question 3: Method Overriding**

**Create a base class Shape with a method draw() that prints "Drawing Shape". Create derived classes Circle and Square that override the draw() method to print "Drawing Circle" and "Drawing Square", respectively.**

**Code:**

```cpp
#include <iostream>
using namespace std;

int startlab7()
{
    cout << "Name: Saad Ali Khan(SE-23083)" << endl;
    cout << "Lab 07" << endl;
    return 0;
}

class Shape
{
public:
    virtual void draw() const
    {
        cout << "Drawing Shape" << endl;
    }
};

class Circle : public Shape
{
public:
    void draw() const override
    {
        cout << "Drawing Circle" << endl;
    }
};

class Square : public Shape
{
public:
    void draw() const override
    {
        cout << "Drawing Square" << endl;
    }
};
```

```cpp
int l7q3()
{
    Shape s;
    Circle c;
    Square sq;

    s.draw();
    c.draw();
    sq.draw();

    return 0;
}

int main()
{
    startlab7();
    l7q3();
    return 0;
}
```

**Output:**

```
Name: Saad Ali Khan(SE-23083)
Lab 07
Drawing Shape
Drawing Circle
Drawing Square
PS D:\SE\oops_labs>
```

**Question 4: Access Specifiers**

**Create a base class Base with a private attribute privateVar, a protected attribute protectedVar, and a public attribute publicVar. Create a derived class Derived and demonstrate access to these attributes. Write your observations.**

**Code:**

```
#include <iostream>
using namespace std;

int startlab7()
{
    cout << "Name: Saad Ali Khan(SE-23083)" << endl;
    cout << "Lab 07" << endl;
    return 0;
}
class Base
{
private:
    int privateVar;

protected:
    int protectedVar;

public:
    int publicVar;

    Base() : privateVar(1), protectedVar(2), publicVar(3) {}
};

class Derived : public Base
{
public:
    void display()
    {
        // cout << "Private Var: " << privateVar << endl; // Not accessible
        cout << "Protected Var: " << protectedVar << endl; // Accessible
        cout << "Public Var: " << publicVar << endl;       // Accessible
    }
};

int l7q4()
{
    Derived d;
```

```
    d.display();

    // cout << d.privateVar << endl; // Not accessible
    // cout << d.protectedVar << endl; // Not accessible
    cout << "Public Var from main: " << d.publicVar << endl; // Accessible
    return 0;
}

int main()
{
    startlab7();
    l7q4();
    return 0;
}
```

**Observations:**

- `privateVar` is not accessible outside the `Base` class.
- `protectedVar` is accessible within the `Derived` class but not outside of it.
- `publicVar` is accessible both within the `Derived` class and outside of it.

## Question 5

**You are required to create a C++ program that demonstrates the concept of inheritance.Consider the following requirements:**

### Base Class - Shape:

**Attributes**:

• color (string)

• Constructor that initializes color.

• A pure virtual function area() which will return the area of the shape.

• A virtual function display() that prints the color of the shape.

### Derived Classes:

### Rectangle:

• Attributes:width (double),height (double)

• Constructor that initializes color, width, and height.

• Override area() to calculate and return the area of the rectangle.

• Override display() to print the color, width, height, and area of the rectangle.

### Circle:

### Attributes:

• radius (double)

• Constructor that initializes color and radius.

• Override area() to calculate and return the area of the circle.

• Override display() to print the color, radius, and area of the circle.

### Main Function:

• Create instances of Rectangle and Circle with different attributes.

• Use a pointer to Shape to store the addresses of these instances and call their display() method

### Code:

```cpp
#include <iostream>
#include <vector>
#include <memory>
using namespace std;

int startlab7()
{
    cout << "Name: Saad Ali Khan(SE-23083)" << endl;
    cout << "Lab 07" << endl;
    return 0;
}
class Shape
{
protected:
    string color;

public:
    Shape(const string &c) : color(c) {}

    virtual double area() const = 0;
```

```
    virtual void display() const
    {
        cout << "Color: " << color << endl;
    }

    virtual ~Shape() = default;
};

class Rectangle : public Shape
{
private:
    double width;
    double height;

public:
    Rectangle(const string &c, double w, double h)
        : Shape(c), width(w), height(h) {}

    double area() const override
    {
        return width * height;
    }

    void display() const override
    {
        Shape::display();
        cout << "Width: " << width << ", Height: " << height << ", Area: " <<
area() << endl;
    }
};

class Circle : public Shape
{
private:
    double radius;

public:
    Circle(const string &c, double r)
        : Shape(c), radius(r) {}

    double area() const override
    {
        return 3.14159 * radius * radius;
    }
```

```
    void display() const override
    {
        Shape::display();
        cout << "Radius: " << radius << ", Area: " << area() << endl;
    }
};

int l7q5()
{
    vector<shared_ptr<Shape>> shapes;

    shapes.push_back(make_shared<Rectangle>("Red", 5.0, 3.0));
    shapes.push_back(make_shared<Circle>("Blue", 7.0));

    for (const auto &shape : shapes)
    {
        shape->display();
    }

    return 0;
}

int main()
{
    startlab7();
    l7q5();
    return 0;
}
```

**Output:**

```
Name: Saad Ali Khan(SE-23083)
Lab 07
Color: Red
Width: 5, Height: 3, Area: 15
Color: Blue
Radius: 7, Area: 153.938
PS D:\SE\oops_labs>
```