**LAB SESSION 6**

**1. Question: Create a class Rectangle with attributes length and width. Implement methods to calculate the area and perimeter of the rectangle.**

**Code:**

#include <iostream>

using namespace std;

int startlab6()

{

    cout << "Name: Saad Ali Khan(SE-23083)" << endl;

    cout << "Start of Lab 06" << endl;

    return 0;

}

class Rectangle

{

private:

    double length;

    double width;

public:

    Rectangle(double l, double w) : length(l), width(w) {}

    double area() const

    {

        return length \* width;

    }

    double perimeter() const

    {

        return 2 \* (length + width);

    }

    void display() const

    {

        cout << "Rectangle [Length: " << length << ", Width: " << width << "]" << endl;

        cout << "Area: " << area() << endl;

        cout << "Perimeter: " << perimeter() << endl;

    }

};

int l6q1()

{

    Rectangle rect(5.0, 3.0);

    rect.display();

    return 0;

}

int main()

{

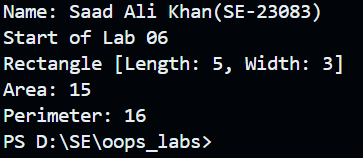
    startlab6();

    l6q1();

    return 0;

}

**Output:**

****

**2. Question: Create a class Circle with attribute radius. Implement methods to calculate the area and circumference of the circle.**

**Code:**

#include <iostream>

using namespace std;

int startlab6()

{

    cout << "Name: Saad Ali Khan(SE-23083)" << endl;

    cout << "Lab 06" << endl;

    return 0;

}

class Circle

{

private:

    double radius;

public:

    Circle(double r) : radius(r) {}

    double area() const

    {

        return 3.14 \* radius \* radius;

    }

    double circumference() const

    {

        return 2 \* 3.14 \* radius;

    }

    void display() const

    {

        cout << "Circle [Radius: " << radius << "]" << endl;

        cout << "Area: " << area() << endl;

        cout << "Circumference: " << circumference() << endl;

    }

};

int l6q2()

{

    Circle circ(5.0);

    circ.display();

    return 0;

}

int main()

{

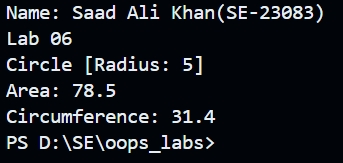
    startlab6();

    l6q2();

    return 0;

}

**Output:**

****

**3. Question: Create a class Employee with attributes name and salary. Implement a method to display the details of the employee.**

**Code:**

#include <iostream>

using namespace std;

int startlab6()

{

    cout << "Name: Saad Ali Khan(SE-23083)" << endl;

    cout << "Lab 06" << endl;

    return 0;

}

class Employee

{

private:

    string name;

    double salary;

public:

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

    void display() const

    {

        cout << "Employee [Name: " << name << ", Salary: " << salary << "]" << endl;

    }

};

int l6q3()

{

    Employee emp("John Doe", 50000.0);

    emp.display();

    return 0;

}

int main()

{

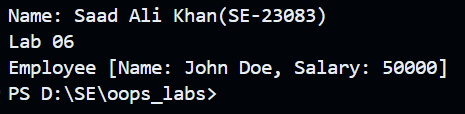
    startlab6();

    l6q3();

    return 0;

}

**Output:**

****

**4. Question: Create a class BankAccount with attributes accountNumber, accountHolder, and balance. Implement methods to deposit and withdraw money from the account.**

**Code:**

#include <iostream>

using namespace std;

int startlab6()

{

    cout << "Name: Saad Ali Khan(SE-23083)" << endl;

    cout << "Lab 06" << endl;

    return 0;

}

class BankAccount

{

private:

    int accountNumber;

    string accountHolder;

    double balance;

public:

    BankAccount(int accNum, string accHolder, double bal)

        : accountNumber(accNum), accountHolder(accHolder), balance(bal) {}

    void deposit(double amount)

    {

        if (amount > 0)

        {

            balance += amount;

        }

    }

    void withdraw(double amount)

    {

        if (amount > 0 && amount <= balance)

        {

            balance -= amount;

        }

        else

        {

            cout << "Insufficient funds." << endl;

        }

    }

    void display() const

    {

        cout << "BankAccount [Account Number: " << accountNumber

             << ", Account Holder: " << accountHolder

             << ", Balance: " << balance << "]" << endl;

    }

};

int l6q4()

{

    BankAccount account(123456, "Saad Ali Khan", 1000.0);

    account.deposit(500.0);

    account.withdraw(200.0);

    account.display();

    return 0;

}

int main()

{

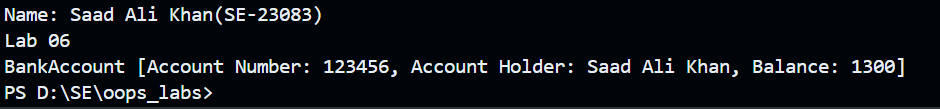
    startlab6();

    l6q4();

    return 0;

}

**Output:**

****

**5. Question: Create a class Car with attributes brand, model, and year. Implement a method to display the details of the car.**

**Code:**

#include <iostream>

using namespace std;

int startlab6()

{

    cout << "Name: Saad Ali Khan(SE-23083)" << endl;

    cout << "Lab 06" << endl;

    return 0;

}

class Car

{

private:

    string brand;

    string model;

    int year;

public:

    Car(string b, string m, int y) : brand(b), model(m), year(y) {}

    void display() const

    {

        cout << "Car [Brand: " << brand << ", Model: " << model << ", Year: " << year << "]" << endl;

    }

};

int l6q5()

{

    Car car("Toyota", "Corolla", 2020);

    car.display();

    return 0;

}

int main()

{

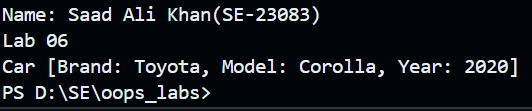
    startlab6();

    l6q5();

    return 0;

}

**Output:**

****

**6. Question: Create a class Fraction with attributes numerator and denominator. Implement a method to simplify the fraction.**

**Code:**

#include <iostream>

using namespace std;

int startlab6()

{

    cout << "Name: Saad Ali Khan(SE-23083)" << endl;

    cout << "Lab 06" << endl;

    return 0;

}

class Fraction

{

private:

    int numerator;

    int denominator;

    int gcd(int a, int b) const

    {

        while (b != 0)

        {

            int temp = b;

            b = a % b;

            a = temp;

        }

        return a;

    }

public:

    Fraction(int num, int den) : numerator(num), denominator(den)

    {

        if (den == 0)

        {

            throw invalid\_argument("Denominator cannot be zero.");

        }

    }

    void simplify()

    {

        int gcdValue = gcd(numerator, denominator);

        numerator /= gcdValue;

        denominator /= gcdValue;

    }

    void display() const

    {

        cout << "Fraction: " << numerator << "/" << denominator << endl;

    }

};

int l6q6()

{

    try

    {

        Fraction frac(10, 20);

        frac.simplify();

        frac.display();

    }

    catch (const invalid\_argument &e)

    {

        cerr << e.what() << endl;

    }

    return 0;

}

int main()

{

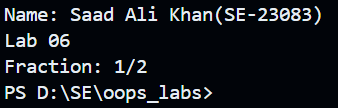
    startlab6();

    l6q6();

    return 0;

}

**Output:**

****