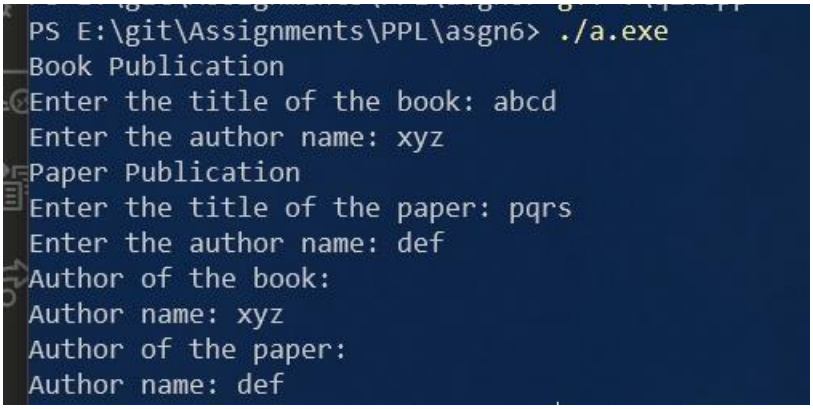
U18CO018
Shubham Shekhaliya
PPL
Lab Assignment 6

1-> Declare a class called author having author_name as private data member. Extend author class to have two sub classes called book_publication & paper_publication. Each of these classes have private member called title. Show usage of dynamic method dispatch (dynamic polymorphism) to display book or paper publications of a given author. Use command line arguments for inputting data.

```
#include <bits/stdc++.h>
using namespace std;
class author
{
private:
    string author_name;

public:
    void get_author_name()
    {
        cout << "Enter the author name: ";
        cin >> author_name;
    }
    void display_author_name()
    {
        cout << "Author name: " << author_name << endl;
    }
};
class book_publication : public author
{
private:
    string title;

public:
    void get_title()
    {
        cout << "Enter the title of the book: ";
        cin >> title;
    }
    void display_title()
    {
        cout << "Title of the book: " << title << endl;
    }
};
class paper_publication : public author
{
```

```
private:
    string title;

public:
    void get_title()
    {
        cout << "Enter the title of the paper: ";
        cin >> title;
    }
    void display_title()
    {
        cout << "Title of the paper: " << title << endl;
    }
};
int main()
{
    book_publication b;
    paper_publication p;
    cout << "Book Publication" << endl;
    b.get_title();
    b.get_author_name();
    cout << "Paper Publication" << endl;
    p.get_title();
    p.get_author_name();
    author *a;
    cout << "\nAuthor of the book: " << endl;
    a = &b;
    a->display_author_name();
    cout << "\nAuthor of the paper: " << endl;
    a = &p;
    a->display_author_name();
}
```

OUTPUT



2-> Write a class named Rectangle to represent a rectangle. It contains the following members:
Data: width (double) and height (double) that specify the width and height of the rectangle.
Methods:

1. A no-arg constructor that creates a default rectangle.
2. A constructor that creates a rectangle with the specified width and height.
3. A method named getArea() that returns the area of this rectangle.
4. A method named getPerimeter() that returns the perimeter

```cpp
#include <bits/stdc++.h>
using namespace std;
class Rectangle
{
private:
    double width;
    double height;

public:
    Rectangle()
    {
        width = 1;
        height = 1;
    }
    Rectangle(double w, double h)
    {
        width = w;
        height = h;
    }
    double getArea()
    {
        return width * height;
    }
    double getPerimeter()
    {
        return 2 * (width + height);
    }
};
int main()
{
    Rectangle r1;
    Rectangle r2(2.2, 3.3);
    cout << "Area of r1: " << r1.getArea() << endl;
    cout << "Perimeter of r1: " << r1.getPerimeter() << endl;
    cout << "Area of r2: " << r2.getArea() << endl;
    cout << "Perimeter of r2: " << r2.getPerimeter() << endl;
    return 0;
}
```

OUTPUT

```
PS E:\git\Assignments\PPL\asgn6> g++ .\q2.cpp
PS E:\git\Assignments\PPL\asgn6> ./a.exe
Area of r1: 1
Perimeter of r1: 4
Area of r2: 7.26
Perimeter of r2: 11
```

3-> It is required to compute SPI (semester performance index) of n students of a class for their registered subjects in a semester.
Assume that all students register for 6 subjects and each subject carry 5 credits. Also, follow SVNIT convention and method for computation of SPI. Declare a class called student having following data members: id_no, grades_obtained and SPI. Define constructor, display and calculate_spi methods. Define main to process data of n students.

```cpp
#include <bits/stdc++.h>
using namespace std;
class student
{
public:
    int id_no;
    int grades_obtained[6];
    double SPI;
    void display()
    {
        cout << "\nID No. : " << id_no;
        cout << "\nGrades obtained : ";
        for (int i = 0; i < 6; i++)
        {
            cout << grades_obtained[i] << " ";
        }
        cout << "\nSPI : " << SPI;
    }
    void calculate_spi()
    {
        int sum = 0;
        for (int i = 0; i < 6; i++)
        {
            sum = sum + grades_obtained[i];
        }
        SPI = (double)sum / 30;
    }
};
int main()
{
    int n;
    cout << "Enter the number of students : ";
    cin >> n;
```

```
    student s[n];
    for (int i = 0; i < n; i++)
    {
        cout << "\nEnter the details of student " << i + 1 << " : ";
        cout << "\nID No. : ";
        cin >> s[i].id_no;
        cout << "Grades obtained : ";
        for (int j = 0; j < 6; j++)
        {
            cin >> s[i].grades_obtained[j];
        }
        s[i].calculate_spi();
    }
    cout << "\n\nDetails of students : ";
    for (int i = 0; i < n; i++)
    {
        s[i].display();
    }
    return 0;
}
```

OUTPUT

```
PS E:\git\Assignments\PPL\asgn6> ./a.exe
Enter the number of students : 3

Enter the details of student 1 :
ID No. : 1
Grades obtained : 3 3 3 4 4 4

Enter the details of student 2 :
ID No. : 2
Grades obtained : 3 3 4 4 5 5

Enter the details of student 3 :
ID No. : 3
Grades obtained : 4 4 4 5 5 5


Details of students :
ID No. : 1
Grades obtained : 3 3 3 4 4 4
SPI : 7
ID No. : 2
Grades obtained : 3 3 4 4 5 5
SPI : 8
ID No. : 3
Grades obtained : 4 4 4 5 5 5
SPI : 9
```

4-> It is required to maintain and process the status of total 9 resources. The status value is to be stored in an integer array of dimensions 3x3. The valid status of a resource can be one of the following:
free: indicated by integer value 0
occupied: indicated by integer value 1
inaccessible: indicated by integer value 2
Declare a class called ResourcesStatus, having data member called statusRef, referring to a two dimensional array (3x3) of integers to be used to refer to the above mentioned status values. Define a member method called processStausCount that counts and displays total number of free resources, total number of occupied resources and total number of inaccessible resources. The exception to be raised and handled if total number of occupied resources exceeds total number of free resources. The handler marks status of all inaccessible resources as free. Accept initial status values from command line arguments and initialize the array. Raise and handle user defined exception if invalid status value given.

```
#include <bits/stdc++.h>
using namespace std;
class ResourcesStatus
{
public:
    int statusRef[3][3];
    ResourcesStatus()
    {
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                statusRef[i][j] = 0;
            }
        }
    }
    ResourcesStatus(int a[3][3])
    {
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                statusRef[i][j] = a[i][j];
            }
        }
    }
    vector<int> processStatusCount()
    {
        int free = 0, occupied = 0, inaccessible = 0;
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
```

```
            {
                if (statusRef[i][j] == 0)
                {
                    free++;
                }
                else if (statusRef[i][j] == 1)
                {
                    occupied++;
                }
                else if (statusRef[i][j] == 2)
                {
                    inaccessible++;
                }
            }
        }
        cout << "Total free resources are : " << free << endl;
        cout << "Total occupied resources are : " << occupied << endl;
        cout << "Total inaccessible resources are : " << inaccessible << endl;
        return {free, occupied, inaccessible};
    }
    void displayStatus()
    {
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                cout << statusRef[i][j] << " ";
            }
            cout << endl;
        }
    }
    void setStatus(int i, int j, int status)
    {
        statusRef[i][j] = status;
        vector<int> v = processStatusCount();
        if (v[1] > v[0])
        {
            string msg = "Invalid status value as occupied resources are more
than free resources ";
                for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    if (statusRef[i][j] == 2)
                    {
                        statusRef[i][j] = 0;
                    }
                }
            }
```

```
            throw msg;
        }
    }
};
int main()
{
    int a[3][3];
    cout << "Enter the status of resources" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++)
        {
            cin >> a[i][j];
        }
    }
    ResourcesStatus rs(a);
    while (1)
    {
        cout << "1. Display Status" << endl;
        cout << "2. Set Status" << endl;
        cout << "3. Exit" << endl;
        int choice;
        cin >> choice;
        switch (choice)
        {
        case 1:
            rs.displayStatus();
            break;
        case 2:
            int i, j, status;
            cout << "Enter row and column of the resource" << endl;
            cin >> i >> j;
            cout << "Enter status of the resource" << endl;
            cin >> status;
            try
            {
                rs.setStatus(i, j, status);
            }
            catch (string msg)
            {
                cout << msg << endl;
            }
            break;
        case 3:
            return 0;
        default:
            cout << "Invalid choice" << endl;
        }
    }
}
```

OUTPUT

```
PS E:\git\Assignments\PPL\asgn6> ./a.exe
Enter the status of resources
1 1 0 0 0 1 2 2 2
1. Display Status
2. Set Status
3. Exit
1
1 1 0
0 0 1
2 2 2
1. Display Status
2. Set Status
3. Exit
2
Enter row and column of the resource
1 1
Enter status of the resource
1
Total free resources are : 2
Total occupied resources are : 4
Total inaccessible resources are : 3
Invalid status value as occupied resources are more than free resources
1. Display Status
2. Set Status
3. Exit
1
1 1 0
0 1 1
0 0 0
```