Наследование:

С++

class Person

{

public:

    string name;       //  имя

    int age;                // возраст

    void display()

    {

        cout << "Name: " << name << "\tAge: " << age << endl;

    }

};

class Employee : public Person

{

public:

    string company;    // компания

};

C#

class Person

{

    private string \_name = "";

    public string Name

    {

        get { return \_name; }

        set { \_name = value; }

    }

    public void Print()

    {

        Console.WriteLine(Name);

    }

}

class Employee : Person

{

}

Java

class Person {

    String name;

    public String getName(){ return name; }

    public Person(String name){

        this.name=name;

    }

    public void display(){

        System.out.println("Name: " + name);

    }

}

class Employee extends Person{

    public Employee(String name){

        super(name);    // если базовый класс определяет конструктор

                        //  то производный класс должен его вызвать

    }

}

Перегрузка:

С++

Counter operator + (Counter c1, Counter c2)

{

    return Counter(c1.seconds + c2.seconds);

}

Counter operator - (Counter c1, Counter c2)

{

    return Counter(c1.seconds - c2.seconds);

}

bool operator == (Counter c1, Counter c2)

{

    return c1.seconds == c2.seconds;

}

bool operator != (Counter c1, Counter c2)

{

    return c1.seconds != c2.seconds;

}

bool operator > (Counter c1, Counter c2)

{

    return c1.seconds > c2.seconds;

}

bool operator < (Counter c1, Counter c2)

{

    return c1.seconds < c2.seconds;

}

C#

public static Counter operator +(Counter counter1, Counter counter2)

    {

        return new Counter { Value = counter1.Value + counter2.Value };

    }

    public static bool operator >(Counter counter1, Counter counter2)

    {

        return counter1.Value > counter2.Value;

    }

    public static bool operator <(Counter counter1, Counter counter2)

    {

        return counter1.Value < counter2.Value;

    }

Абстрактные:

С++

class Figure

{

public:

    virtual double getSquare() =0;

    virtual double getPerimeter() =0;

    virtual void showFigureType()=0;

};

class Rectangle : public Figure

{

private:

    double width;

    double height;

public:

    Rectangle(double w, double h) : width(w), height(h)

    {

    }

    double getSquare() override

    {

        return width \* height;

    }

    double getPerimeter() override

    {

        return width \* 2 + height \* 2;

    }

    void showFigureType()

    {

        std::cout << "Rectangle" << std::endl;

    }

};

class Circle : public Figure

{

private:

    double radius;

public:

    Circle(double r) : radius(r)

    {

    }

    double getSquare() override

    {

        return radius \* radius \* 3.14;

    }

    double getPerimeter() override

    {

        return 2 \* 3.14 \* radius;

    }

    void showFigureType()

    {

        std::cout << "Circle" << std::endl;

    }

};

C#

abstract class Transport

{

    public abstract void Move();

}

// класс корабля

class Ship : Transport

{

    // мы должны реализовать все абстрактные методы и свойства базового класса

    public override void Move()

    {

        Console.WriteLine("Корабль плывет");

    }

}

// класс самолета

class Aircraft : Transport

{

    public override void Move()

    {

        Console.WriteLine("Самолет летит");

    }

}

// класс машины

class Car : Transport

{

    public override void Move()

    {

        Console.WriteLine("Машина едет");

    }

}

Java

// абстрактный класс фигуры

abstract class Figure{

    float x; // x-координата точки

    float y; // y-координата точки

    Figure(float x, float y){

        this.x=x;

        this.y=y;

    }

    // абстрактный метод для получения периметра

    public abstract float getPerimeter();

    // абстрактный метод для получения площади

    public abstract float getArea();

}

// производный класс прямоугольника

class Rectangle extends Figure

{

    private float width;

    private float height;

    // конструктор с обращением к конструктору класса Figure

    Rectangle(float x, float y, float width, float height){

        super(x,y);

        this.width = width;

        this.height = height;

    }

    public float getPerimeter(){

        return width \* 2 + height \* 2;

    }

    public float getArea(){

        return width \* height;

    }

}

Интерфейсы:

C#

interface IMovable

{

    // константа

    const int minSpeed = 0;     // минимальная скорость

    // статическая переменная

    static int maxSpeed = 60;   // максимальная скорость

    // метод

    void Move();                // движение

    // свойство

    string Name { get; set; }   // название

}

class Movable : IMovable

{

// константа

    const int minSpeed = 0;     // минимальная скорость

    // статическая переменная

    static int maxSpeed = 60;   // максимальная скорость

    // метод

    void Move()

{

// реализация метода - движение

}

    // свойство

    string Name { get; set; }   // название

}

Java

interface Printable{

    void print();

}

class Book implements Printable{

    String name;

    String author;

    Book(String name, String author){

        this.name = name;

        this.author = author;

    }

    public void print() {

        System.out.printf("%s (%s) \n", name, author);

    }

}

Шаблоны:

С++

template <typename T>

T myMax(T x, T y)

{

    return (x > y) ? x : y;

}

C#

class Person<T>

{

    public T Id { get; set; }

    public string Name { get; set; }

    public Person(T id, string name)

    {

        Id = id;

        Name = name;

    }

}

Java

class Account<T>{

    private T id;

    private int sum;

    Account(T id, int sum){

        this.id = id;

        this.sum = sum;

    }

    public T getId() { return id; }

    public int getSum() { return sum; }

    public void setSum(int sum) { this.sum = sum; }

}