Class & Object

Name: Alice
Age: 15
Job: student

setInformation → (Alice, 15, student)
printInformation → Alice, 15, student
Class & Object

[reality] → Object: exists in real

[shadow] → Class: defines the object

class Person {
private:
    string name;
    int age;
    string job;

public:
    void setInformation();
    void printInformation();
};
Define the class

class Person {
private:
    string name;
    int age;
    string job;
public:
    void setInformation();
    void printInformation();
};

Create the object

Person A, B;
class Person {
private:
    string name;
    int age;
    string job;
public:
    void setInformation();
    void printInformation();
};

void Person::setInformation() {
    cin >> name >> age >> job;
}

void Person::printInformation() {
    cout << "Person name is " << name << endl;
    cout << "Age is " << age << endl;
    cout << "Job is " << job << endl;
}
Inheritance

: can make new class with existing class

| shape | Rectangle | Triangle |
Inheritance

class Shape {
  protected:
    int width, height;
  public:
    Shape(int a=0, int b=0)
    {
      width = a;
      height = b;
    }
    virtual int area()
    {
      cout << "Parent class area:" << endl;
      return 0;
    }
};

class Triangle : public Shape{
public:
  int area()
  {
    cout << "Triangle class area:" << endl;
    return (width * height / 2);
  }
};

class Rectangle : public Shape{
public:
  int area()
  {
    cout << "Rectangle class area:" << endl;
    return (width * height);
  }
};
Polymorphism

- can have different forms
- C++ polymorphism means that a call to a member function will cause a different function to be executed depending on the type of object that invokes the function

```cpp
Shape *shape;
Rectangle rec;
Triangle tri;
// store the address of Rectangle
shape = &rec;
// call rectangle area.
shape->area();

// store the address of Triangle
shape = &tri;
// call triangle area.
shape->area();
```
Example

Make car class has
- attributes:
  - car’s color (string)
  - number of doors (int)
- methods:
  - void drive() { cout << “Driving” << endl;}
  - virtual void function() { cout << “ this is car” << endl;}

Make two child class
1. FireEngine
   → void function() { cout << “water” << endl;}
2. Ambulance
   → void function() { cout << “siren” << endl;}