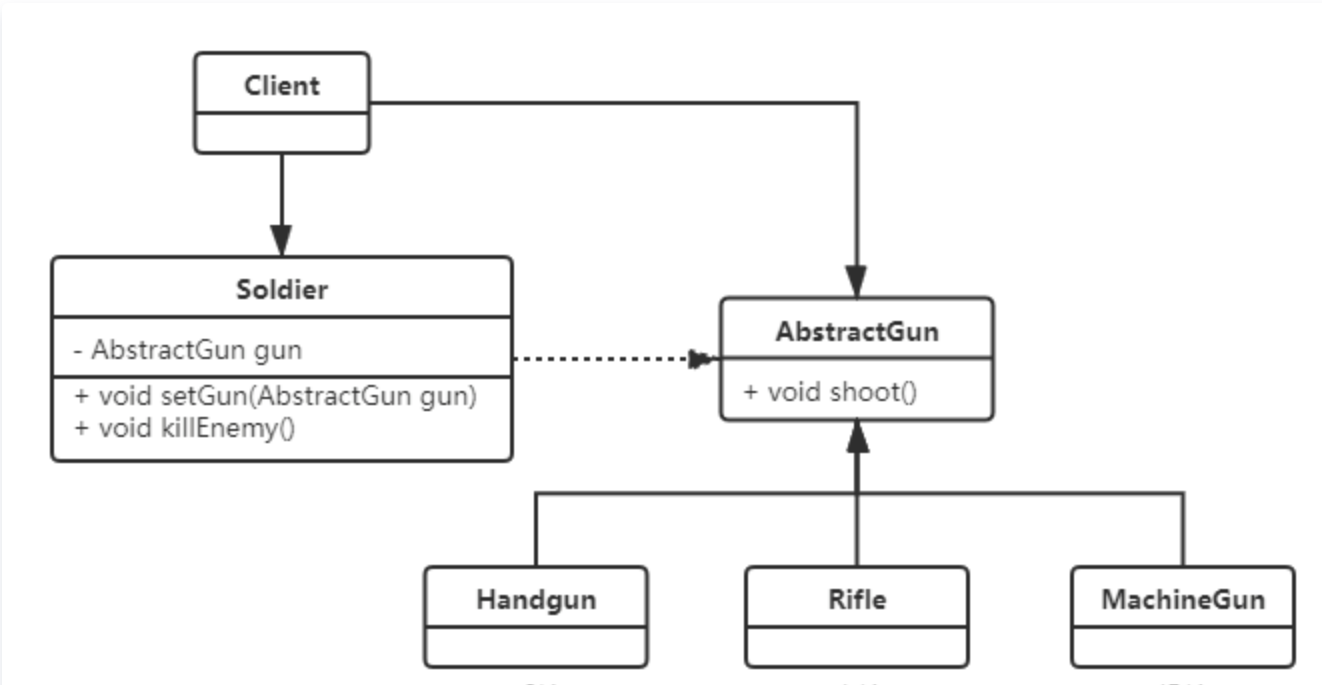
Client Sever

Client server architecture is a kind of two-layer architecture normally.In this structure, the application is divided into two parts: the server part and the client part. The server part share the information and functions to multiple users, and performs background services, such as controlling the operation of the shared database and part of the data is exclusively for the user, responsible for performing the front desk functions.It also has powerful functions in error prompts, online help, etc. , What is more,it can switch freely between subprograms.

In our project, we divide it to three-layer which is client,edge server and main server.We designed the edge server as a middle layer to connect the client and main server and it is not only a data transfer station,but also deal with some logical calculation.That means the edge server will calculate or filter the huge data firstly in order to change the data to be the handy result for the main server when it receive the data from the client and then send the result.When the data is returned,the edge server will filter the necessary data and send it to the client.In this way,it can reduce the delay caused by the interaction between client and server or internet congestion so that the game can react faster.At the same time,it can reduce the risk of main server overload and keep the security of data because of the less data transmission.There is an other advantage is that the edge server can store and resend the data when the main server down which prevent the data losing.

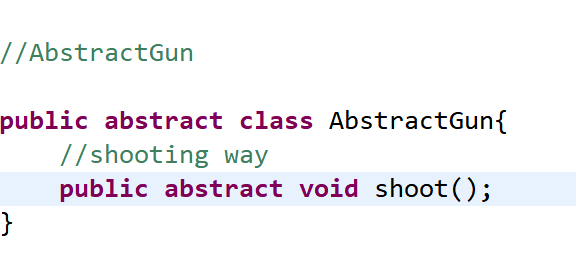
The main features using client server architecture is that this structure have strong interaction and fast response.The Liskov Substitution Principle means as long as the parent class can appear, the child class can appear, and the replacement of the child class will not produce any errors or exceptions.This guarantee the strong interaction and fast response.The following explain the Liskov Substitution Principle with a shooting game in different way.

Subclass must fully implement the method of the parent class.When we design a system, we often define an interface or abstract class, and then code to achieve, the calling class is directly passed into the interface or abstract class.Such as when we describe the gun in the CS

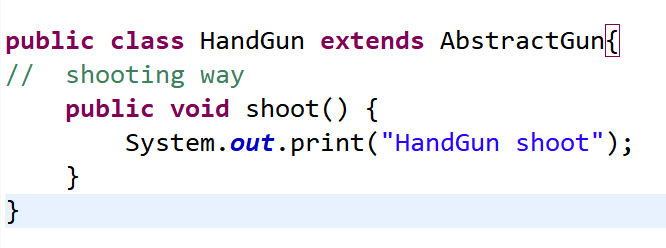


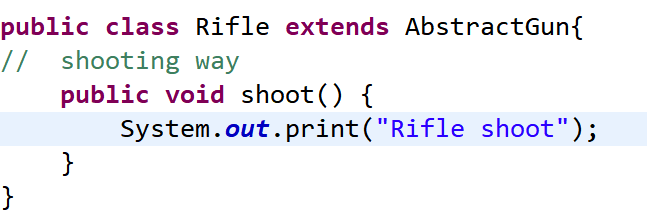
The main responsibility of the gun is shooting. How to shoot is implemented in each specific subclass. The soldier class Soldier defines a method killEnemy, which uses the gun to kill the enemy. The specific gun is only known when it is called.

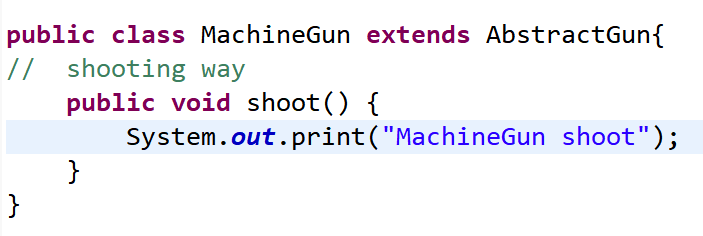
AbstractGun source code:



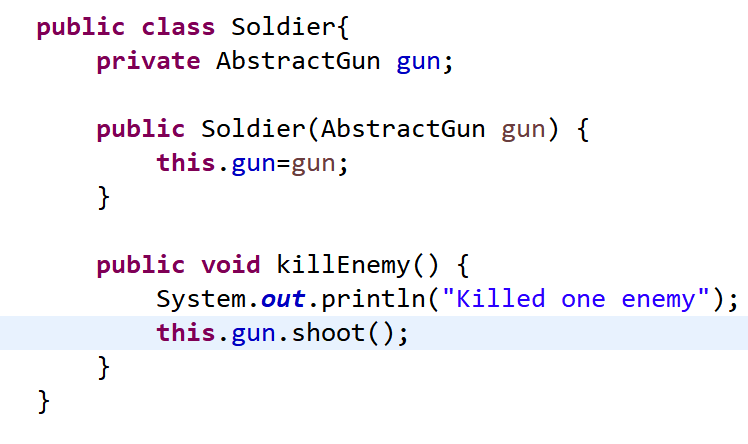
The code of handgun,rifle,machinegun implement:



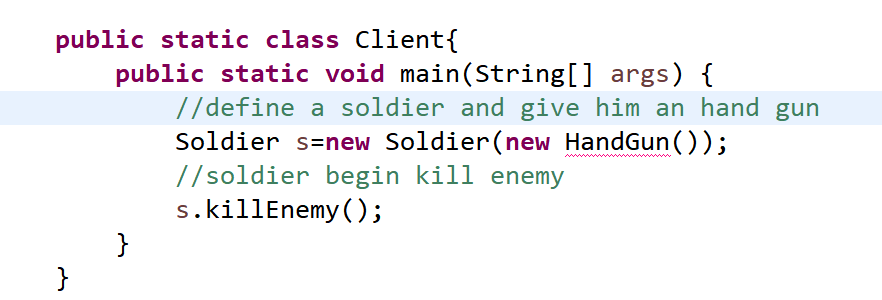




The source code of solider: The gun used in the killEnemy method of the soldier class is abstract, and the specific time of the gun needs to be determined by the client (Client) calling the constructor of the Soldier.



The code of client:



When calling other classes in a class, you must use the parent class or interface. If you cannot use the parent class or interface, it means that the design of the class has violated the LSP principle.This principle maintain the code sharing, reducing the workload of creating classes, each subclass has the attributes and methods of the parent class and also improve code reuse and scalability.