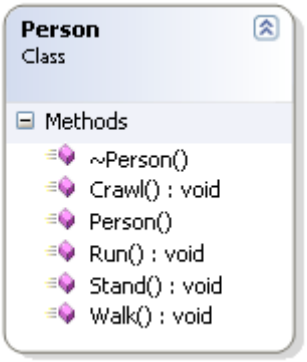
C++ Week 3 Classes

During this session you will create a set of classes to mimic the behaviour within a basic First-Person Shooter video game.

Exercise 1

Create a new C++ console project.
Define a new class called Person that conforms to the UML class opposite. Use the **Project->Add Class->C++ Class** wizard to create the class header and cpp files.

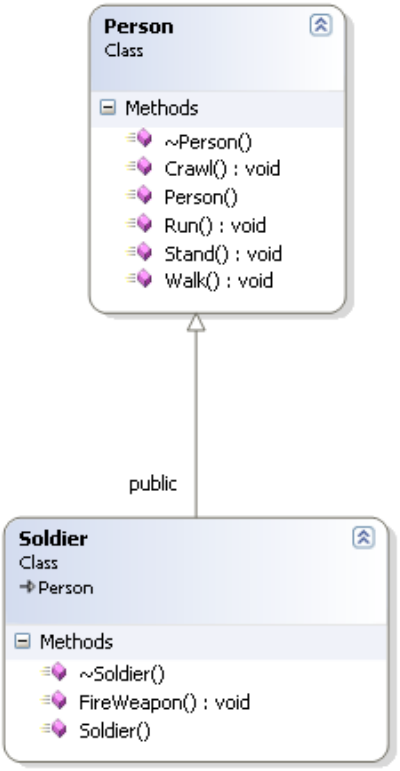


The four member functions describing the movements, should all include a single statement that outputs the movement to the console, **"Standing"** etc

Within the main() function create an dynamic instance of person and call each of the movement member functions in turn (remember to include person.h).
Note: By including <iostream> and using namespace std in the header file it will be available to the person.cpp file and all files that include person.h.

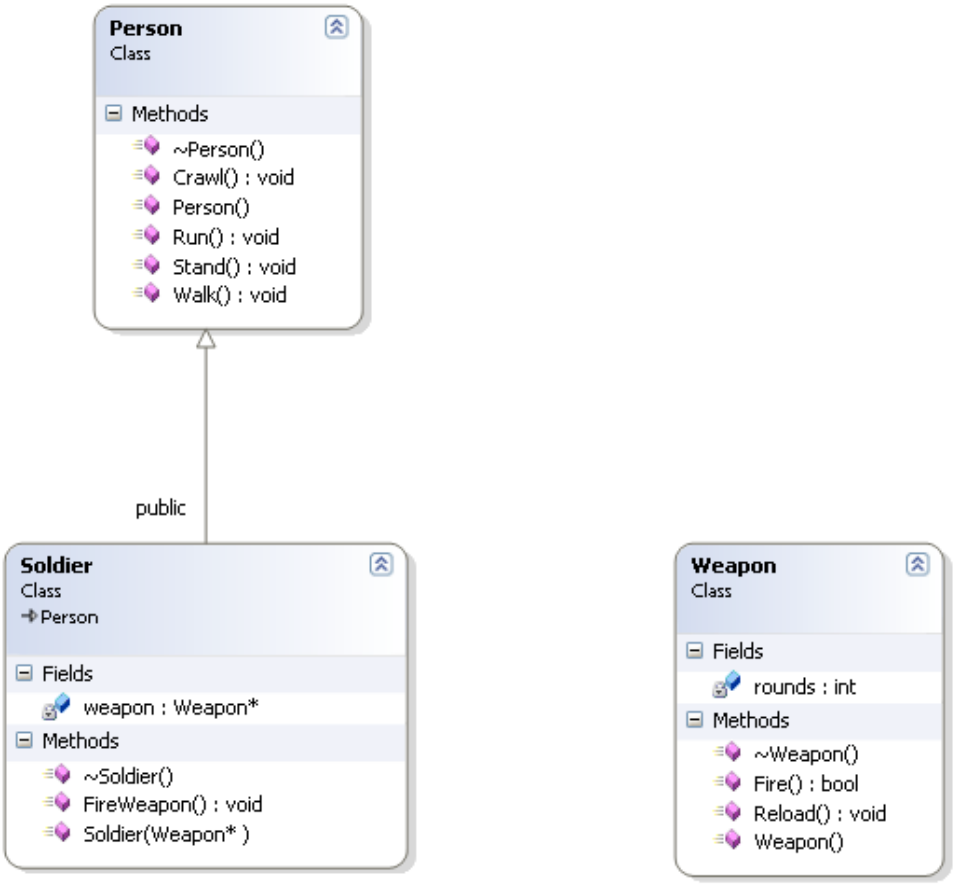
Exercise 2

- Using the wizard add the Soldier class to your project. It is **publicly** derived from Person, so add Person as the Base class within the wizard.
- Soldier has an additional member function FireWeapon, within this simply output the text **"Firing Weapon"**
- Now create a dynamic instance of this class within main and call the Stand and FireWeapon member functions.



C++ Week 3 Classes

Exercise 3



- Using the wizard create the Weapon class illustrated above.
- The rounds should be set to 10 within the constructor.
- Within Fire() output the message **"Weapon Fired"** and decrement the rounds by one and return true.
- However only do this if there are rounds left, if there aren't return false.
- The reload member function increments the rounds by 10 and displays the message **"Reloading"**.
- Each soldier has a weapon which is assigned to him within the constructor. So within soldier create a pointer variable to hold the weapon. Be sure to include weapon.h within soldier.h
- When instantiating a soldier you must now pass it a pointer to a weapon object, as an argument to its constructor.
- Finally within Soldier's FireWeapon method comment out the **"Firing Weapon"** message and call the weapon's Fire method.
- Make sure the weapon pointer variable has been assigned a weapon before you try calling its methods hint: != 0

Run the game once more.

C++ Week 3 Classes

Exercise 4

It should not be possible for a soldier to fire his/her weapon whilst crawling. Implement some state controlling code using enumerate values to ensure this can't happen and test it out.

Hint:

- Within the Person class create an enumerate type with the values crawl, stand, walk and run.
- Now declare a variable of this type within Person.
- Ensure the variable is updated when the movement functions are invoked.
- Check the variable has not been assigned the value crawl before firing the weapon in Soldier.

Test the new code.

Exercise 5

It should be possible to fire a weapon in automatic mode as well as the current single shot mode.

- Create an overload member function Fire() (within Weapon) that takes a single integer argument (the number of rounds fired in the burst).
- The new overloaded member function should decrement the rounds appropriately. Display a **"Automatic Weapon Fire"** message from within the new member function and return true there were rounds to fire otherwise false.
- Create a new overload FireWeapon() (include the rounds to fire as a formal parameter) member function within Soldier to call the new Weapon member function.

Your classes should resemble those below.

Invoke the new Soldier's FireWeapon() from within main and run the program.

C++ Week 3 Classes

Exercise 6

We will now create a health system for our Person objects and tell the Weapon which Person has been hit so that their health can be decremented.

- Within Person declare a new public **health** data member (int) and set it to an initial value of 100 within the constructor.
- Declare and Define a public Damage member function within Person that takes a single parameter level of type int.
- Damage should reduce health by level and output **"Damaged"** to the console.
- Include a person pointer parameter within all the overloaded FireWeapon and Fire member functions.
- Within the Fire member functions of weapon call the Damage method, reducing the health by the number of rounds fired.
- Within Soldier whenever firing the weapon pass the Person object that is being fired at as an argument.
- Make the appropriate changes in main that ensures the soldier fires at the person object instantiated in exercise 1 and display their health value after being shot.
- Test the modified program Fully!