Week 15 : C++ VS JAVA

Part1. Method Invocation and member Initialization

- Method Invocation

C++	Class::method();
Java	Class.method();

- Member Initialization

C++	No In-Class Initialization **e.g** static int i ;
Java	In-Class Initialization **e.g** public static int i = 10;

Part2. C++ Virtual function and JAVA Overriding

- C++ : When you call sub-classes method, you have to use virtual funtion

```
class First
{
public:
        virtual void MyFunc() { cout<<"FirstFunc"<<endl; }
};

class Second: public First
{
public:
        virtual void MyFunc() { cout<<"SecondFunc"<<endl; }
};
```

- Java

```
class First {
        void MyFunc() {
                System.out.println("FirstFunc");
}

class Second extends First {
        void MyFunc() {
                System.out.println("SecondFunc");
        }
}
```

Part3. Multiple Inheritance

C++ : support multiple inheritance

```
#include <iostream>
#include <string.h>
using namespace std;

class Actor
{
     public:
          void printJob();
};

void Actor::printJob()
{
          cout << "Actor" << "₩n"
}

class Singer
{
     public:
          void printJob();
};

void Singer::printJob()
{
        cout << "Singer" << "₩n"
}

class ActorSinger : public Actor, public Singer
{
    public :
        void printJob();
};

void ActorSinger::printJob()
{
        cout << "ActorSinger" << "₩n"
}

void main()
{
        ActorSinger worker;
        worker.printJob();
        worker.Actor::printJob();
        worker.Singer::printJob();
}
```

Java : not support multiple inheritance, you can make multiple inhertance using an 'interface'

```
interface iTest1{}
interface iTest2{}
interface iTest3 extends iTest1, iTest2{} (o,x)
```

```
class Test1{}
class Test2{}
class Test3 extends Test1, Test2{}   (o,x)
class mainTest extends Test2 implements iTest2, iTest3{}   (o,x)

public class testJava{
public static void main(String[] args){
        mainTest test = new mainTest();
        }
}
```

oxo

Part4. C++ Operator Overloading

C++ : two forms of the operator overloading

① Member function

```
class Point
{
public:
        Point operator+(Point &p);
};

Point Point::operator+(Point &p)
{
        Point temp(x+p.x,y+p.y);
        return temp;
}
```

$$\begin{tikzcd}
p1 & + & p2 \\
p1 & .operator+ & (p2)
\arrow[from=1-1, to=2-1, "\textcircled{1}"]
\arrow[from=1-2, to=2-2, "\textcircled{2}"]
\arrow[from=1-3, to=2-3, "\textcircled{3}"]
\end{tikzcd}$$

① object which calls member function

② function name

③ function's parameter

② Global function

```
class Point
{
public :
        friend Point operator+(Point &p);
};

Point operator+(Point &p1,Point &p2)
{
        Point temp(p1.x+p2.x,p1.y+p2.y);
        return temp;
}
```

friend Point operator+(Point &p);
using 'friend' to access 'p' declared by private

```
p1            +          p2
operator+     (p1        ,p2)
```

Part5. C++ Namespace

```
#include <iostream>
namespace AAA
{
	int num=5;

	namespace BBB
	{
		int num=6;
	}
	namespace CCC
	{
		int num=7;
	}
}

int main()
{
	std::cout<<"AAA_num:"<<AAA::num<<"
BBB_num:"<<AAA::BBB::num<<"
CCC_num:"<<AAA::CCC::num<<std::endl;
	return 0;
}
```

Part6. C++ Default Parameter

int adder(int num1, int num2=0)
{
return num1+num2;
}
	OK

int adder(int num1=0, int num2)
{
return num1+num2;
}
	ERROR

[Exercise]

Make a Student class which contains String name and double GPA value. Consider the return type of compareTo method. Print out student based on GPA.

int compareTo(T o)

Returns:a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

```
public class ObjArray {

 public static void main(String[] args) {
  Student[] students = new Student[4];
  students[0] = new Student("Mike", 1.39);
  students[1] = new Student("Bob", 4.23);
  students[2] = new Student("Mary", 2.19);
  students[3] = new Student("Jake", 3.29);

  Arrays.sort(students);
  for (Student s : students) {
   System.out.println("Name=" + s.getName()+" GPA=" + s.getGPA());
  }
 }

}

class Student implements Comparable{
 private String name;
 private double gpa;

 /* BLANK * /

 public int compareTo(Object obj) {
  /* BLANK */
 }
}
```