定義類別的建構子

1. 它的作用就是在記憶體資料區(Data)

   <1>.配置物件實體(因為 new)➔跟記憶體要一個連續空間

   <2>.所寫的程式碼內容，通常是用來把產生的物件初始化的➔清乾淨

   <3>.帶地址回來

2. 建構子的函數名與類別名必須完全相同
3. 無傳回值，無修飾字
4. 建構函數只能配合 “new” 來自動呼叫，而無法讓物件使用

預設的建構子

1. 在所宣告的類別內，如果未定義任何的建構子，那麼編譯器會自動給它一個 “預設建構子” (不需傳入參數的那種)
2. 以 Person 為例：如果我們並未替 Person 類別定義任何的 “建構子” ，仍然可利用 new Person()來建構物件實體，不過，由於並未設定成員變數之值，所以產生的實體中值為 null，0，0 (預設初始化)
3. 當我們在 Person 類別內定義了任何一個 “建構子” 之後，Person 類別就不會自動擁有 “預設建構子” ，所以我們無法使用 new Person( )，來產生物件實體

```java
public class Person {

    public String 名字;
    public int 身高;
    public int 體重;
    public static int 人數;

    public static void 我愛你() {
        System.out.println("我愛你");
    }

    public void 秀名字身高體重人數() {
        System.out.println("名字=" + 名字);
        System.out.println("身高=" + 身高);
        System.out.println("體重=" + 體重);
        System.out.println("目前人數=" + 人數);
    }
}
```

我們可以將它視為

```java
public class Person {

    public String 名字;
    public int 身高;
    public int 體重;
    public static int 人數;

    public Person() {
        super();
    }

    public static void 我愛你() {
        System.out.println("我愛你");
    }

    public void 秀名字身高體重人數() {
        System.out.println("名字=" + 名字);
        System.out.println("身高=" + 身高);
        System.out.println("體重=" + 體重);
        System.out.println("目前人數=" + 人數);
    }
}
```

```java
public class Person {

    public String 名字;
    public int 身高;
    public int 體重;
    public static int 人數;

//    public Person() { //預設的被收走了
//        super();
//    }
    public Person(String n, int h, int w) { //自行定義
        //super()
        名字 = n;
        身高 = h;
        體重 = w;
    }
    public static void 我愛你() {
        System.out.println("我愛你");
    }
    public void 秀名字身高體重人數() {
        System.out.println("名字=" + 名字);
        System.out.println("身高=" + 身高);
        System.out.println("體重=" + 體重);
        System.out.println("目前人數=" + 人數);
    }
}
```

```java
public static void 建構子1() {
    Person p1 = new Person("林青霞", 162, 50);
    p1.秀名字身高體重人數();

    Person p2 = new Person();
    //上行錯誤，Person 類別沒有 Person() 這個建構子
}
```

```
林青霞： 有！ 身高=162.0 體重=50.0
```

初始化順序 與 super()➔繼承發生時，子類別不需繼承父親的建構函數，但卻會從 super() 轉向到父親，順便執行父親的建構函數

因此 new 子類別時，因為 super() 的關係，也會自動產生父類別的實體➔因此 執行一次 super() 就等於執行一次 new 父類別()

1. 預設初始化
2. 明顯初始化
3. 建構函數初始化

```
//class Object{
//
//}
public class New1P {

//public int x=0 預設始始化
    public int x = 1; //明顯始始化
//public int y=0 預設始始化
    public int y = 2; //明顯始始化

    public New1P() {
        super();
        x = 11; //建構函數初始化
        y = 12; //建構函數初始化
    }

    public void s1() {

    }
}

class New1C extends New1P {

//public int a=0 預設始始化
    public int a = 3; //明顯始始化
//public int b=0 預設始始化
    public int b = 4; //明顯始始化

    public New1C() {
        super();
        a = 13; //建構函數初始化
        b = 14; //建構函數初始化
    }

    public void m1() {

    }
}
```

從 Object 轉向回 New1P
預設初始化 (父類別)
x=0
明顯初始化 (父類別)
x=1
預設初始化 (父類別)
y=0
明顯初始化 (父類別)
y=2
建構函數初始化 (父類別)
x=11
y=12
從 New1P 轉向回 New1C
預設初始化 (子類別)
a=0
明顯初始化 (子類別)
a=3
預設初始化 (子類別)
b=0
明顯初始化 (子類別)
b=4
建構函數初始化 (子類別)
a=13
b=14

建構子的多載➔一個類別可以定義多個建構子，每個建構子的 “參數型式” 必須都不同 (型態，個數，順序)

```
public class Person5 {

    public String 名字;
    public int 身高;
    public int 體重;
    public static int 人數;
    public Head5 頭 = new Head5(); //明顯初始化
    public static Say5 說 = new Say5(); //明顯初始化

    public Person5() {
        super();
        人數++;
    }
    public Person5(String n) {
        super();
        名字 = n;
        人數++;
    }
    public Person5(String n, int h, int w) {
        super();
        名字 = n;
        身高 = h;
        體重 = w;
        人數++;
    }
    public Person5(String n, int h, int w, Head5 head) {
        super();
        名字 = n;
        身高 = h;
        體重 = w;
        頭 = head;
        人數++;
    }
    public void 秀名字身高體重人數() {...9 lines }

    public String toString() {...9 lines }
}

class Head5 {

    public String 眼睛;
    public String 鼻子;
    public String 嘴巴;

    public Head5() {
        super();
    }

    public Head5(String e, String n, String m) {
        super();
        眼睛 = e;
        鼻子 = n;
        嘴巴 = m;
    }
}

class Say5 {

    public void 我愛你() {
        System.out.println("我愛你");
    }

    public void 我恨你() {
        System.out.println("我恨你");
    }
}
```

```
public static void 建構子多載() {

    Person5 p1 = new Person5();
    System.out.println(p1);
    System.out.println("==============================");
    Person5 p2 = new Person5("張曼玉");
    System.out.println(p2);
    System.out.println("==============================");
    Person5 p3 = new Person5("林青霞", 162, 50);
    System.out.println(p3);
    System.out.println("==============================");
    //先把 head new 出來
    Head5 head = new Head5("藍色", "很挺", "很大");
    Person5 p4 = new Person5("林青霞", 162, 50, head);
    System.out.println(p4);
}
```

```
名字=null
身高=0
體重=0
眼睛=null
鼻子=null
嘴巴=null
目前人數=1

===========
名字=張曼玉
身高=0
體重=0
眼睛=null
鼻子=null
嘴巴=null
目前人數=2

===========
名字=林青霞
身高=162
體重=50
眼睛=null
鼻子=null
嘴巴=null
目前人數=3

===========
名字=林青霞
身高=162
體重=50
眼睛=藍色
鼻子=很挺
嘴巴=很大
目前人數=4
```

this 和 super 關鍵字

1. 類別內的兩個隱含的指標 this 指向自己，super 指向父親 只用在 物件等級，

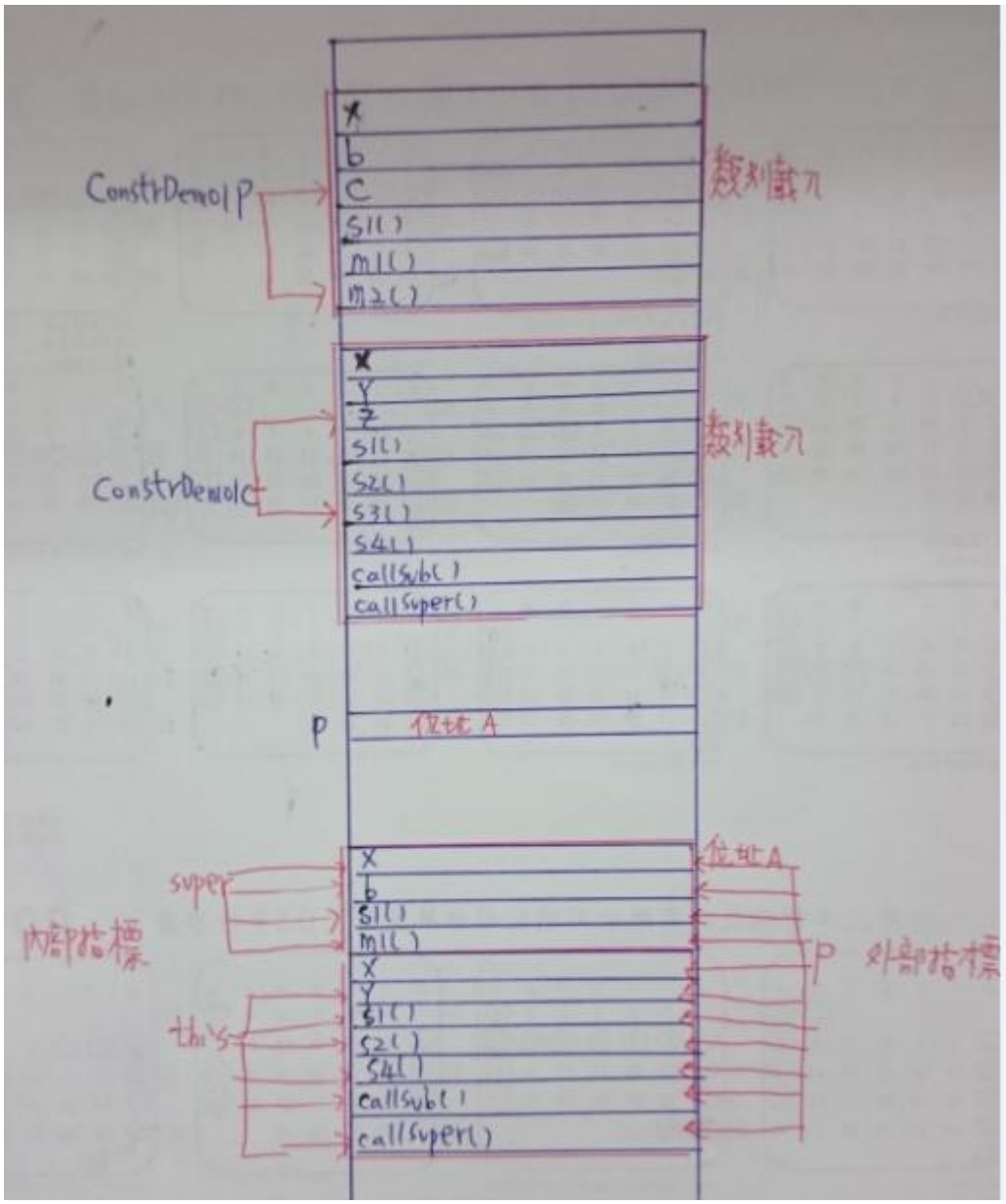
   ➔ super.成員 ➔存取繼承而來的成員，this.成員➔存取子類別的成員

   ➔ static 成員 沒有 this 與 super 指標

```java
public class ConstrDemo1P { //父

    public int x = 10;
    public int b = 20;
    public static int c = 30;

    public void s1() {
    }

    public void m1() {
    }

    public static void m2() {
    }
}
```

```java
//外面的類別
class ConstrDemo1T {

    public void t1() {

        ConstrDemo1C p = new ConstrDemo1C();
        System.out.println(p.x); //20.5 子
        System.out.println(p.b);//20 父
        System.out.println(ConstrDemo1P.c); //30 父
        System.out.println(ConstrDemo1C.c);//30 父
        p.callSuper(); //父
        p.m1(); //父
        ConstrDemo1P.m2(); //父
        ConstrDemo1C.m2(); //父
        ////////////////////////
        System.out.println(p.x); //20.5 子
        System.out.println(p.y); //40 子
        System.out.println(ConstrDemo1C.z);//50 子
        p.s1(); //子
        p.s2(); //子
        ConstrDemo1C.s3(); //子
        p.s4(10, 20); //子
    }
}
```

```java
class ConstrDemo1C extends ConstrDemo1P {//子

    public double x = 20.5;
    public int y = 40;
    public static int z = 50;

    public void s1() {
        System.out.println(super.x); //10
        System.out.println(super.b); //20
        System.out.println(this.b); //20
        System.out.println(b); //20
        System.out.println(ConstrDemo1P.c); //30
        System.out.println(ConstrDemo1C.c); //30
        System.out.println(c); //30
        /////////
        System.out.println(this.x); //20.5
        System.out.println(x); //20.5
        System.out.println(this.y); //40
        System.out.println(y); //40
        System.out.println(ConstrDemo1C.z); //50
        System.out.println(z); //50
    }

    public void s2() {
        super.s1();
        super.m1();
        this.m1();
        m1();
        ConstrDemo1P.m2();
        ConstrDemo1C.m2();
        m2();
        ///////
        this.s1();
        s1();
    }

    public static void s3() {
//static 內不能有 this指標 , 因為目前 this 是 null 不能使用
//        System.out.println(this.x);
//        System.out.println(x);
//        this.s1();
//        s1();
        ConstrDemo1C p = new ConstrDemo1C();
        p.x = 123;
    }

    public void s4(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public void callSuper() {
        super.s1(); // 父
    }
}
```


ConstrDemo1 P
x
b
c
s1()
m1()
m2()
類別載入
ConstrDemo1 C
x
y
z
s1()
s2()
s3()
s4()
callSub()
callSuper()
類別載入
P
位址A
super
內部指標
x
b
s1()
m1()
x
y
s1()
s2()
s4()
callSub()
callSuper()
this
位址A
P
外部指標

```
public class ConstrDemo2P {

    public int var1 = 100;

    protected void showVar1() {
        System.out.println("ConstrDemo2P 定義的 showVar1() var1 = " + var1);
    }
}
class ConstrDemo2C extends ConstrDemo2P {

    public double var1 = 1111.111;

    public void showVar1() {
        System.out.println("ConstrDemo2C 定義的 showVar1() var1 = " + var1);
        double add = var1 + super.var1;//編譯器會將 var1 視為 this.var1
        //double add = this.var1 + super.var1; //上行同於此行

        System.out.println("add = " + add);
    }

    public void callSuper() {
        super.showVar1(); //呼叫父類別的 showVar1()

    }
}
```

```
public static void this與super() {
    ConstrDemo2C obj = new ConstrDemo2C();
    obj.showVar1();
    System.out.println("===============");
    obj.callSuper();
}
```

```
ConstrDemo2C 定義的 showVar1() var1 = 1111.111
add = 1211.111
===============
ConstrDemo2P 定義的 showVar1() var1 = 100
```

2. 使用於建構子內的 “this(參數列)” 和 “super(參數列)” 只能放在第一行，有 super( ) 就不能放 this( )

<1>.super()

(1).在子類別的任何建構子內，預設的第一個敘述句就是呼叫父類別的 “預設建構子”

(2).假設父類別並沒有 “預設建構子” 則需在子類別的建構子內第一個敘述句使用 super(參數列) 來指定呼叫父類別的建構子

```
/*
 class Object{

 }
 */
class SuperSuper01 {

    public SuperSuper01() {
        //super();
        System.out.println("我是丫公 SuperSuper01");
    }
}

public class Super01 extends SuperSuper01 {

    public Super01() {
        //super();
        System.out.println("我是父親 Super01");
    }
}

class Sub01 extends Super01 {

    public Sub01() {
        //super();
        System.out.println("我是小孩 Sub01");
    }
}
```

```
/*
 class Object{

 }
 */
class Super02 {

    public int var1;

    public Super02() {
        // super(); //可以不寫
        System.out.println("我是父親 Super02");
    }

    public Super02(int var1) {
        //super(); //可以不寫
        System.out.println("我是父親 Super02(int)");
        this.var1 = var1;
    }
}

class Sub02 extends Super02 {

    public int var2;

    public Sub02() {
        //super(); //可以不寫，會自動呼叫 Super02()
        System.out.println("我是小孩 Sub02");
    }

    public Sub02(int var2) {
        super(var2); //必須要寫 ，指定傳入 var2 值來呼叫 Super02(int)
        System.out.println("我是小孩 Sub02(int)");
        this.var2 = var2;
    }
}
```

```
public static void 建構子2() {
    new Sub01();
}
```

```
public static void 建構子3() {
    new Sub02();
    new Sub02(100);
}
```

```
我是丫公 SuperSuper01
我是父親 Super01
我是小孩 Sub01
```

```
我是父親 Super02
我是小孩 Sub02
我是父親 Super02(int)
我是小孩 Sub02(int)
```

<2>.this()

(1).在一個類別的某個建構子想要呼叫此類別的 “另一個建構子” 可於建構子內的第一個敘述句利用 this(參數列) 來呼叫另一建構子，但此狀況就不得在此建構子內以 super(參數列) 呼叫父類別的建構子，也就是他們在一個建構子內只能夠二選一

(2).成員函數內定義了與 “成員變數” 同名的 “區域變數” ，就需要以 “this.成員變數” 的方式，來指定存取成員變數

```
class Super03 {

    public int var1;

    public Super03() {
        this(10); //呼叫兄弟建構函數
    }

    public Super03(int var1) {
        super(); //呼叫父親建構函數
        System.out.println("我是父親 Super03(int)");
        this.var1 = var1;
    }
}

class Sub03 extends Super03 {

    public int var2;

    public Sub03() {
        this(10); //呼叫兄弟建構函數
    }

    public Sub03(int var2) {
        super(var2); //呼叫父親建構函數
        System.out.println("我是小孩 Sub03(int)");
        this.var2 = var2;
    }
}
```

```
public static void 建構子4() {
    new Sub03();
}
```

```
我是父親 Super03(int)
我是小孩 Sub03(int)
```

```java
public class Person6 {

    public String 名字;
    public int 身高;
    public int 體重;
    public static int 人數;
    public Head6 頭 = new Head6();
    public static Say6 說 = new Say6();

    public Person6() {
        this("賴玉珊", 160, 50);
    }
    public Person6(String 名字) {
        super();
        this.名字 = 名字;
        人數++;
    }
    public Person6(String 名字, int 身高, int 體重) {
        super();
        this.名字 = 名字;
        this.身高 = 身高;
        this.體重 = 體重;
        人數++;
    }
    public Person6(String 名字, int 身高, int 體重, Head6 頭) {
        super();
        this.名字 = 名字;
        this.身高 = 身高;
        this.體重 = 體重;
        this.頭 = 頭;
        人數++;
    }

    public void 秀名字身高體重人數() {...9 lines }

    public String toString() {...9 lines }
}
```

```java
class Head6 {

    public String 眼睛;
    public String 鼻子;
    public String 嘴巴;

    public Head6() {
        super();
    }

    public Head6(String 眼睛, String 鼻子, String 嘴巴) {
        super();
        this.眼睛 = 眼睛;
        this.鼻子 = 鼻子;
        this.嘴巴 = 嘴巴;
    }
}

class Say6 {

    public void 我愛你() {
        System.out.println("我愛你");
    }

    public void 我恨你() {
        System.out.println("我恨你");
    }
}
```

組成類別 7 大項

```java
//類別組成
public class ConstrDemo3 {
    //成員屬性//////////////////////

    public int x;
    public static int y;

    //instance 區塊///////////////
    {
        x = 10;
    }
    //static 區塊////////////////

    static {
        y = 20;
    }
    //建構函數//////////////////

    public ConstrDemo3() {
        x = 30;
    }
    //成員方法//////////////////

    public void xyz() {
    }
    //巢狀類別//////////////////

    private static class ConstrDemo4 {

        int a;

        void abc() {
        }
    }
    //列舉//////////////////////

    public enum Color5 {

        RED, YELLOW, BLUE
    }
}
```

Instance 區塊範例

```java
package source;

public class Outer {

    public static void main(String[] args) {
        new Outer().abc();
    }

    public void abc() {
//        Inner p = new Inner();
//        p.show();
        Super p = new Super() {
            public int x;
            public int y;

            {
                x = 10;
                y = 20;
            }

            public void show() {
                System.out.println("x=" + x);
                System.out.println("y=" + y);
            }
        };
        p.show();
    }
}

////////////////////////////////////////////

abstract class Super { //父類別

    public abstract void show();
}

class Inner extends Super {//子類別

    public int x;
    public int y;

    public Inner() {

        x = 10;
        y = 20;
    }

    public void show() {
        System.out.println("x=" + x);
        System.out.println("y=" + y);
    }
}
```

類別與物件等級成員變數初始化的順序

1. 類別等級變數
   <1>.預設初始化
   <2>.明顯初始化
   <3>.static 區塊初始化➔類別載入時從頭到尾按照順序執行

2. 物件等級變數
   <1>.預設初始化
   <2>.明顯初始化
   <3>.instance 區塊初始化➔new 時從頭到尾按照順序執行
   <4>.建構子初始化

類別與物件等級常數(final) 的初始化

1. 物件等級的常數要在每個建構子中設定初始值 ➔因為不確定會 new 到那個建構函數
2. 注意➔設定全域常數的初始值不可在 “普通方法” 中設定➔因為不確定是否會呼叫到此方法

3. 類別等級的常數要在 其中一個 static 區塊中設定初始值➔因為 static 區塊是無條件從頭到尾執行一次

```
public class ConstrDemo5 {

    public int x = 1;
    public int y = 2;
    public static int z = 3;

    {
        x = 4;
        y = 5;
    }
    static {
        z = 6;
    }
    public ConstrDemo5() {
        x = 100;
        y = 100;
    }
    public ConstrDemo5(int x, int y) {
        this.x = x;
        this.y = y;
    }
    static {
        z = 100;
    }
    {
        x = 99;
        y = 99;
    }
    public static void 印1() {
        System.out.println("z=" + z);
    }
    public void 印2() {
        System.out.println("x=" + x);
        System.out.println("y=" + y);
    }
}
```

```
public class ConstrDemo6 {

    public int x = 1;
    public int y = 2;
    public static int z = 3;
    public final int a; //物件等級常數
    public static final int b; //類別等級常數

    {
        x = 4;
        y = 5;
    }
    static {
        z = 4;
        b = 2;
    }
    public ConstrDemo6() {
        this(100, 100);
        // a = 100;
    }
    public ConstrDemo6(int x, int y) {
        this.x = x;
        this.y = y;
        a = 100;
    }
    static {
        z = 100;
        // b = 9; //不能再設定一次
    }
    {
        x = 99;
        y = 99;
    }
    public void abc() {
        // a=10; 不可在方法內設定 常數值
    }
}
```

```
public static void 初始化的順序1() {
    ConstrDemo5.印1();
    new ConstrDemo5().印2();
}
```

```
z=100
x=100
y=100
```

```
//class Object{
//
//}
public class ConstrDemo7P {
    public int a = 1;   //6
    public static int b = 2;  // 1
    {
        a = 11;
        System.out.println("7");
    }
    static {
        b = 22;
        System.out.println("2");
    }
    public ConstrDemo7P() {
        //super();
        System.out.println("8");
    }
}
class ConstrDemo7C extends ConstrDemo7P {
    public int x = 1;  //9
    public static int z = 3;  //3
    {
        x = 4;
        System.out.println("10");
    }
    static {
        z = 6;
        System.out.println("4");
    }
    public ConstrDemo7C() {
        x = 100;
        System.out.println("12");
    }
    static {
        z = 100;
        System.out.println("5");
    }
    {
        x = 99;
        System.out.println("11");
    }
}
```

```
2
4
5
7
8
10
11
12
```

```java
public class ConstrDemo8P {
    public int x = 1;
    public static int y = 2;
    {
        x = 11;
        System.out.println("我是父親instance區塊1");
    }
    static {
        y = 22;
        System.out.println("我是父親static區塊1");
    }
    public ConstrDemo8P() {
        //super();
        System.out.println("我是父親建構函數");
    }
    {
        x = 21;
        System.out.println("我是父親instance區塊2");
    }
}
class ConstrDemo8C extends ConstrDemo8P {
    public int a = 2;
    public static int b = 4;
    {
        a = 21;
        System.out.println("我是小孩instance區塊1");
    }
    static {
        b = 8;
        System.out.println("我是小孩static區塊1");
    }
    public ConstrDemo8C() {
        //super()
        a = 11;
        System.out.println("我是小孩建構函數");
    }
    static {
        b = 10;
        System.out.println("我是小孩static區塊2");
    }
}
```

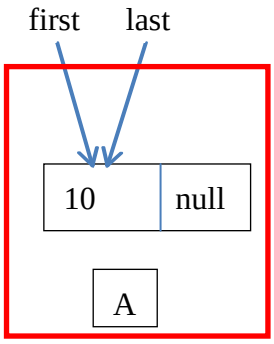
```
我是父親static區塊1
我是小孩static區塊1
我是小孩static區塊2
我是父親instance區塊1
我是父親instance區塊2
我是父親建構函數
我是小孩instance區塊1
我是小孩建構函數
```

```
xx 33 yy 22 11 aa bb cc dd 99
33 22 11 aa cc
```

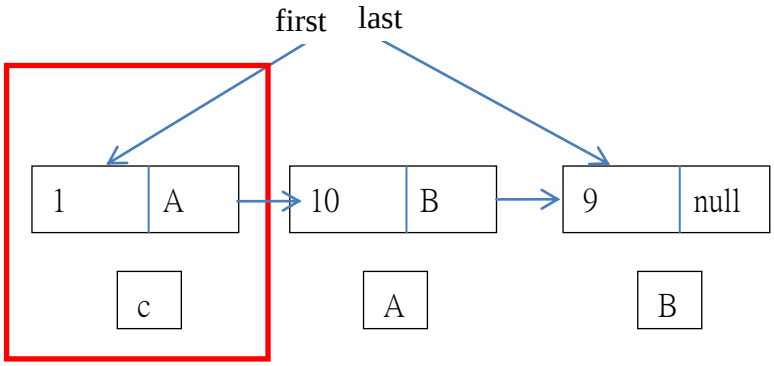
```java
public class TestMyList {

    public static void main(String[] args) {

        MyList myList = new MyList();

        myList.addFirst("11");
        myList.addFirst("22");
        myList.addFirst("33");
        myList.addLast("aa");
        myList.addLast("bb");
        myList.addLast("cc");
        myList.addLast("dd");
        myList.add(0, "xx");
        myList.add(2, "yy");
        myList.add(10, "99");
        ////////////////////////
        myList.removeFirst();
        myList.removeLast();
        myList.remove(1);
        myList.remove(99);
        myList.remove(4);

        System.out.println(myList);
    }
}
```

```java
class Node {

    public String value;
    public Node address;

    public Node(String value, Node address) {
        this.value = value;
        this.address = address;
    }
}
```

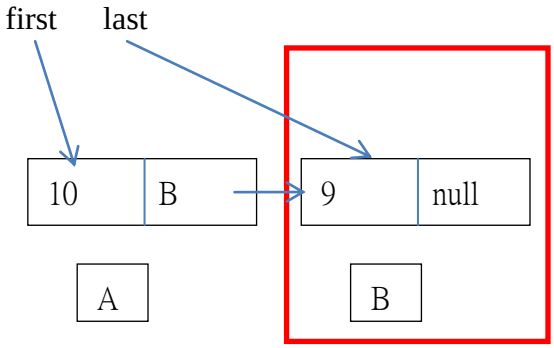
```java
class MyList {
    public Node first, last;
    public int size;

    public void addFirst(String item) {
        Node newNode = new Node(item, null);
        newNode.address = first;
        first = newNode;
        size++;

        if (last == null) {
            last = first;
        }
    }
    public void addLast(String item) {

    }
    public void add(int index, String item) {

    }
    public void removeFirst() {
        first = first.address;
        size--;
    }
    public void removeLast() {

    }
    public void remove(int index) {

    }
    public String toString() {

        StringBuilder str = new StringBuilder();
        Node current = first;
        while (current != null) {
            str.append(current.value).append(" ");
            current = current.address;
        }
        return str.toString();
    }
}
```



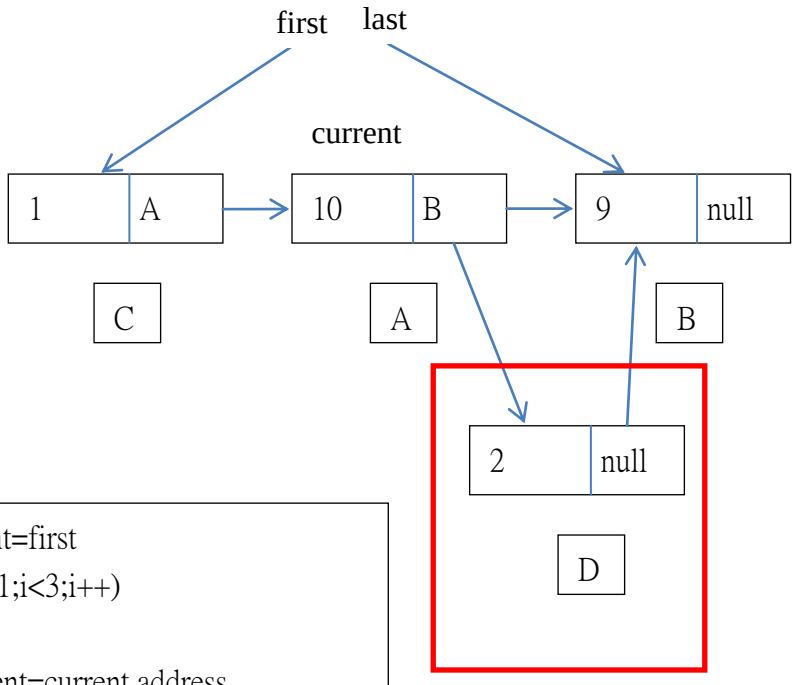
first=last=new Node(10,null)



newnode=new Node(1,null)
newnode.address=first
first=newnode



newnode=new Node(9,null)
last. address = newnode
last=last.address




```
current=first
for(i=1;i<3;i++)
{
 current=current.address
}
 temp=current.address
   newnode=new Node(2,null)
 current. address = newnode
 (current.address) .address =temp
```