**封裝**

封裝的目的是要可以達到資訊隱藏，隱藏物件細節，將物件當黑箱進行操作，程式設計師與使用者各司其職，對程式設計師而言必須仔細了解封裝功能撰寫內容，對使用者而言只要知道怎麼用就好了，而不需要了解是怎麼實作。

**封裝初始流程**

利用java建構式語法實現物件初始化流程，使用者可以不用重複撰寫物件初始化，也不用知道物件如何初始化。

Ex:

class CashCcard{

int balance;

int bonus;

}

CashCcard card1=new CashCcard();

card1.balance=500;

card1.bonus=0;

CashCcard card2=new CashCcard();

card2.balance=100;

card2.bonus=1;

封裝初始流程

class CashCcard{

int balance;

int bonus;

CashCcard(int balance,int bonus)

{

this.balance=balance;

this.bonus=bonus;

}

}

CashCcard card1=new CashCcard(500,0);

CashCcard card2=new CashCcard(100,1);

**封裝物件操作的流程**

在類別裡定義方法(method)封裝操作物件的流程

Ex:

class CashCcard{

int balance;

int bonus;

CashCcard(int balance,int bonus)

{

this.balance=balance;

this.bonus=bonus;

}

void store(int money){

this.balance+=money;

}

}

CashCcard card1=new CashCcard(500,0);

card1.store(100);

使用者只能透過定義的方法去操作物件，但上述的程式碼仍有瑕疵

CashCcard card1=new CashCcard(500,0);

card1.balance+=100;

這是因為沒有封裝不想讓使用者直接存取的私有資料。

**封裝物件內部資料**

將資料用private修飾詞，封裝類別私有資料，讓使用者無法直接存取，只能透過程式設計師所提供的操作方法，取得私有資料。

Ex:

class CashCcard{

private int balance;

private int bonus;

int getBalance(){

return balance;

}

Int getBonus(){

return bonus;

}

**修飾詞**

**Public**在其他套件下，可在其他套件類別使用。

**Private**只能在相同套件下，才能直接存取。

**建構式**

建構物件時，資料成員就會初始化，如果沒有指定初始值，則會使用預設值。

如果定義類別時，沒有撰寫任何建構式，**編譯器會自動加入一個無參數，內容為空的建構式。**

定義方法時可以進行**重載**，方法名稱相同，但根據參數型態或個數的不同呼叫而有對應的方法。

**Static 類別成員**

被宣告為static的成員，不會讓個別物件擁有，而是屬於類別。是將類別名稱作為命名空間。

Ex:

class Ball{

double radius;

static final double PI=3.141596;

}

System.out.print(Ball.PI);

}

透過類別名稱與.可以取得static成員。

也可以宣告方法為static成員。

由於static成員屬於類別，而非個別物件，所以在static方法或區塊不能出現this關鍵字(this 表示這個物件)，或非static成員。因為會有一種語意的錯誤，所以只能在static方法或區塊用static資料成員。