

inheritance

```
public class SavingAccount extends BankAccount{
private double interestRate;
...
public void addInterest(){
    double interest = getBalance() * interestRate/100;
    deposit(interest);
}
```



ถึงจะเป็นสับคลาส ก็ต้องใช้เมธอดเพื่อให้
อ่าน **private variable** ได้

บางคนใช้วิธีแก้ผิดๆ นี้ก็จะเข้าถึง **private variable**

ได้ ด้วยการนิยามตัวแปรใหม่ แต่มันผิดนะ

```
public class SavingAccount extends  
    BankAccount{
```

```
private double balance;
```



ไ้มันเป็นอีกตัวแปรหนึ่ง สรุปคือ ตอนนี คลาสนี้จะมีตัวแปร
balance สองตัว ตัวนี้มาจาก **BankAccount**

```
public class CheckingAccount extends BankAccount{
```

นี่คือการ **override**

```
public void deposit(double amount){
```

```
    counter++ ; // นี่คือการที่มีเพิ่มมาจากของเดิม
```

```
    deposit(amount); //พยายามเรียกเมธอดของ superclass
```

```
}
```

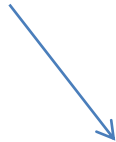
แต่ที่นี่จะกลายเป็นเรียกตัวเอง **infinite**

ต้องใช้ **super.deposit(amount)** แทน

ในการเขียนสับคลาส ต้องระวัง **overload** โดยไม่ตั้งใจ

```
public class CheckingAccount extends  
    BankAccount{
```

```
public void deposit(int amount){
```



Type ของพารามิเตอร์ เป็นคนละอันกับของเดิม ดังนั้น
จาวาจะนึกว่าเขียนเมธอดขึ้นใหม่ ไม่ได้ **override** ของ
เก่า

คอนสตรัคเตอร์

```
public class CheckingAccount extends  
    BankAccount{
```

```
public CheckingAccount(double initialBalance){
```

```
    super(initialBalance);
```

```
    counter = 0 ;
```

```
}
```

เรียกโค้ดที่เขียนไว้ในคอนสตรัคเตอร์
ของซูเปอร์คลาส การเรียกคอนสตรัค
เตอร์ของซูเปอร์คลาสต้องเป็นบรรทัด
แรกเสมอ

ถ้าเราไม่เรียกคอนสตรัคเตอร์ของซูเปอร์คลาส จาวาจะเรียก **super()** ให้ แต่ถ้า
ในซูเปอร์คลาสเราเขียนแต่คอนสตรัคเตอร์ที่มีพารามิเตอร์ จาวาจะถือว่าไม่มีการ
เขียนคอนสตรัคเตอร์ที่ไม่มีพารามิเตอร์เอาไว้เลย และจะ **error**

การ convert type

```
SavingAccount a = new SavingAccount(10.0);
```

```
BankAccount anAccount = a; // อันนี้ทำได้โอเค
```

```
Object anObject = a; // นี่ก็ทำได้โอเค
```

```
anAccount.deposit(1000); // ok
```

```
anAccount.addInterest(); // ทำไม่ได้
```

จะมีคำถามคือ แล้วจะทำให้มันเรียกเมธอดได้น้อยลงทำไม

มันก็มีที่ต้องใช้ **type** ของตัวที่เรียกเมธอดได้น้อยกว่า เช่น

```
Public void transfer(double amount, BankAccount  
    other){  
    withdraw(amount);  
    other.deposit(amount);  
}
```

อันนี้ใช้ใน **transfer** ได้ ไง
เพราะ **transfer** มันควรจะใช้
กับบัญชีธนาคารแบบไหนก็ได้

....

```
CheckingAccount myAcc = new CheckingAccount(..);  
momAccount.transfer(1000,myAcc);
```


ถ้าต้องการเปลี่ยน **type** อีกข้าง

```
If (anObject instanceof BankAccount){  
    BankAccount anAccount = (BankAccount) anObject;  
  
}
```

ดูว่าเมธอดไหนถูกเรียก

```
BankAccount anAccount = new CheckingAccount();  
anAccount.deposit(1000);
```

ทั้ง **BankAccount** และ
CheckingAccount มีเมธอด **deposit**
ทั้งคู่ ตกลงมันเรียกอันไหน

สองขั้นตอน อย่าลืม