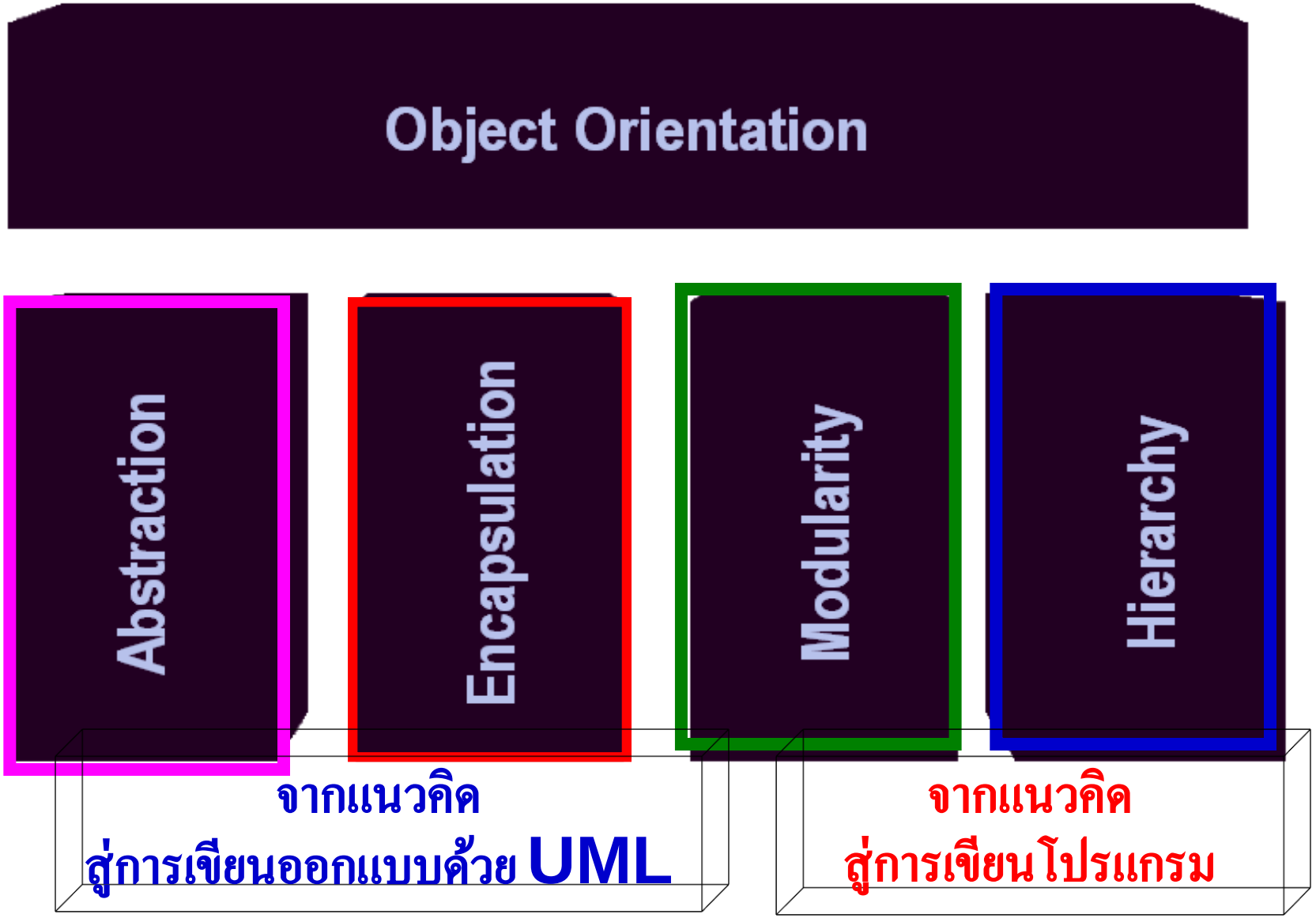
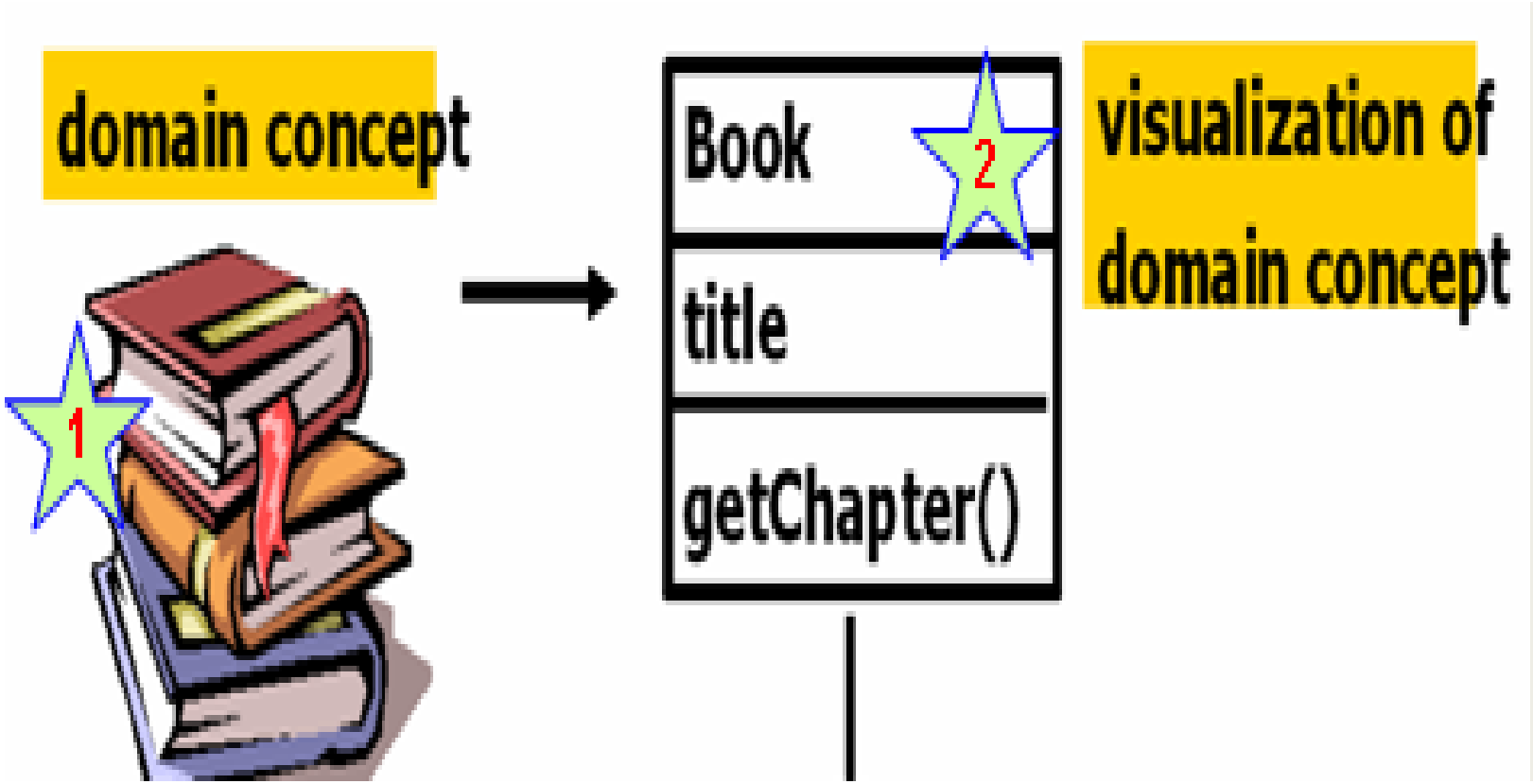
Basic Principles of Object Orientation



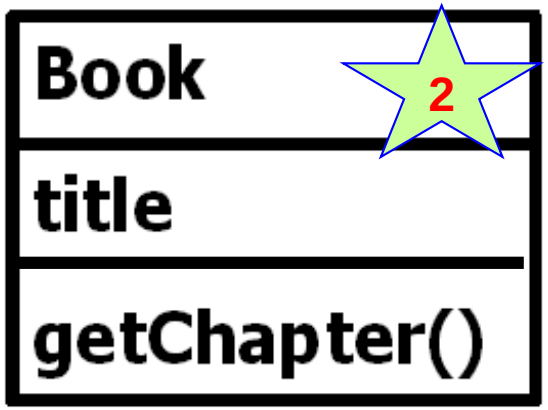
Abstraction

- คือกระบวนการในการให้**concept**กับ**object**จนทำให้เกิด**คลาส**



จากกระบวนการ Abstraction เข้าสู่การเขียนโปรแกรม

domain concept



visualization of domain concept

representation in an object-oriented programming language

3

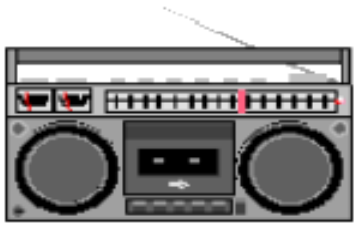
```
public class Book
{
   private String title;
   public Chapter getChapter(int) {...}
}
```

What is Abstraction?

การสั่งซื้อสินค้า

Salesperson

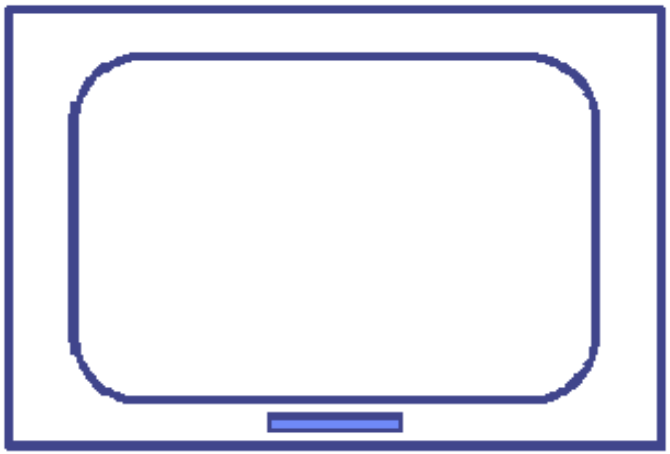
Customer

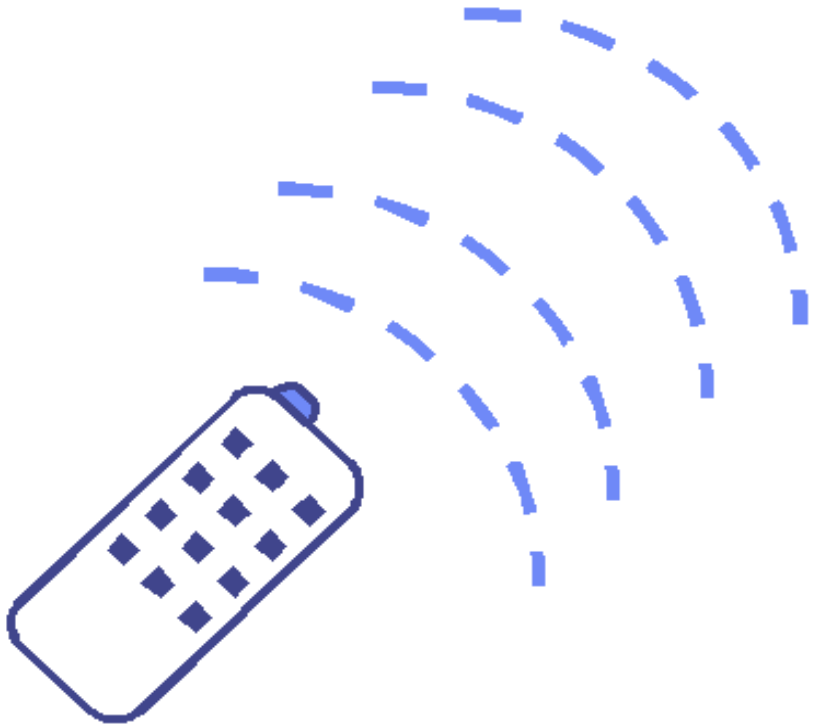


Product

What is Encapsulation?

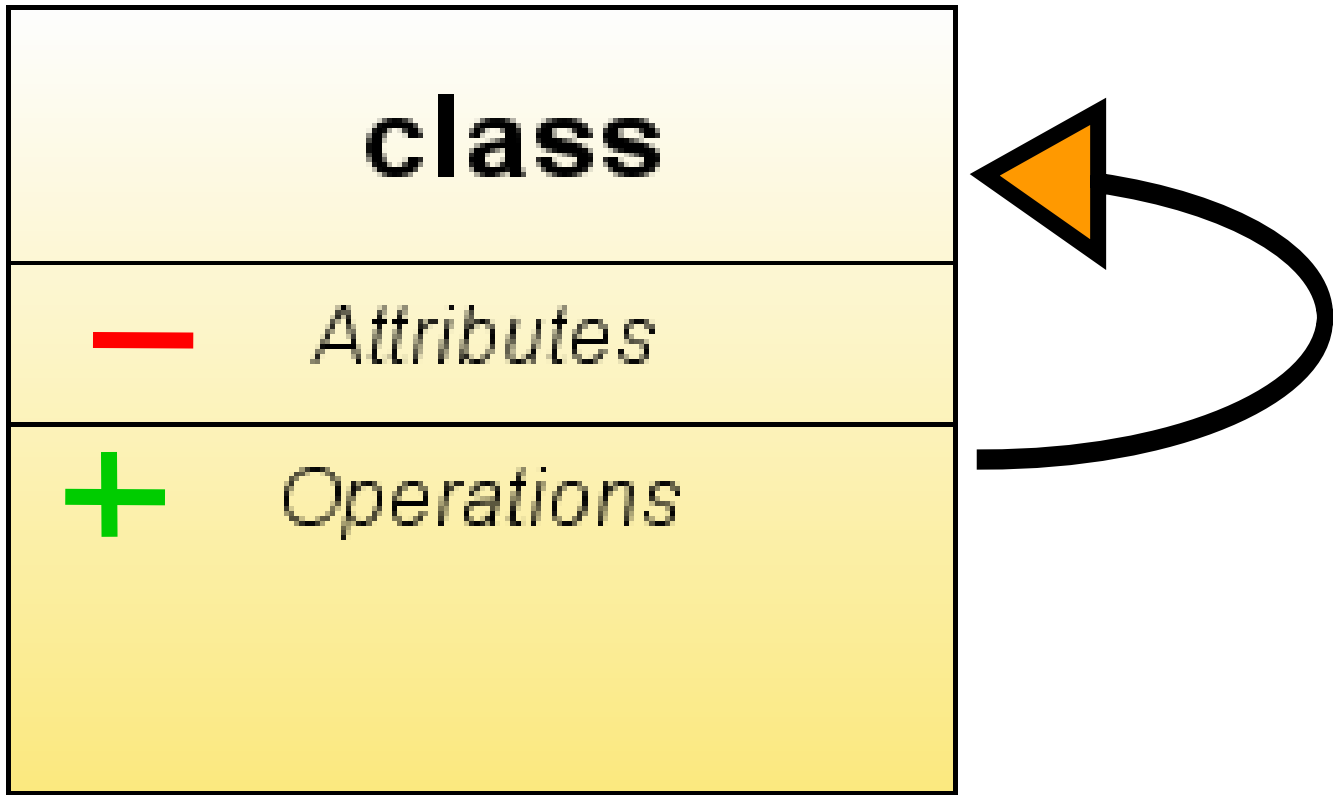
- Hide implementation from clients
  - Clients depend on interface





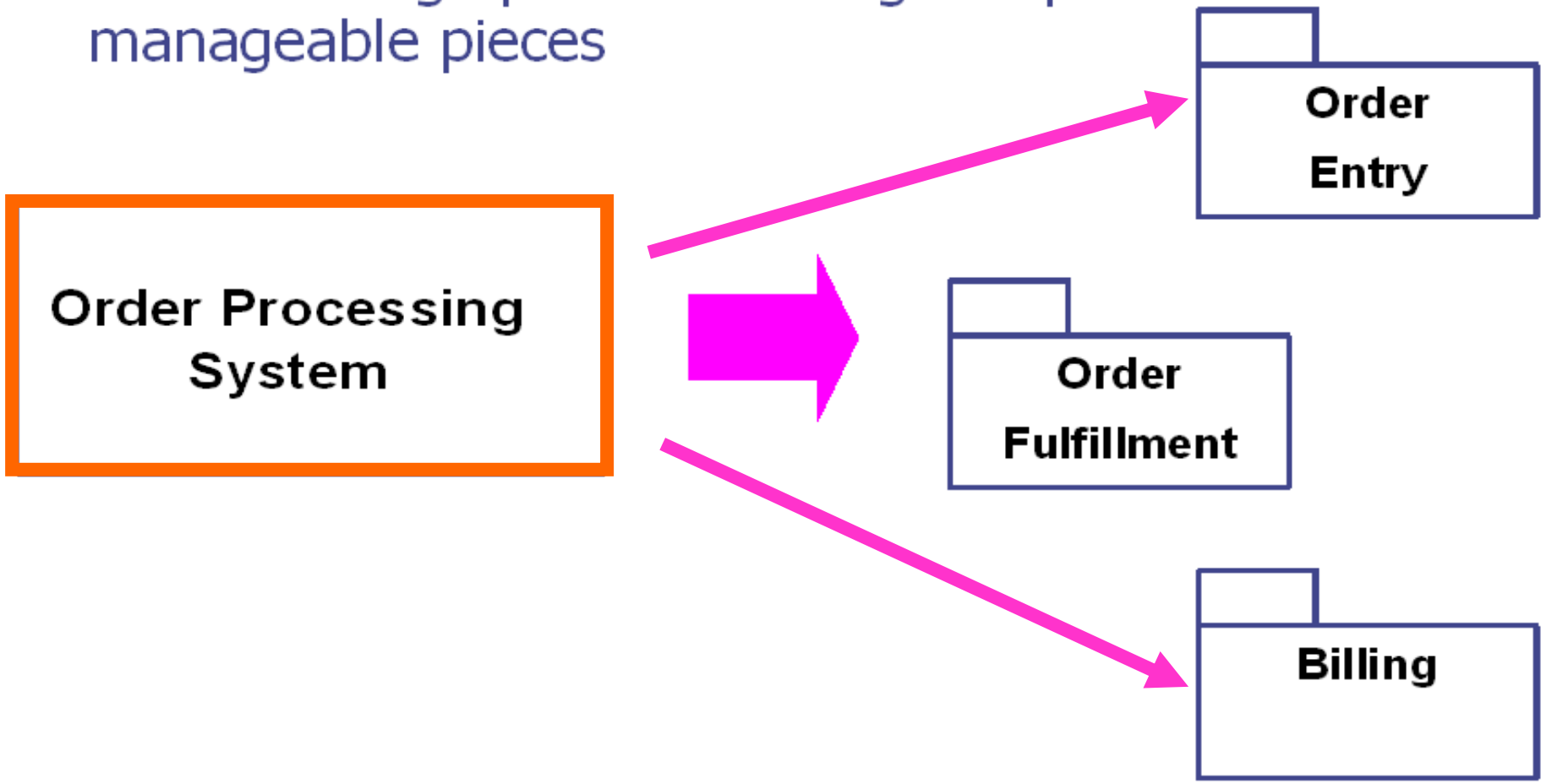
Encapsulation

คือการที่ **attribute** บางส่วนในคลาสได้ถูกปกปิดเอาไว้ เป็นการส่วนตัว การเข้าไปเปลี่ยนแปลงหรือดึงมาใช้งาน ทำได้โดยผ่าน **Operation** เท่านั้น



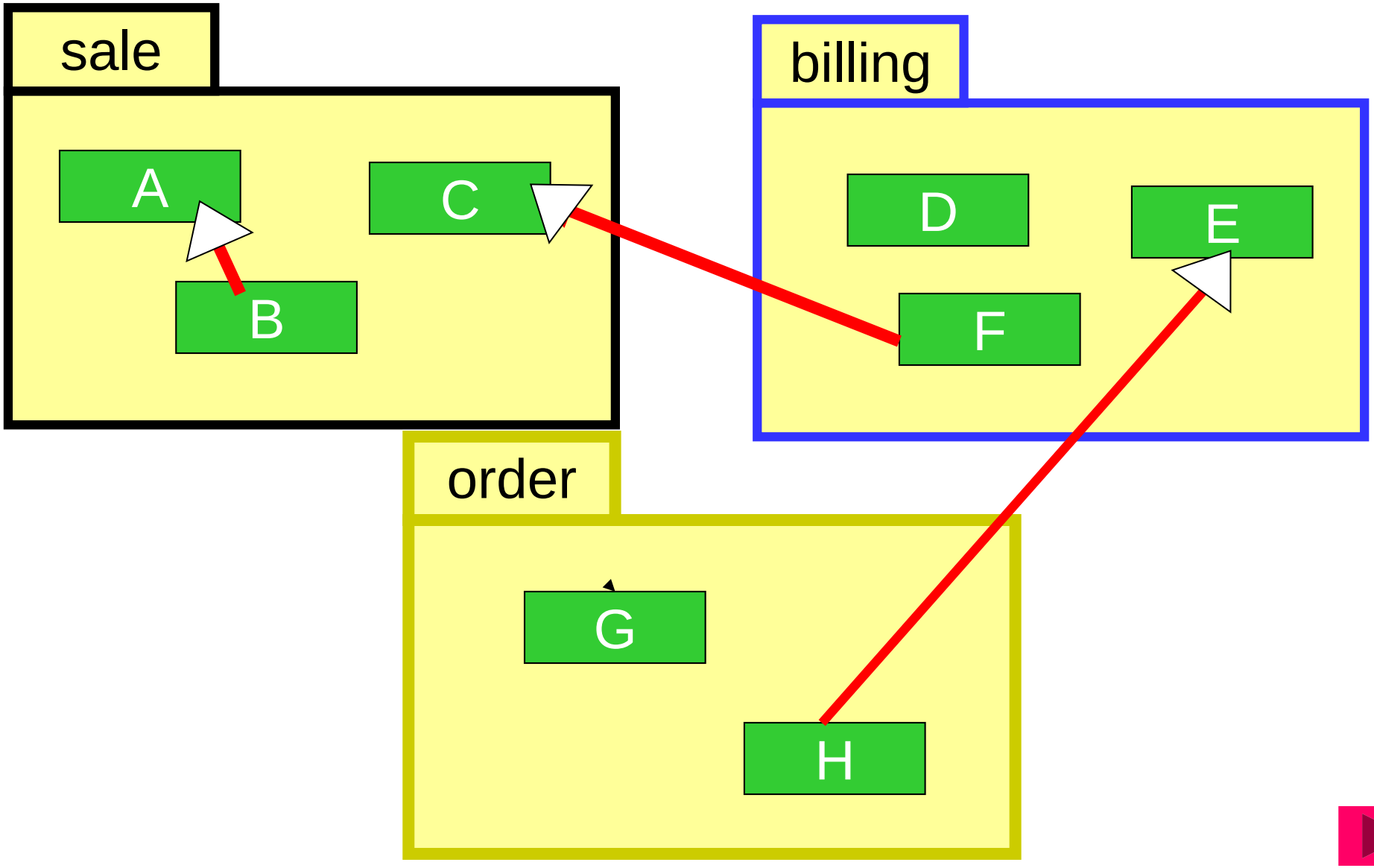
What is Modularity?

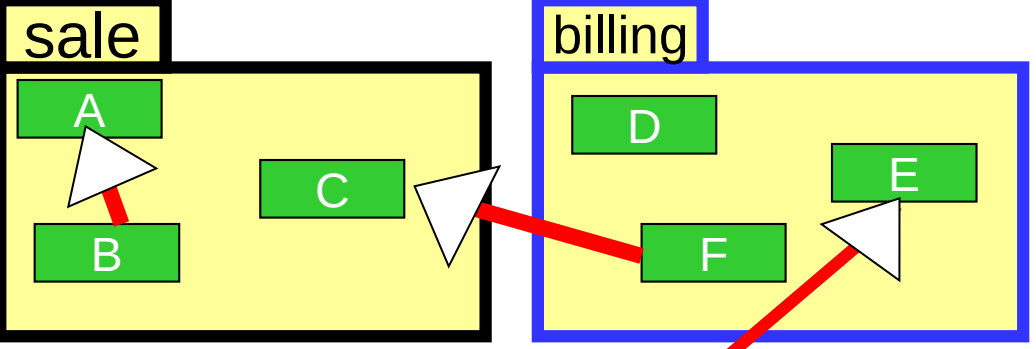
- The breaking up of something complex into manageable pieces



Package

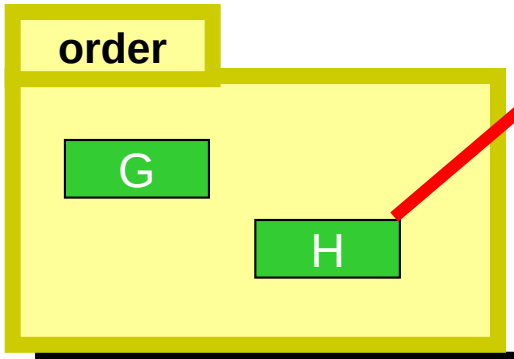
คลาสที่อยู่ Folder **เดียวกันแสดงว่าอยู่ใน** package เดียวกัน






```
1 package sale;
2 public class C {}
3
```

```
1 package billing;
2 import sale.C;
3
4 class F extends C{}
```




```
1 package billing;
2
3 class D {}
```

```
1 package billing;
2
3 class E {}
```

```
1 package sale;
2
3 class A{ }
```

```
1 package sale;
2
3 class B extends A{}
4
```

```
1 package sale;
2
3 class C {}
```

```
1 package billing;
2 import sale.C;
3
4 class F extends C{}
```

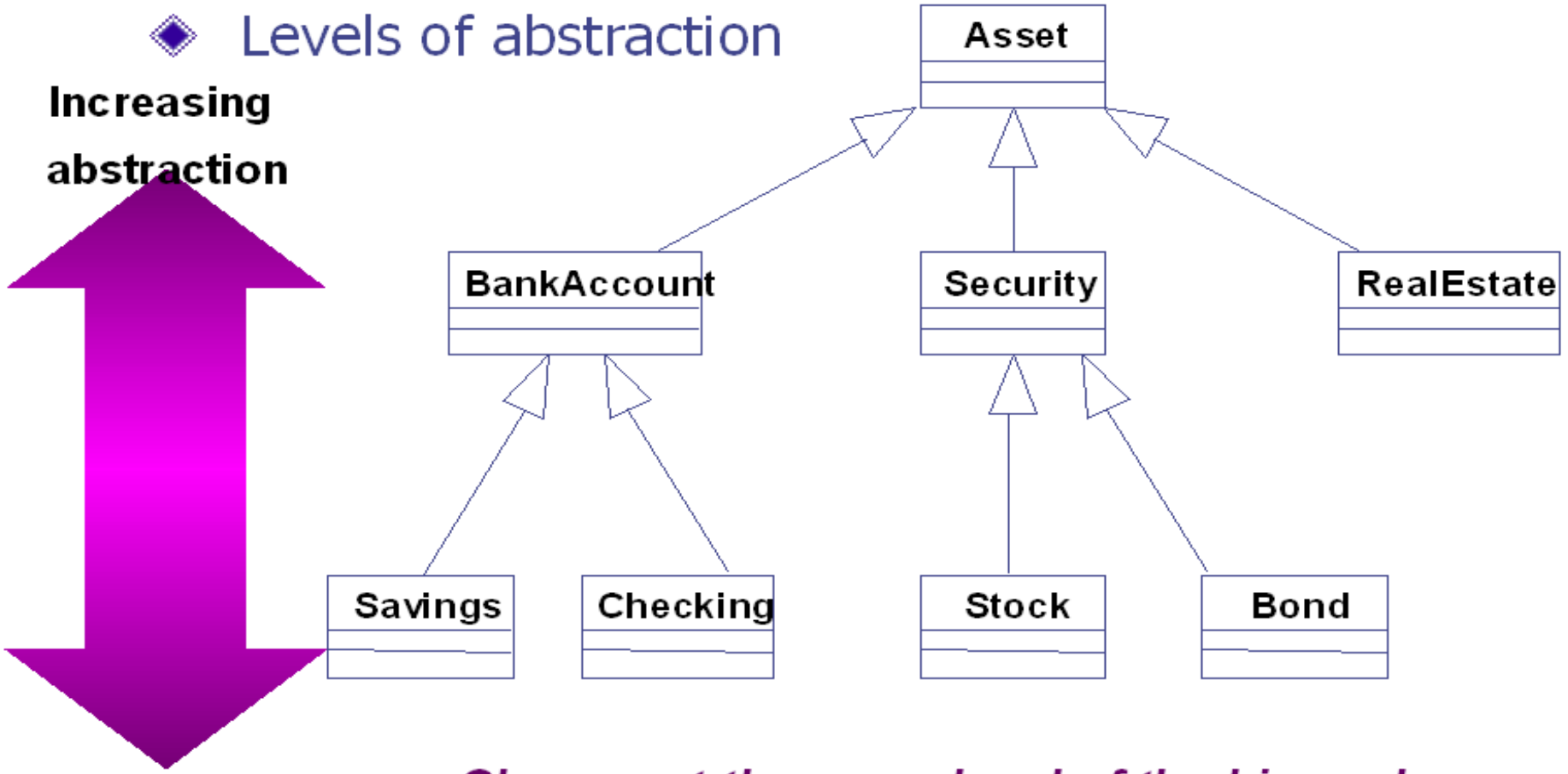
```
1 package sale;
2
3 class C {}
```

Error

```
A B C D E F
--------------------Configuration: j2sdk1.4.2_07 <Default>--------------------
H:\F.java:2: sale.C is not public in sale; cannot be accessed from outside package
```

What is Hierarchy?

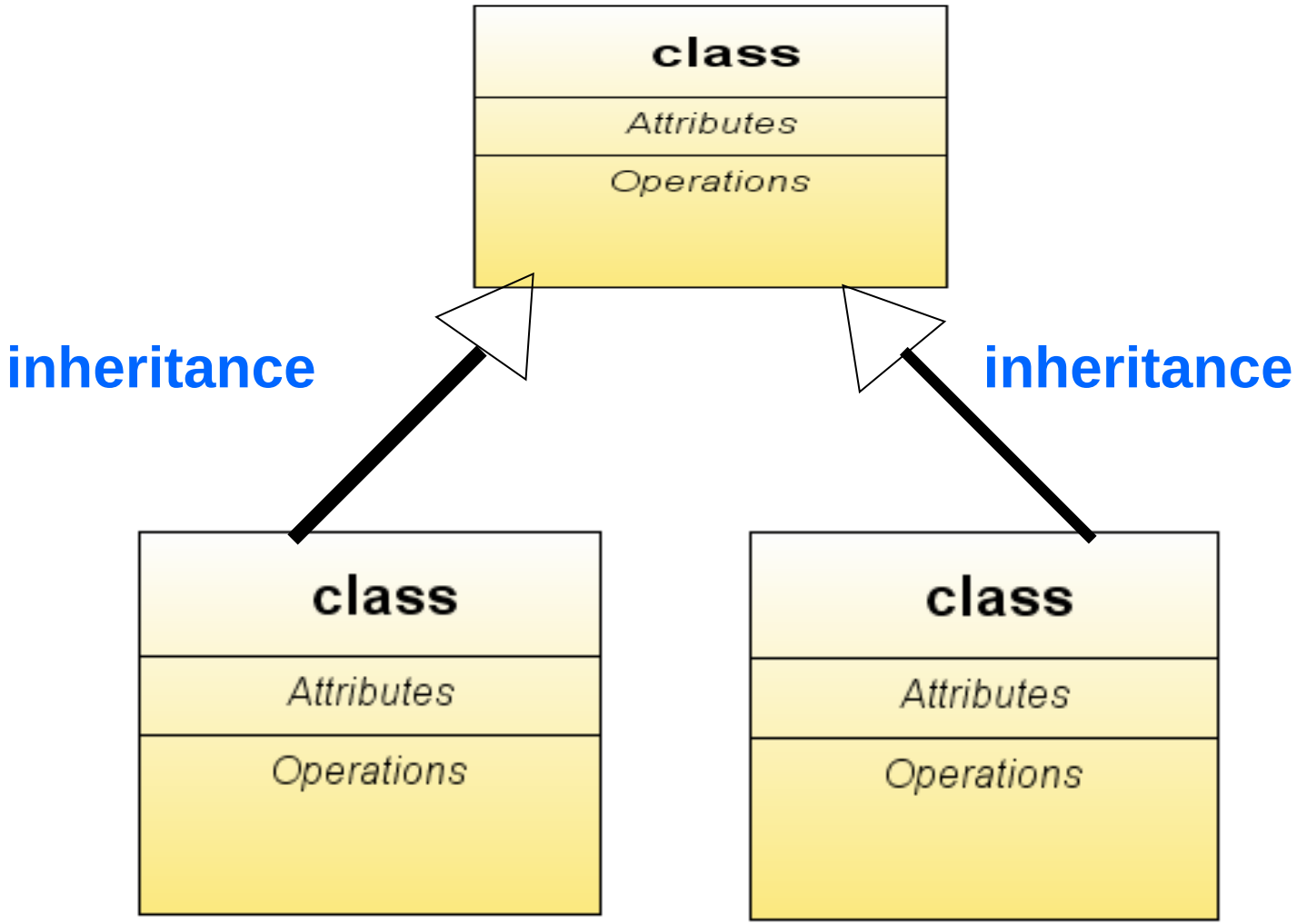
- Levels of abstraction



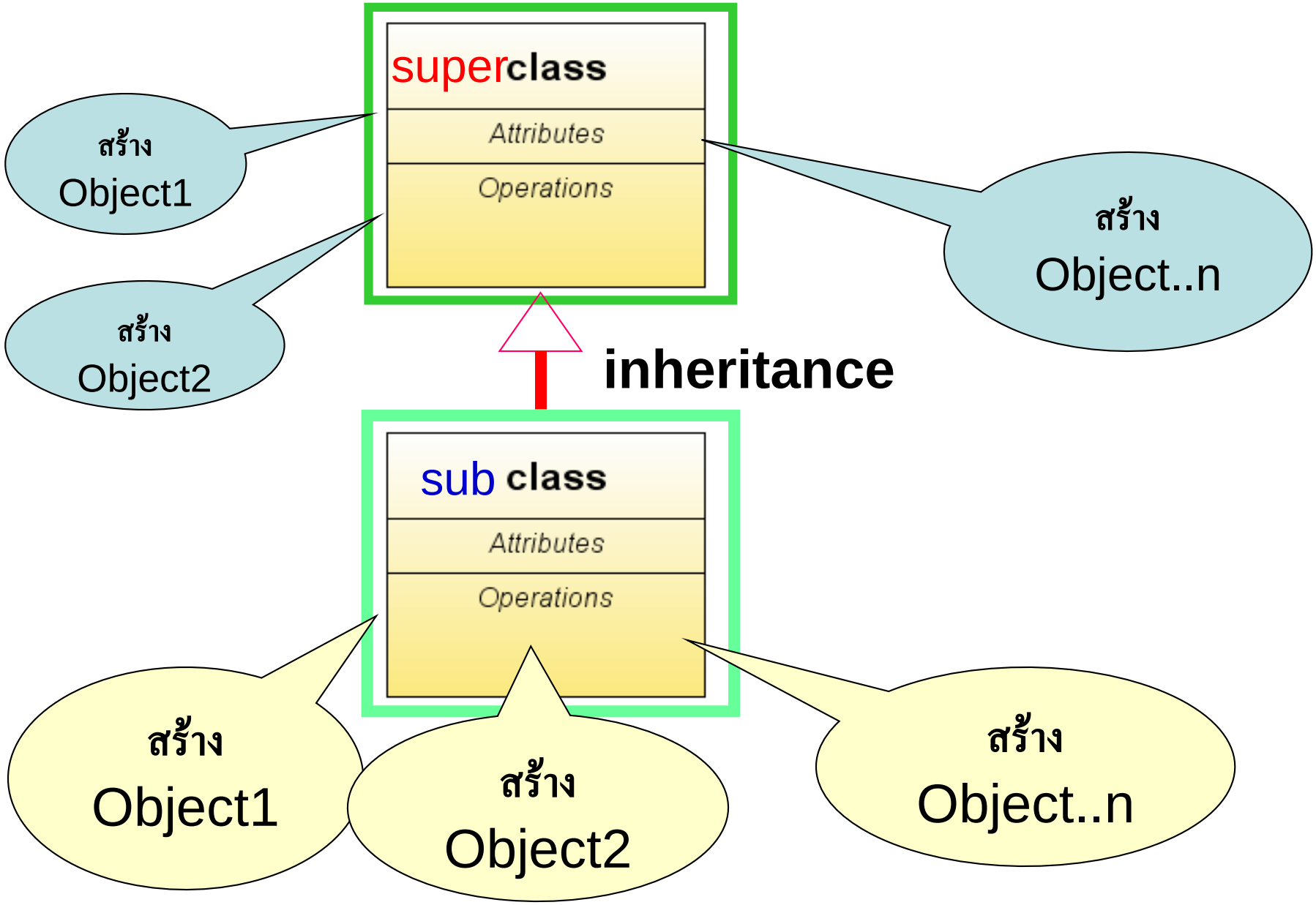
Classes at the same level of the hierarchy should be at the same level of abstraction

Inheritance

คือ subclass รับคุณสมบัติ (Attribute and Function)ทุกอย่าง มาจาก Superclass แล้วผนวกคุณสมบัติพิเศษ เพิ่มเข้าไปในแต่ละ subclass

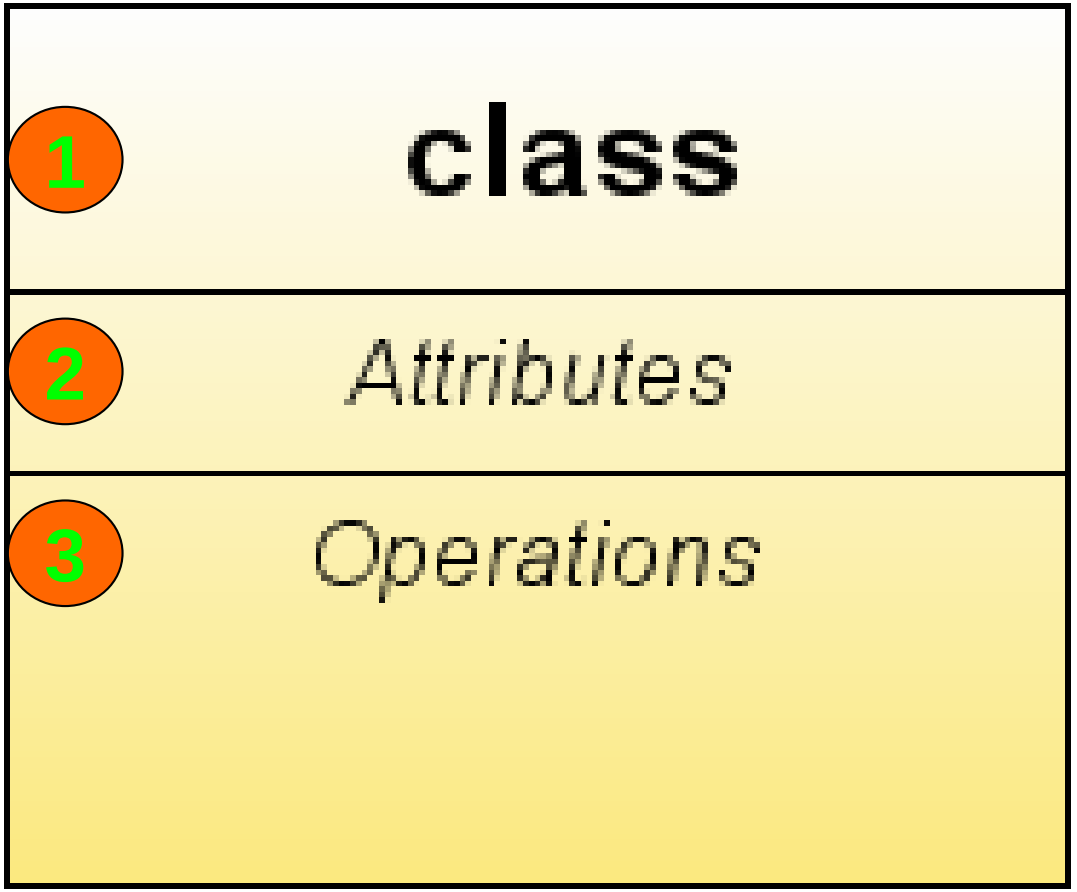


Concrete class คือคลาสที่สามารถสร้าง object



จากแนวคิด
สู่การเขียนออกแบบด้วย
UML

UML
class diagram



- Problem domain?
- Concept
- abstraction

ใส่ชื่อ class

Student
Attributes
Operations

constructor method

Constructor

Student
Attributes
Operations public Student ( )

ใส่ attribute, method

Student
Attributes private String id
Operations public Student ( ) public void setID(String pid) public String getID( )

ใส่ชื่อ attribute,method

Student
Attributes private String id private String name
Operations public Student ( ) public void setID(String pid) public String getID( ) public void setName(String pname) public String getName( )

ใส่ชื่อ attribute,method

Student
Attributes private String id private String name private double gpa
Operations public Student ( ) public void setID(String pid) public String getID( ) public void setName(String pname) public String getName( ) public void setGPA(double pgpa) public double getGPA( )

ใส่ชื่อmethod

Student

Attributes

private String id
private String name
private double gpa

Operations

public Student ( )
public void setID(String pid)
public String getID( )
public void setName(String pname)
public String getName( )
public void setGPA(double pgpa)
public double getGPA( )
public void showDetail( )

เครื่องหมาย Attribute and Operation

"+" for *public*

"#" for *protected*

"-" for *private*

"~" for *package*

โครงสร้างคลาส

```
1 class Student{
2
3 }
4
```

ใส่ Constructor method

```
1 class Student{
2     public Student (){
3
4     }
5
6 }
```

ใส่ data fields

```
 1 class Student{
 2       private String id;
 3       private String name;
 4       private double gpa;
 5
 6       public Student (){
 7
 8        }
 9
10 }
```

ใส่ method

```
 1 class Student{
 2       private String id;
 3       private String name;
 4       private double gpa;
 5
 6       public Student (){
 7
 8       }
 9
10       public void setID(String pid){
11          id = pid;
12       }
13
14 }
```

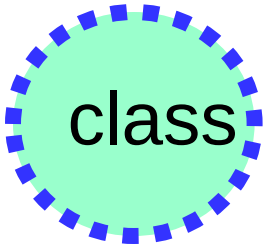
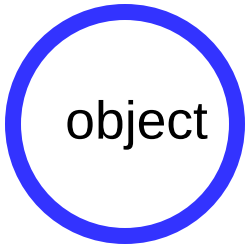
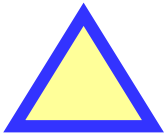
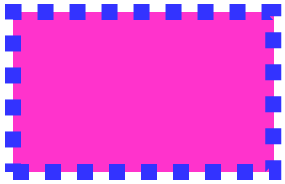
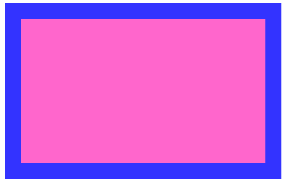
จากแนวคิด

สู่การเขียนโปรแกรม

ด้วยภาษา JAVA

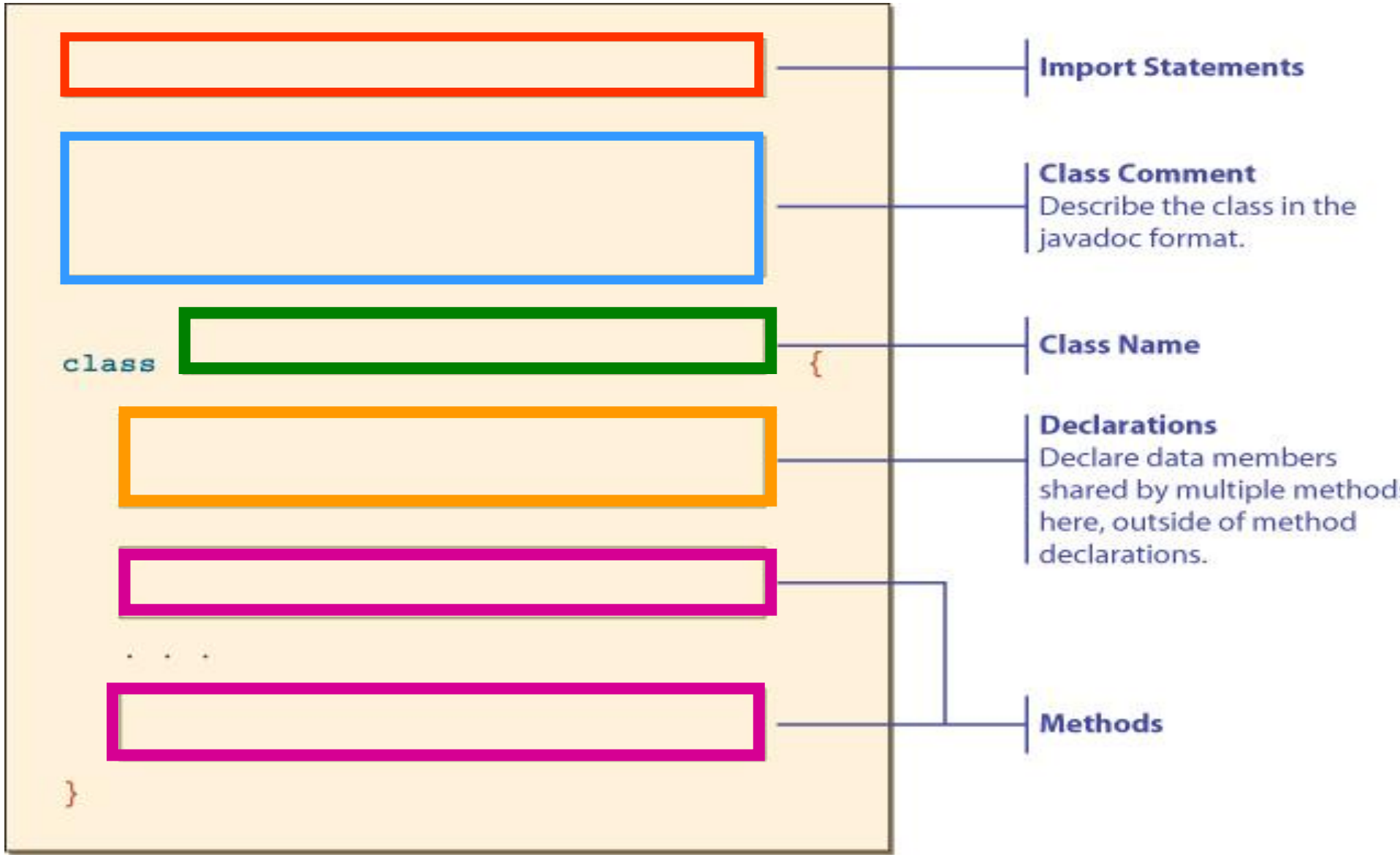
ข้อตกลง

สัญลักษณ์สำหรับแนวคิดในการทำความเข้าใจกับการเขียนโปรแกรมแบบOOP

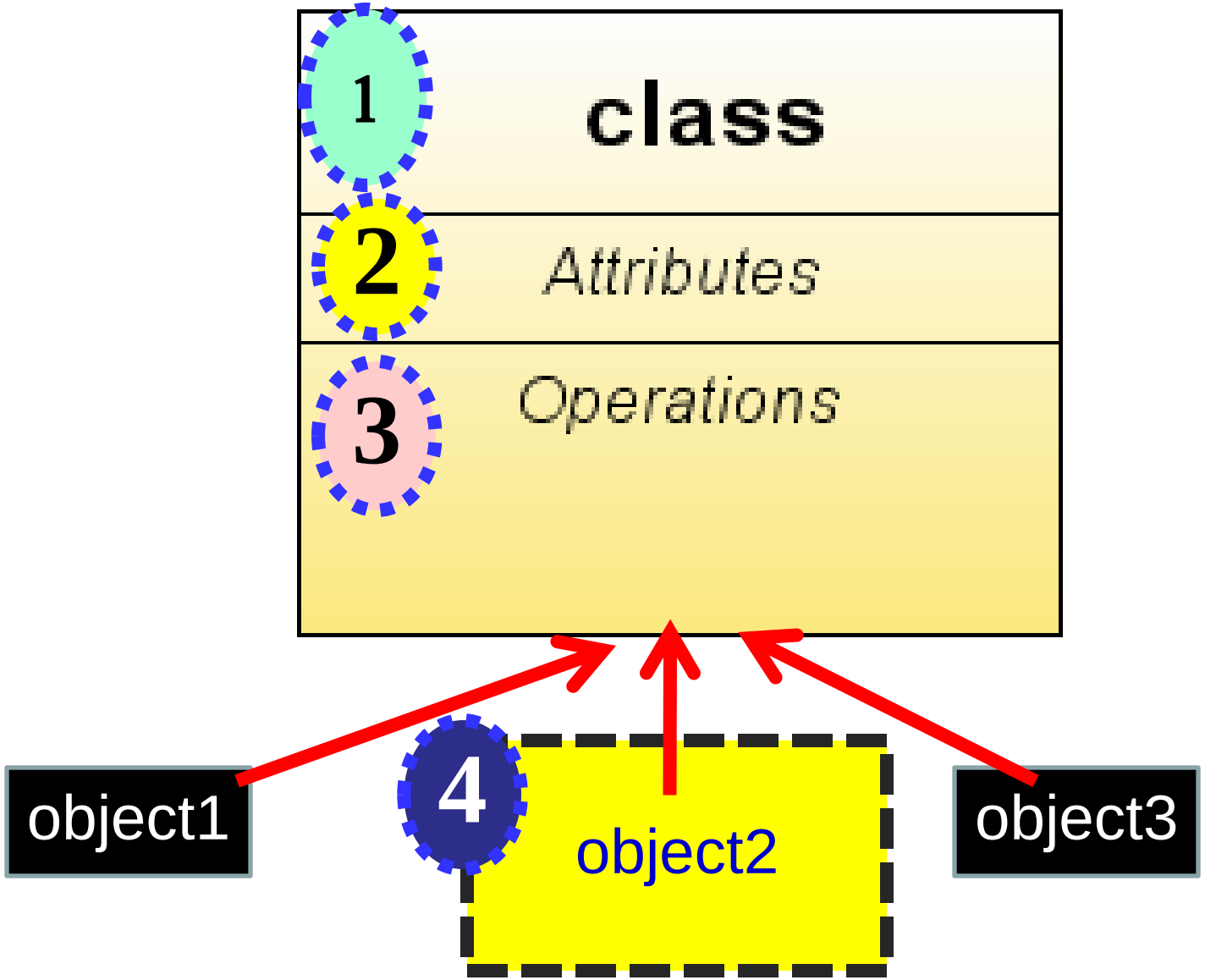
เป็นแม่แบบใช้งานจริงไม่ได้	พร้อมใช้งานในหน่วยความจำ	
		แสดงถึง class→object
		แสดงถึง attribute ที่อยู่ภายใน object
		แสดงถึง **method** ที่อยู่ภายใน **object**

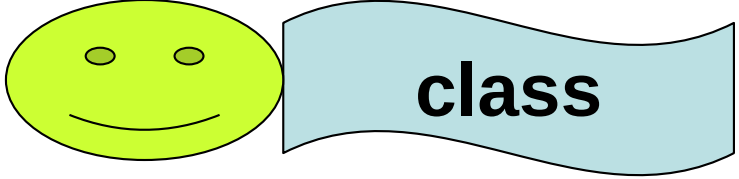
คำว่าพร้อมใช้งานในทางคอมพิวเตอร์สิ่งนั้นถูกสร้างและโหลดเข้าสู่หน่วยความจำและพร้อมสำหรับประมวลผล

A program template for a class definition.



Java Syntax


1
class
2
Attributes
3
Operations
4
object1
object2
object3



ประกอบด้วย

Access modifier

- public
- *default*
- abstract
- final

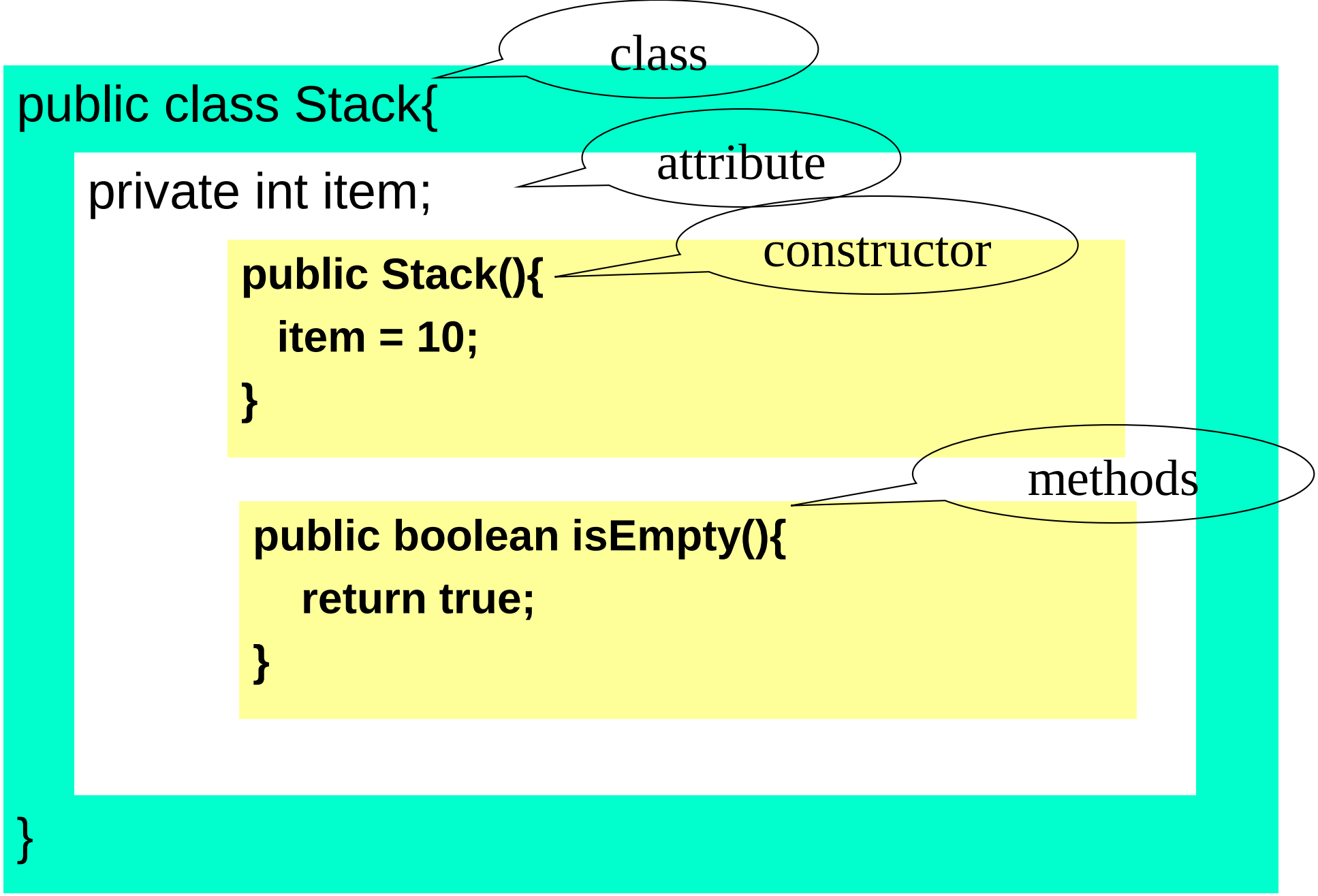
class

```
Class_Name{
//attributes
//methods

}
```

หมายเหตุ **Class_name** คือชื่อคลาส

class เป็นคีย์เวิร์ดสำหรับกำหนดคลาส

Access modifier ระดับของการเข้าถึงคลาส


class
public class Stack{
attribute
private int item;
constructor
public Stack(){
item = 10;
}
methods
public boolean isEmpty(){
return true;
}
}

Class Declaration Elements

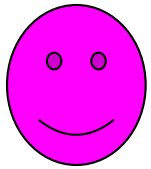
Element	Function
public	(Optional) Class is publicly accessible
abstract	(Optional) Class cannot be instantiated
final	(Optional) Class cannot be subclassed
class *NameOfClass*	Name of the class
extends *Super*	(Optional) Superclass of the class
implements *Interfaces*	(Optional) Interfaces implemented by the class

{
ClassBody
}
	Provides the class's functionality

ตัวอย่างการเขียนโครงร่างของ class

```
1 public class Student{
2 }
```

```
1 class Student{
2 }
```



```
1 public abstract class Student{
2
3 }
```

```
1 public final class Student{
2
3 }
```

ไวยากรณ์การประกาศ Data_member (attribute)

ประกอบด้วย

Access modifier

- public
- protected
- private
- *default*
- **final**
- **static**
- **transient**

Data_Type

- **-primitive**
- **-Reference**

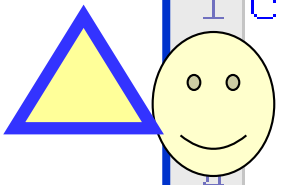
Data_name;

หมายเหตุ **Data_name** คือชื่อตัวแปร

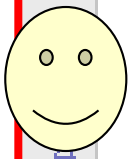
Data_Type คือประเภทของข้อมูล

Access modifier ระดับของการเข้าถึงคลาส

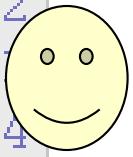
public
protected
private
default
final
static
transient



```
1 class Student{
      private String id;
      private String name;
      private double gpa;
5 }
```



```
1 class Student{
                String id;
    protected   String name;
    public      double gpa;
5 }
```



```
1 class Student{
2     private final String ID="99";
      private static  String name;
4     private   double gpa;
5 }
```

Variable Declaration Elements

Element	Function
accessLevel	(Optional) Access level for the variable
static	(Optional) Declares a class variable
final	(Optional) Indicates that the variable is a constant
transient	(Optional) Indicates that the variable is transient
volatile	(Optional) Indicates that the variable is volatile
type name	The type and name of the variable

Access Levels

Specifier	Class	Package	Subclass	World
private	Y	N	N	N
no specifier	Y	Y	N	N
protected	Y	Y	Y	N
public	Y	Y	Y	Y

3

ไวยากรณ์ในการสร้าง method_member

ประกอบด้วย

Access modifier

- public
- protected
- private
- final
- static
- abstract
- synchronized
- Strictfp
- native

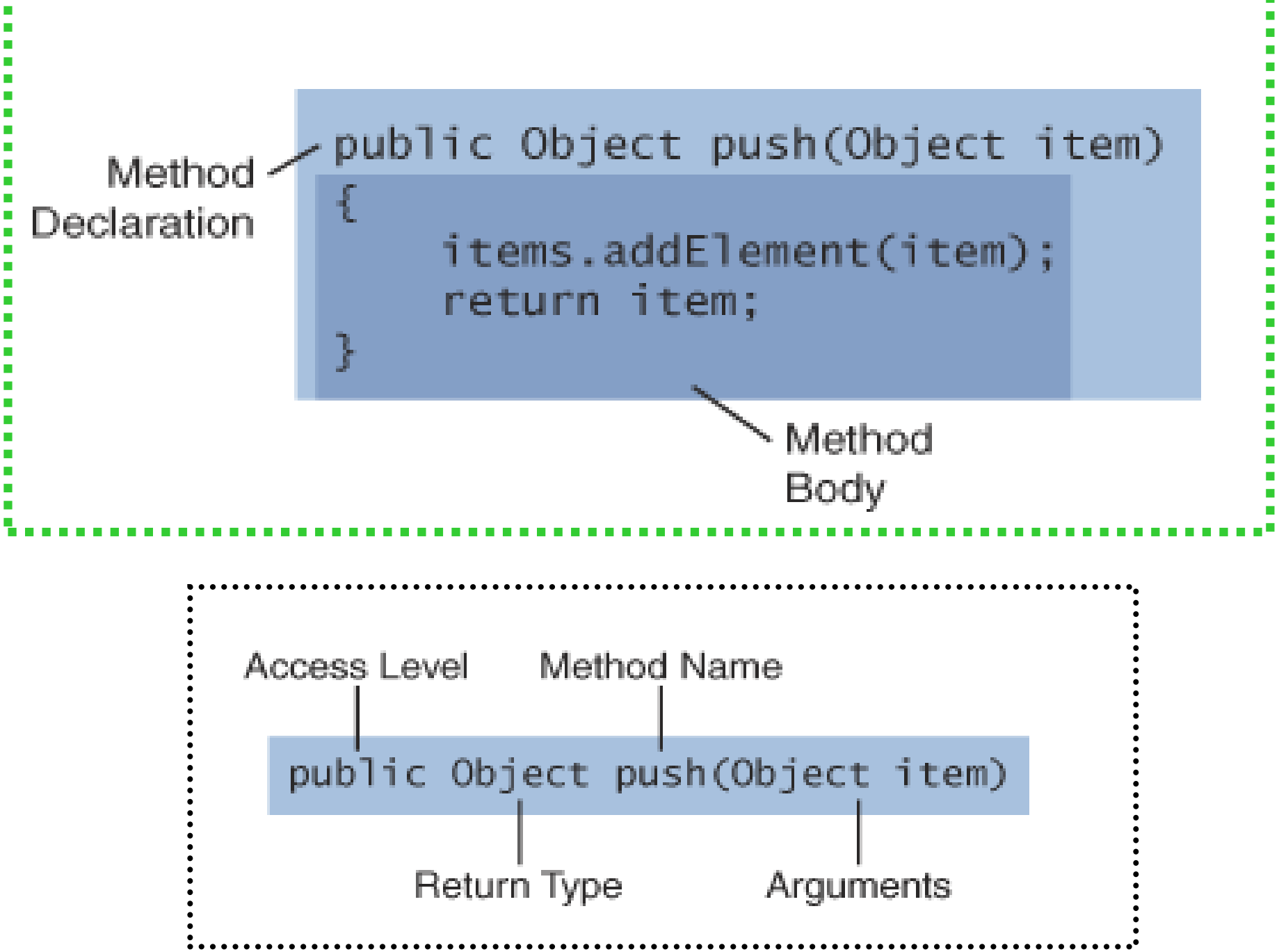
Return_data_type

- -primitive
- -Reference
- -void

method_name (Argument list){

statement;

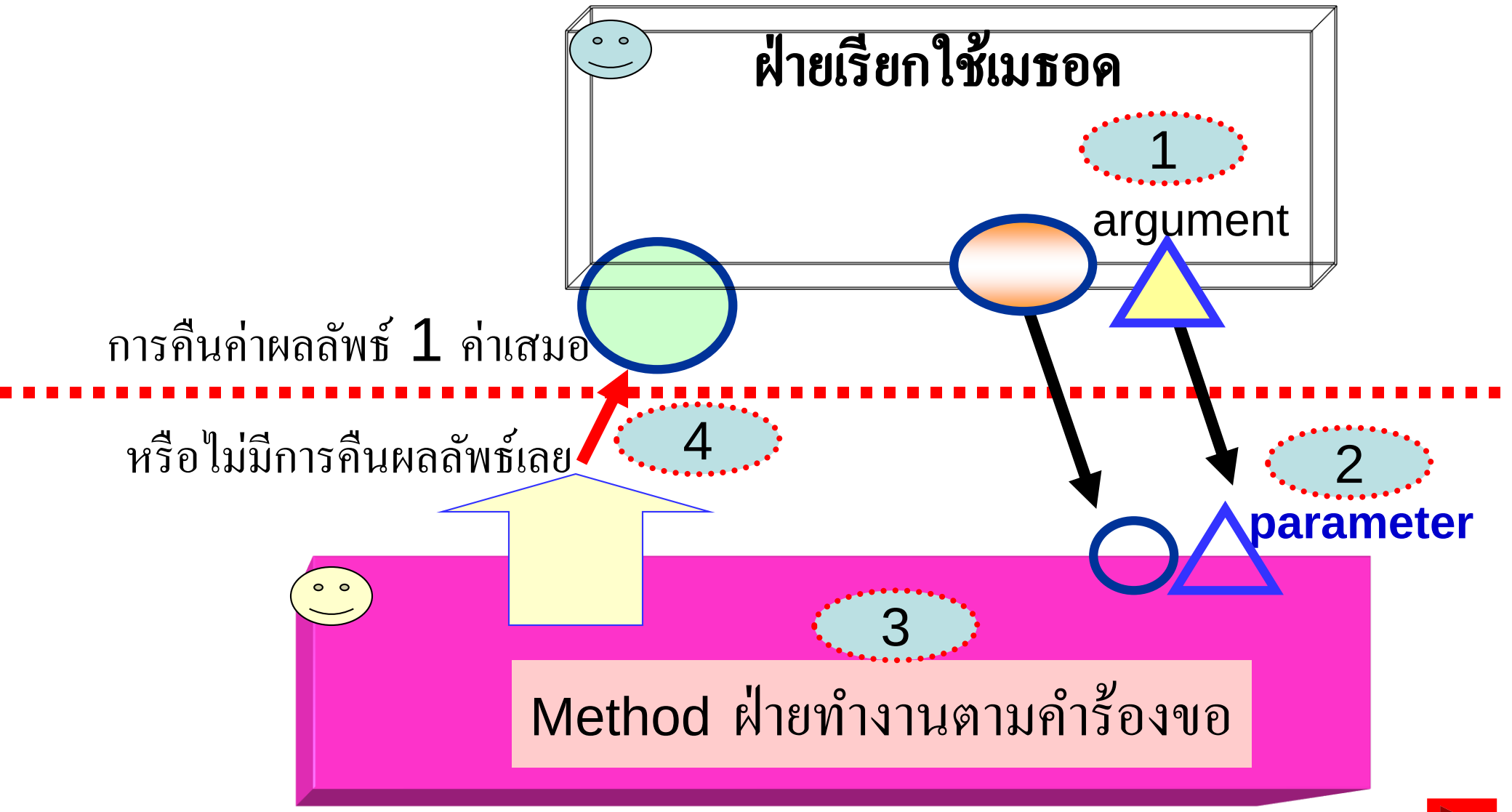
}


Method
Declaration
public Object push(Object item)
{
items.addElement(item);
return item;
}
Method
Body

Access Level
Method Name
public Object push(Object item)
Return Type
Arguments

แนวคิดเกี่ยวกับ Parameter Variable

ตัวแปรที่ใช้ส่งผ่านข้อมูลที่มีการเรียกใช้เมธอดกันเพื่อส่งข่าวสาร



public
protected
private
final
static

```
public void setID(String pid){
   id = pid;
}

public String getID(){
   return id;
}
```

```
public static void main (String args[]){
   char name = 'A';
   char uniname = '\u0041';
   System.out.println(name+"\n"+uniname);
}
```

```
1 class Student{
2       private int x;
3
4       protected int doStuff(){
5          return x;
6       }
7 }
```

Constructor

เป็นเมธอด ประเภทหนึ่งที่อยู่ภายในคลาส และ constructor จะถูกประมวลผลอัตโนมัติเมื่อมีการสร้าง instance ใด ๆ

จากคลาสนั้น ๆ ต้องตั้งเหมือนชื่อคลาสทุกประการและไม่ระบุการคืนค่าใด ๆ ไม่ว่าจะเป็น void ,int ,float อื่น ๆ

```
1  class Student{
2     private String id;
3     private String name;
4
5     public Student(){} //constructor 1
6
7     public Student(String idp) {
8        id = idp;
9     }
10
11    public Student(String idp,String namep){
12       id = idp;
13       name = namep;
14     }
15 }
```

Overloading

คือเมธอดที่ชื่อเหมือนกันแต่ *parameter* ต่างกัน

```
1  class Student{
2    private String id;
3    private String name;
4
5    public Student(){} //constructor 1
6
7    public Student(String idp) {
8      id = idp;
9    }
10
11   public Student(String idp,String namep)
12     id = idp;
13     name = namep;
14    }
15
16    public void showDetails(){
17      System.out.println("id = "+id);
18    }
19
20    public void showDetails(int n){
21      System.out.println("id = "+id);
22      System.out.println("name = "+name);
23      System.out.println("parameter= "+n);
24    }
25  }
```

```
1  public class ObjectS{
2    public static void main(String args[]){
3
4      Student top=new Student("4804249125","Krengsak");
5
6      top.showDetails();
7
8      top.showDetails(5);
9    }
10 }
```

Student | ObjectS

```
--------------------Configuration: j2sdk1.4.2_07 <Default>--------------------
id = 4804249125
id = 4804249125
name = Krengsak
parameter= 5
```

Method Declaration Elements

Element	Function
accessLevel	(Optional) Access level for the method
static	(Optional) Declares a class method
abstract	(Optional) Indicates that the method is not implemented
final	(Optional) Indicates that the method cannot be overridden
native	(Optional) Indicates that the method is implemented in another language
synchronized	(Optional) The method requires a monitor to run
returnType methodName	The method's return type and name
(*paramList*)	The list of arguments to the method
throws exceptions	(Optional) The exceptions thrown by the method

Access Levels

Specifier	Class	Package	Subclass	World
private	Y	N	N	N
no specifier	Y	Y	N	N
protected	Y	Y	Y	N
public	Y	Y	Y	Y

การสร้าง object จาก class

1.การประกาศชื่อตัวแปรอ้างถึง(Reference Variable)

```
Class_name  data_name;
```

2.ขั้นตอนการสร้าง Instance ของ class (การจองพื้นที่ในหน่วยความจำ)

```
data_name = new Class_Constructor(Argument_List)
```

คำสั่งย่อ

```
Class_name data_name = new Class_Constructor (Argument_List)
```

ตัวอย่างการสร้าง object ด้วย java

1.การประกาศชื่อตัวแปรอ้างถึง(Reference Variable)

```
Double  d;
```

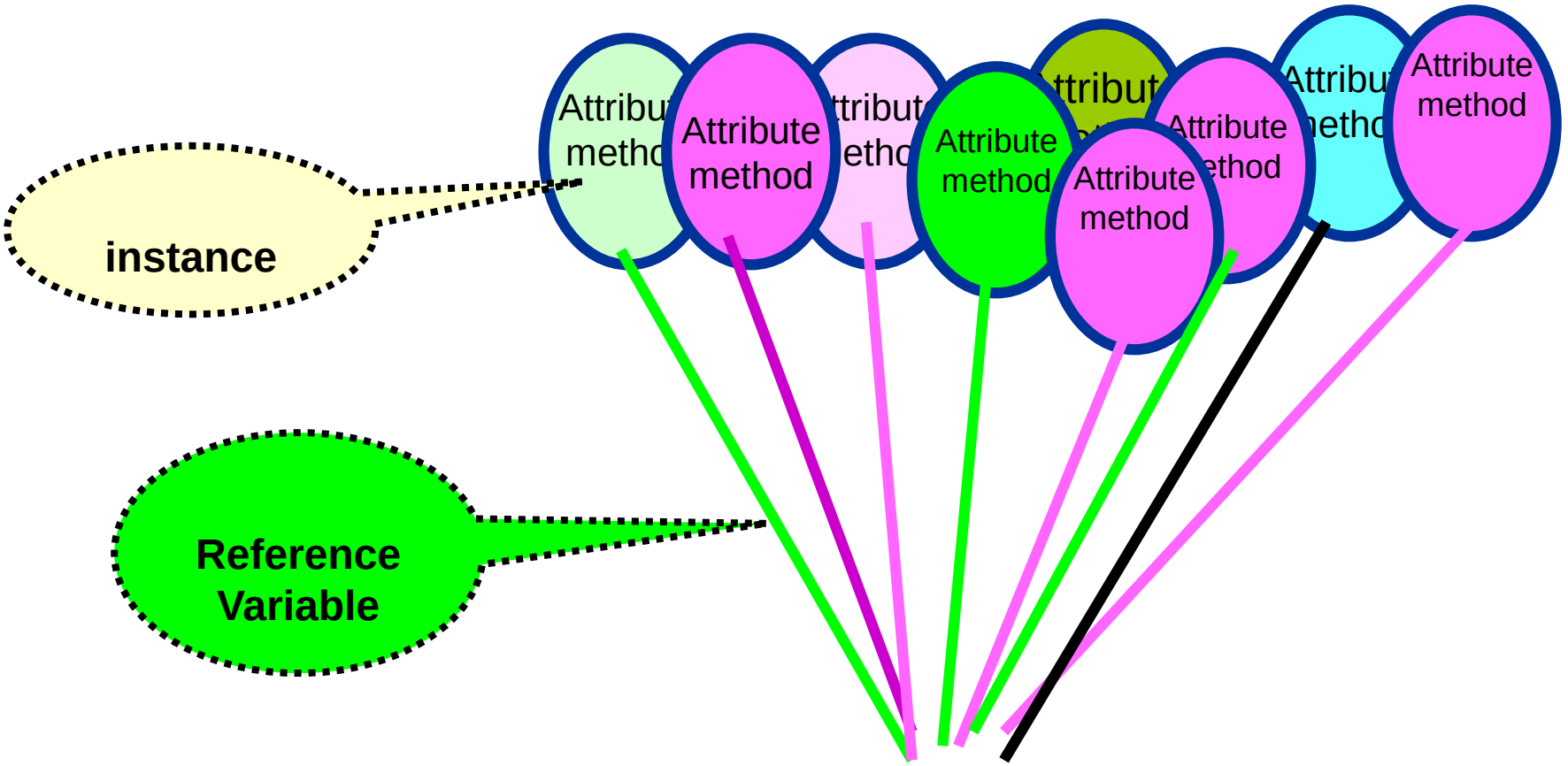
2.ขั้นตอนการสร้าง Instance ของ class (การจองพื้นที่ในหน่วยความจำ)

```
d = new Double(3.5);
```

คำสั่งย่อ

```
Double d1 = new Double(5.05);
```

Concept of Instance of Class and Reference Variable

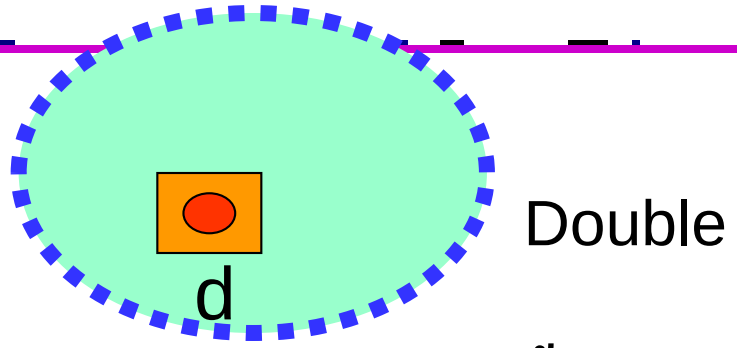


Reference Variable กำหนดขึ้นเพื่อเป็นประโยชน์ในการอ้างถึง ไปยังตำแหน่งของ Instance ที่อยู่ในหน่วยความจำ

ภาพจินตนาการของการสร้าง object

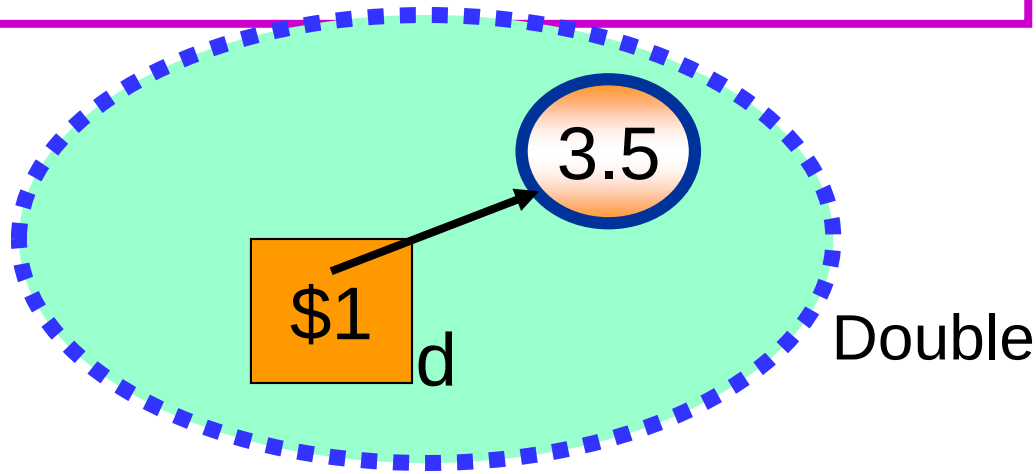
1.การประกาศชื่อตัวแปรอ้างถึง(Reference Variable)

```
Double d;
```



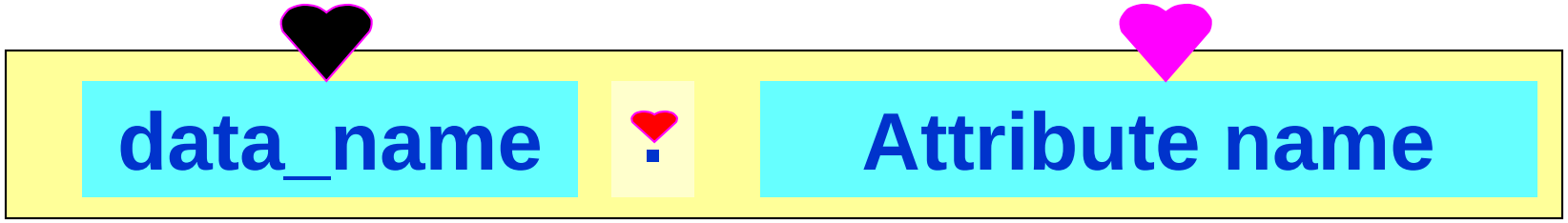
2.ขั้นตอนการสร้าง Instance ของ class (การจองพื้นที่ในหน่วยความจำ)

```
d = new Double(3.5);
```



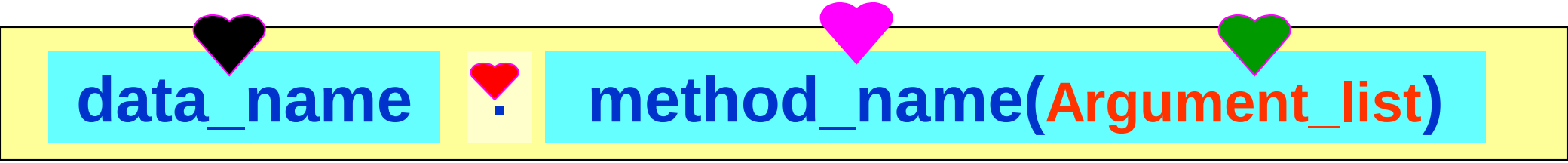
การเรียกใช้สมาชิกของ object

1. เรียกใช้ Attribute



ชื่อของอ็อบเจ็กจุดและชื่อของคุณสมบัติ

2. เรียกใช้ Method



ชื่อของอ็อบเจ็กจุดและชื่อเมธอด

1. เรียกใช้ Attribute

```
1  public class Student{
2        static String id;
3
4       public void setID(String ID){
5            id = ID;
6       }
7       public String getID( ){
8            return id;
9       }
10 }
```

data_name . **Attribute name**

```
1 class CreateObject{
2     public static void main(String args[]){
3        Student s1 = new Student();
4        s1.id="480011";
5        System.out.println(s1.id);
6  }
7 }
```

480011

2. เรียกใช้ Method

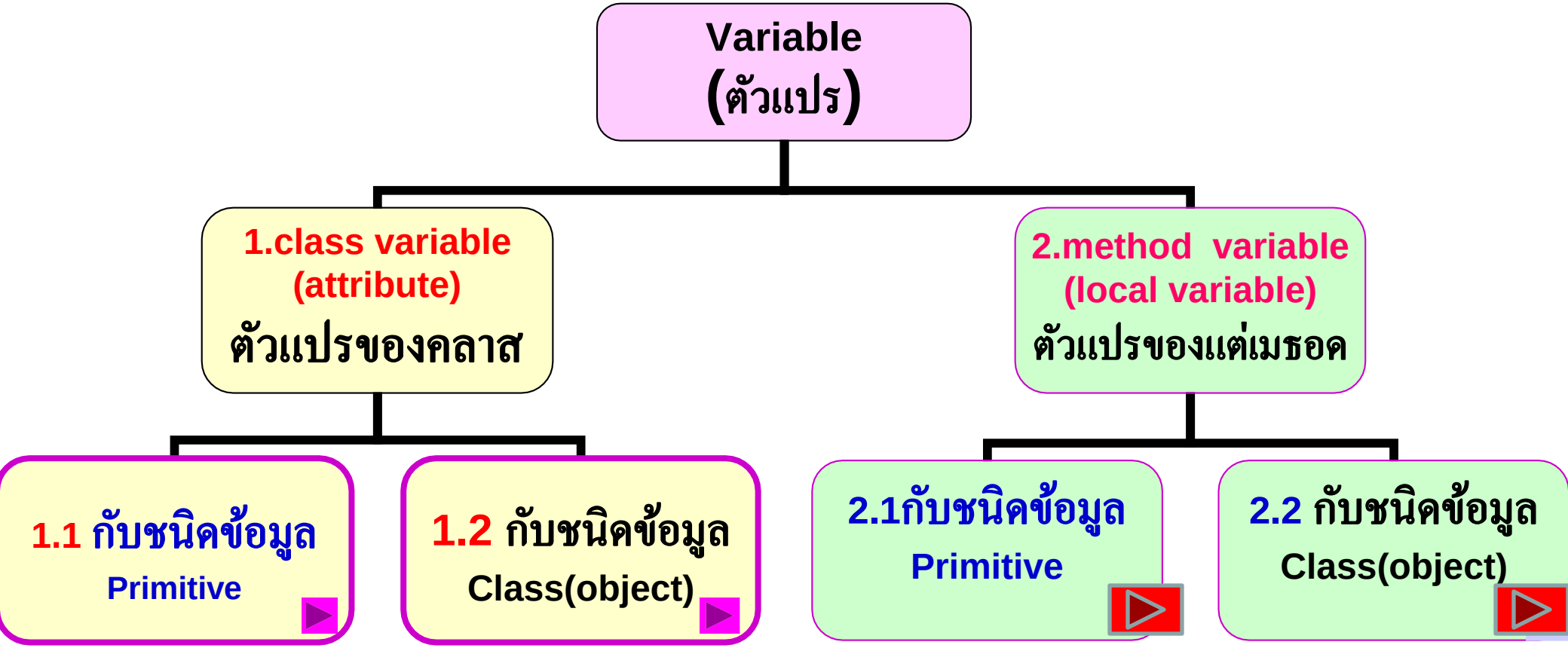
```
 1 public class Student{
 2     private String id;
 3
 4       public void setID(String ID){
 5             id = ID;
 6       }
 7       public String getID( ){
 8             return id;
 9       }
10 }
```

data_name . **method_name(Argument_list)**

```
1 class CreateObject{
2     public static void main(String args[]){
3       Student s1 = new Student();
4       s1.setID("48011");
5       System.out.println(s1.getID());
6  }
7 }
```

480011

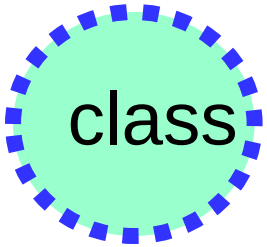
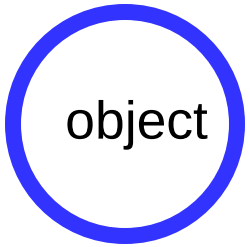
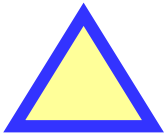
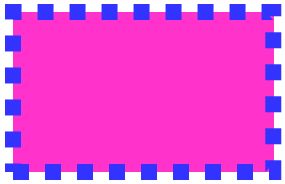
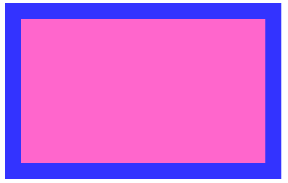
ประเภทตัวแปรในภาษาจาวา



ตัวแปร หมายถึงชื่อของพื้นที่หน่วยความจำที่ถูกตั้งขึ้นโดยโปรแกรมเมอร์ เพื่อความสะดวกในในการอ้างถึงข้อมูลในตำแหน่งนั้นๆ

ข้อตกลง

สัญลักษณ์สำหรับแนวคิดในการทำความเข้าใจกับการเขียนโปรแกรมแบบOOP

เป็นแม่แบบใช้งานจริงไม่ได้	พร้อมใช้งานในหน่วยความจำ	
		แสดงถึง class→object
		แสดงถึง attribute ที่อยู่ภายใน object
		แสดงถึง **method** ที่อยู่ภายใน **object**

คำว่าพร้อมใช้งานในทางคอมพิวเตอร์สิ่งนั้นถูกสร้างและโหลดเข้าสู่หน่วยความจำและพร้อมสำหรับประมวลผล

ตัวอย่างการเขียนคลาส และ ตัวแปรคลาส

```
public class Ro3 {
    private  int salary;
    public static void main(String args[ ]) {

    }
}
```

Class variable

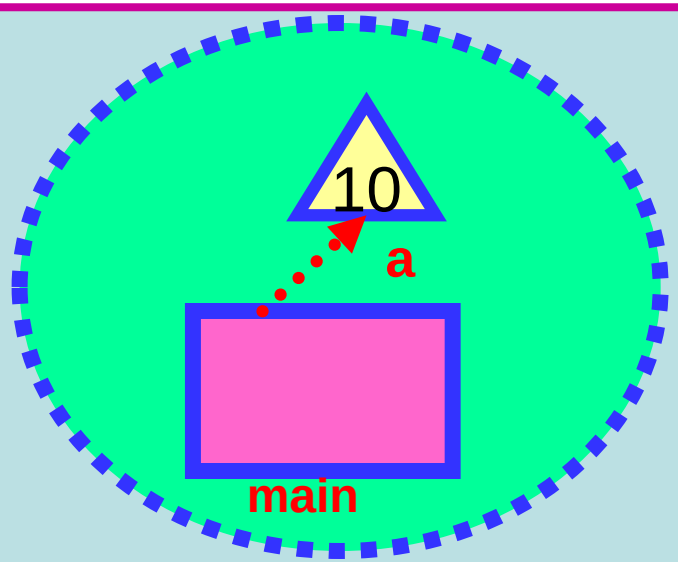
1.1 Class variable(attribute) ซึ่งเป็นชนิดข้อมูล primitive และมี static

```
1 class PublicStaticData{
2  public static int a = 10;
3
4  public static void main(String args[]){
5     System.out.println(a);
6  }
7 }
```

Class variable

10

จินตนาการ



PublicStaticData

Default package

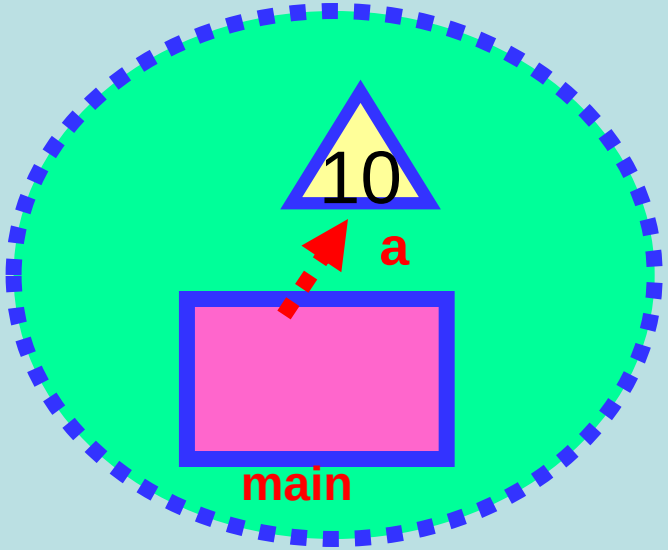
1.1 Class variable(attribute)

ซึ่งเป็นชนิดข้อมูล primitive และมี **final , static**

```
1 class FinalStaticData{
2   final static int a = 10;
3                  .......
4  public static void main(String args[]){
5     System.out.println(a);
6  }
7 }
```

10

จินตนาการ



FinalStaticData

Default package

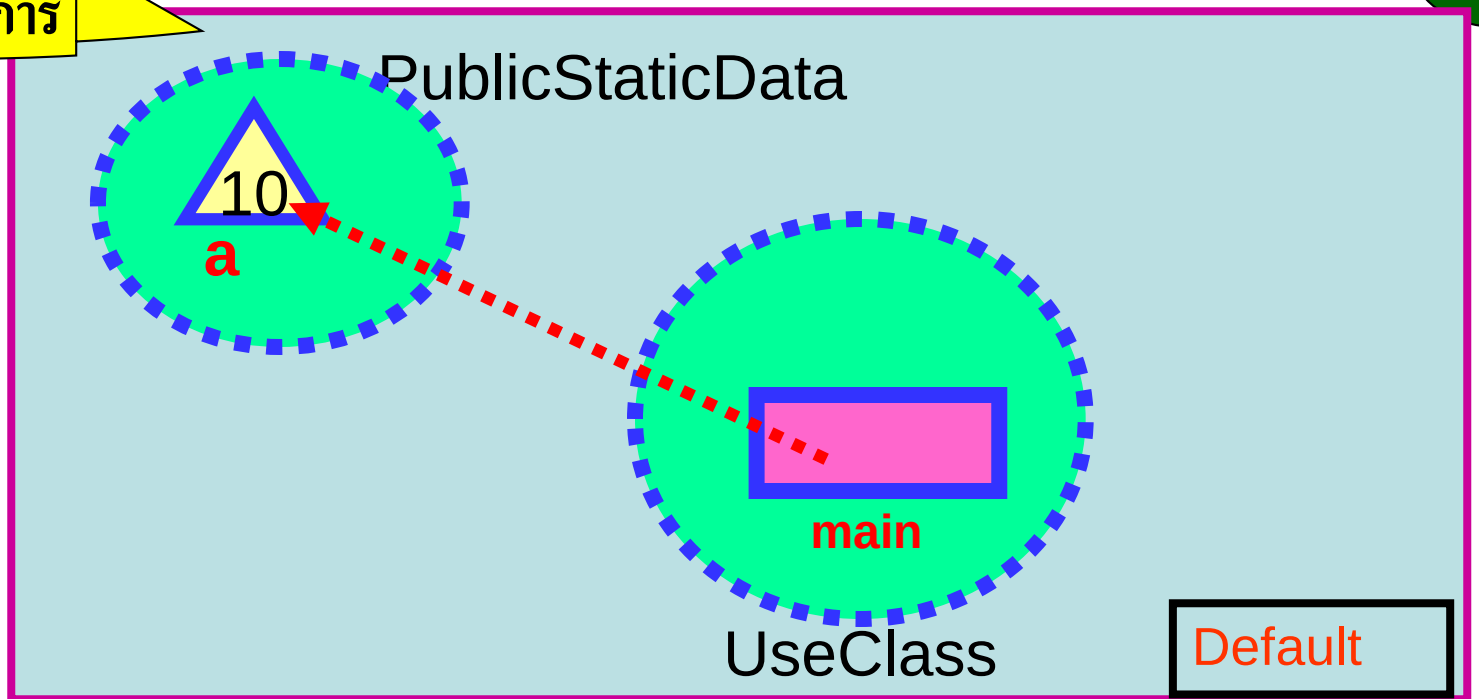
1.1 Class variable(attribute)

ซึ่งเป็นชนิดข้อมูล primitive และมี **static** แต่อยู่คนละคลาส

```
1 class PublicStaticData{
2     public static int a = 10;
3 }
4
5 class UseClass {
6  public static void main(String args[]){
7      System.out.println(PublicStaticData.a);
8  }
9 }
```

จินตนาการ

10



1.1 Class variable(attribute)

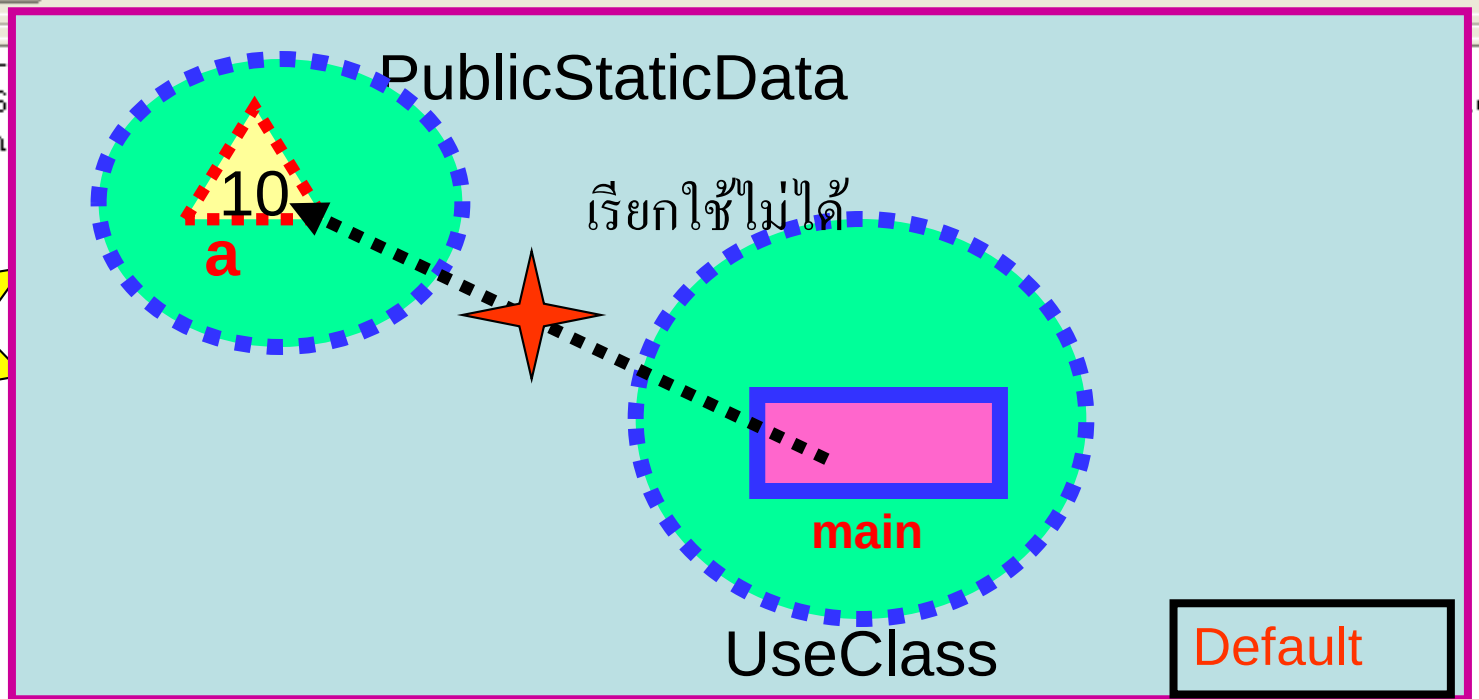
ซึ่งเป็นชนิดข้อมูล primitive**และ**ไม่มี static และอยู่ต่างคลาส

```
1 class PublicStaticData{
2   public int a = 10;
3 }
4
5 class UseClass {
6  public static void main(String args[]){
7     System.out.println(PublicStaticData.a);
8  }
```

PublicStaticData

```
----------
H:\PublicS
    System
```

c context

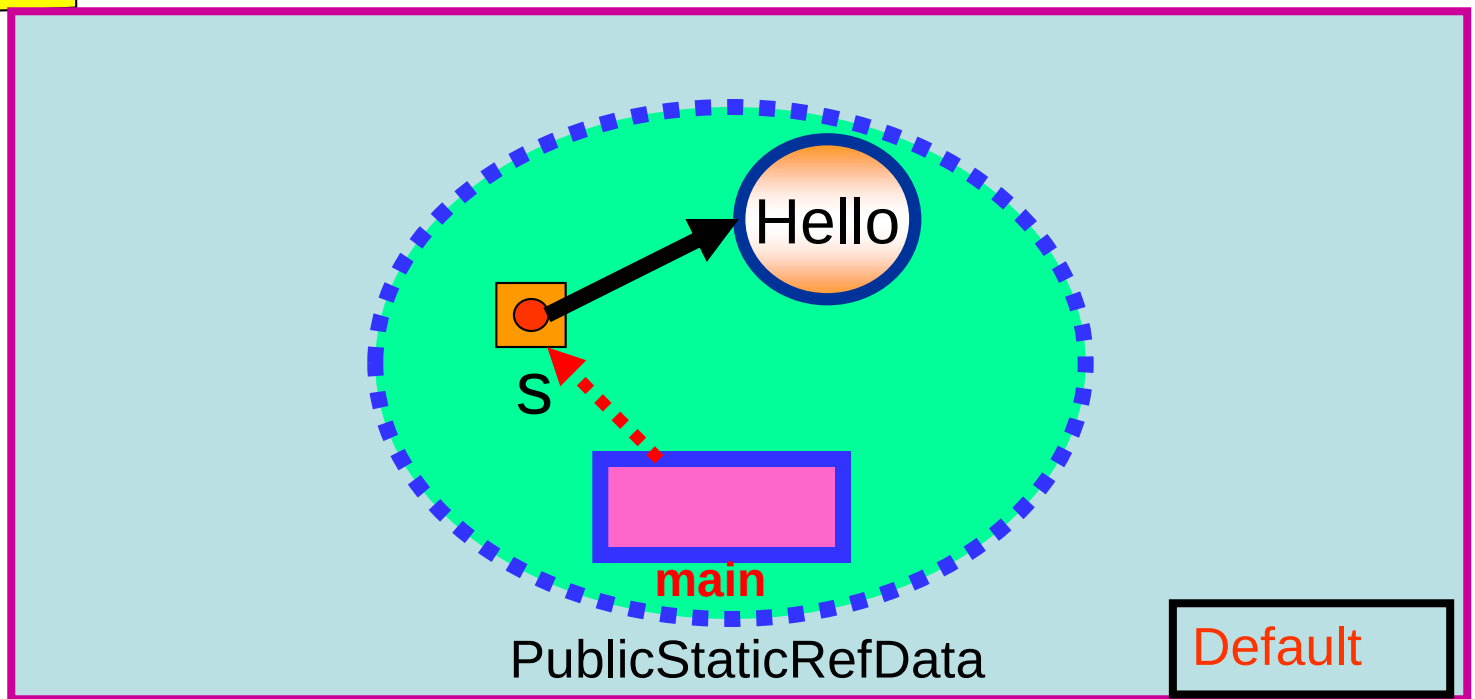


จินตนาการ

Class variable

1.2 Class variable ซึ่งเป็นชนิดข้อมูล class และมี static

```
1 class PublicStaticRefData{
2   public static String s = new String("Hello");
3
4   public static void main(String args[]){
5      System.out.println(s);
6   }
7 }
8
```



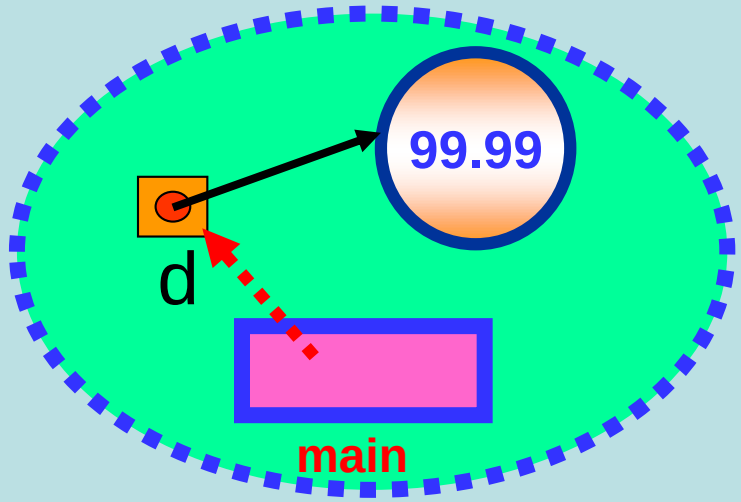
1.2 Class variable ซึ่งเป็นชนิดข้อมูล class และไม่มีการระบุการเข้าถึง

Class variable

```
1 class DefaultStaticRefData{
2     static Double d = new Double(99.99);
3
4   public static void main(String args[]){
5       System.out.println(d.toString());
6   }
7 }
```

99.99

จินตนาการ



DefaultStaticRefData

Default

package

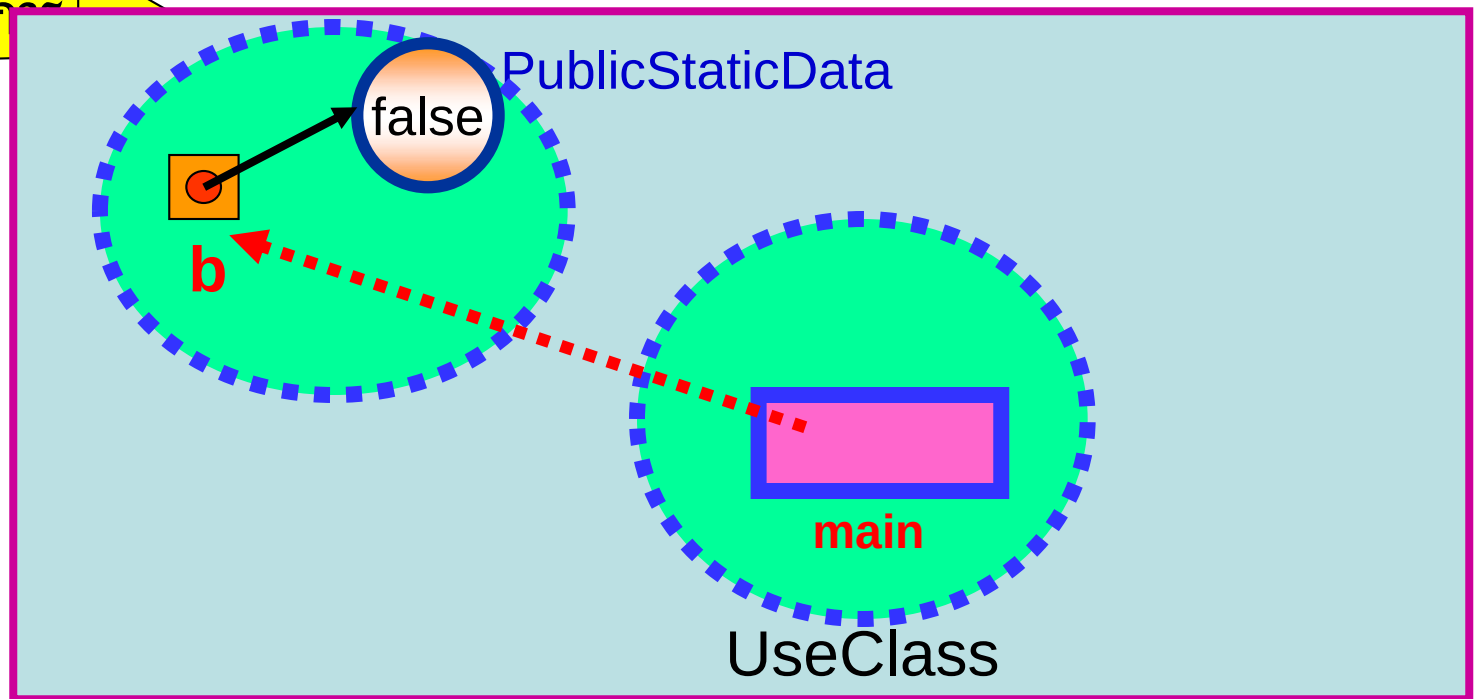
1.2 Class variable ซึ่งเป็นชนิดข้อมูล class เข้าถึง data แต่อยู่ละคลาส

```
1 class PublicStaticData{
2   public static Boolean b = new Boolean(10>20);
3 }
4
5 class UseClass {
6  public static void main(String args[]){
7     System.out.println(PublicStaticData.b.booleanValue());
8  }
9 }
```

Class variable

false

จินตนาก



ตัวอย่างการเขียนคลาส และ ตัวแปรคลาส

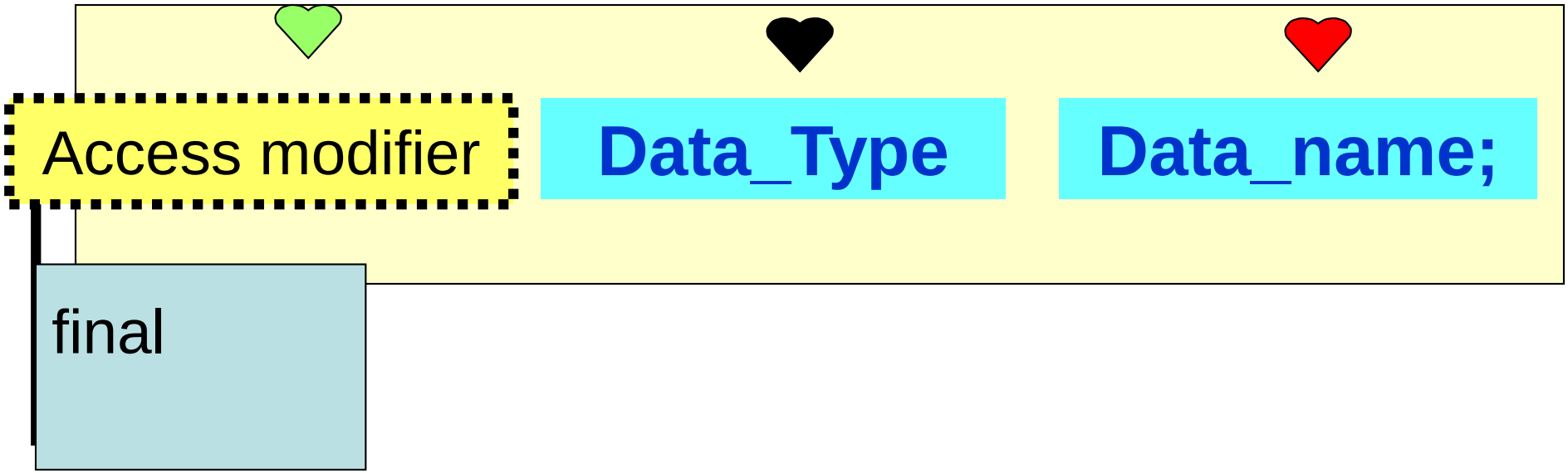
```
public class Ro3 {

    private  int salary;

    public static void main(String args[ ]) {
       int a;
       final  int B=10;

    }
}
```

Class variable

Method variable

2.Method Variable

รูปแบบตัวแปรภายในเมธอดประกอบด้วย



หมายเหตุ **Data_name** คือชื่อตัวแปร

Data_Type คือประเภทของข้อมูล

final กำหนดจะทำให้ตัวแปรไม่สามารถเปลี่ยนค่าได้

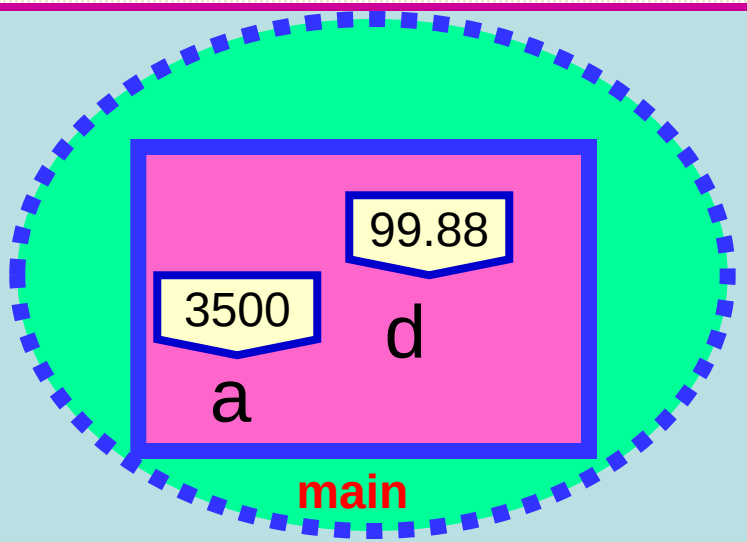
2.1 Method variable ซึ่งเป็นชนิดข้อมูล primitive

Method variable

```
1 class PrimitiveVariable{
2   public static void main(String args[]){
3     double d;
4     d = 99.88;
5     System.out.println(d);
6     int a = 3500;
7     System.out.println(a);
8   }
9 }
```

99.88
3500

จินตนาการ



PrimitiveVariable

Default package

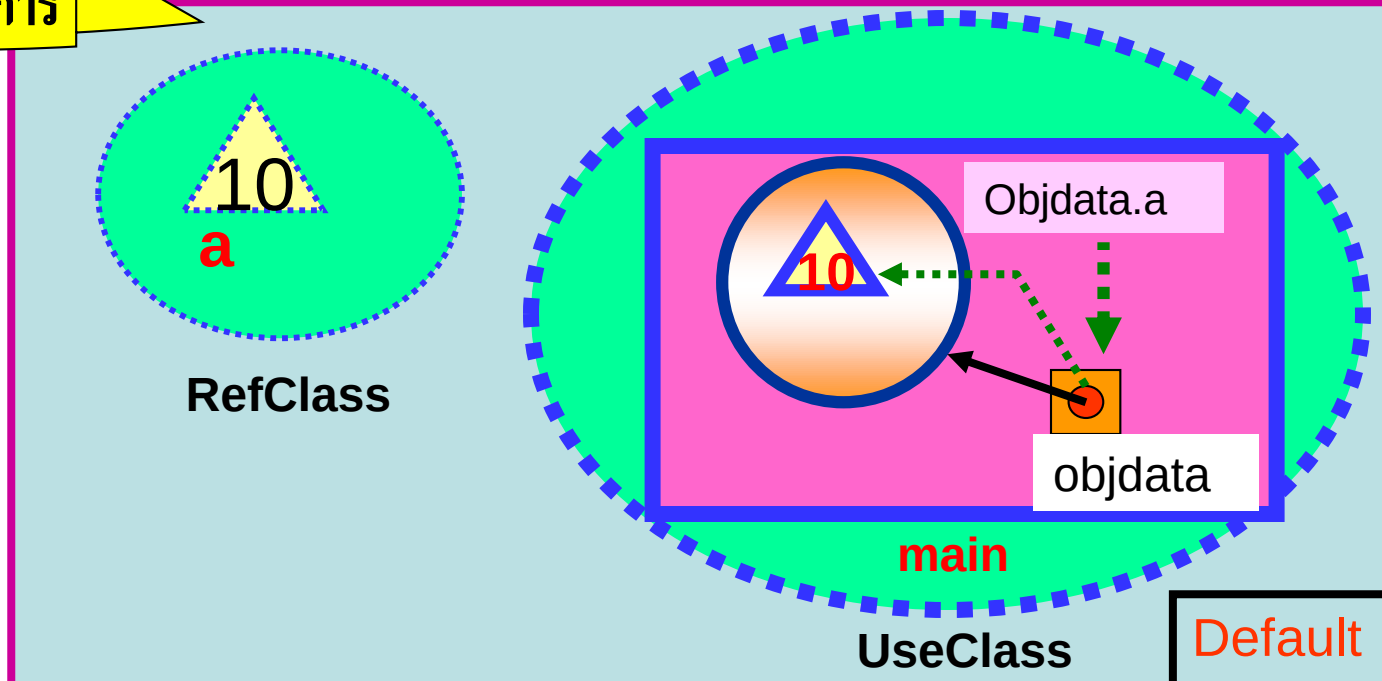
2.2 Method variable ซึ่งเป็นชนิดข้อมูล class

```
1  class RefClass{
2    public int a = 10;
3  }
4
5  class UseClass {
6    public static void main(String args[]){
7      RefClass objdata = new RefClass();
8      System.out.println(objdata.a);
9    }
10 }
```

Method variable

10

จินตนาการ

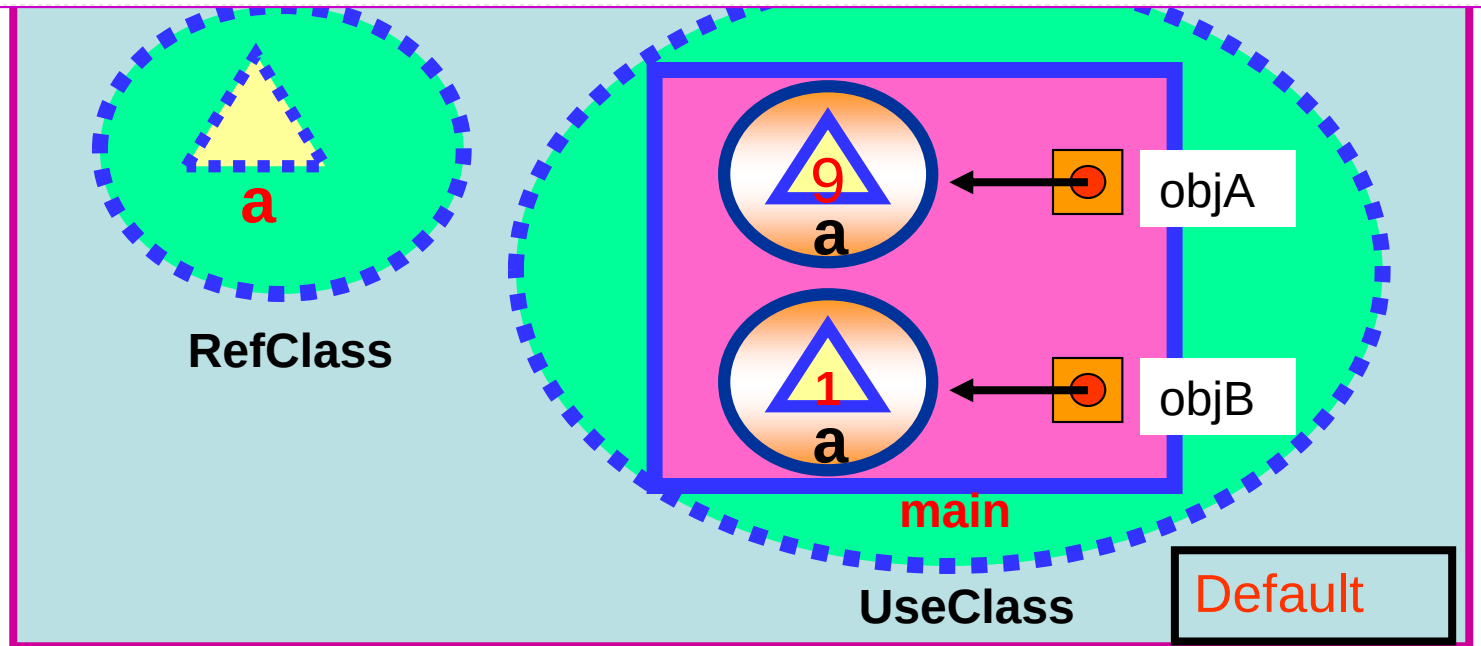


2.2 Method variable ซึ่งเป็นชนิดข้อมูล class

```
1  public class RefClass{
2     protected int a;
3  }
4
5  class UseClass {
6     public static void main(String args[]) {
7        RefClass objA = new RefClass();
8        objA.a = 9;
9        System.out.println(objA.a);
10       RefClass objB = new RefClass();
11       objB.a = 1;
12       System.out.println(objB.a);
13    }
14 }
```

Method variable

9
1



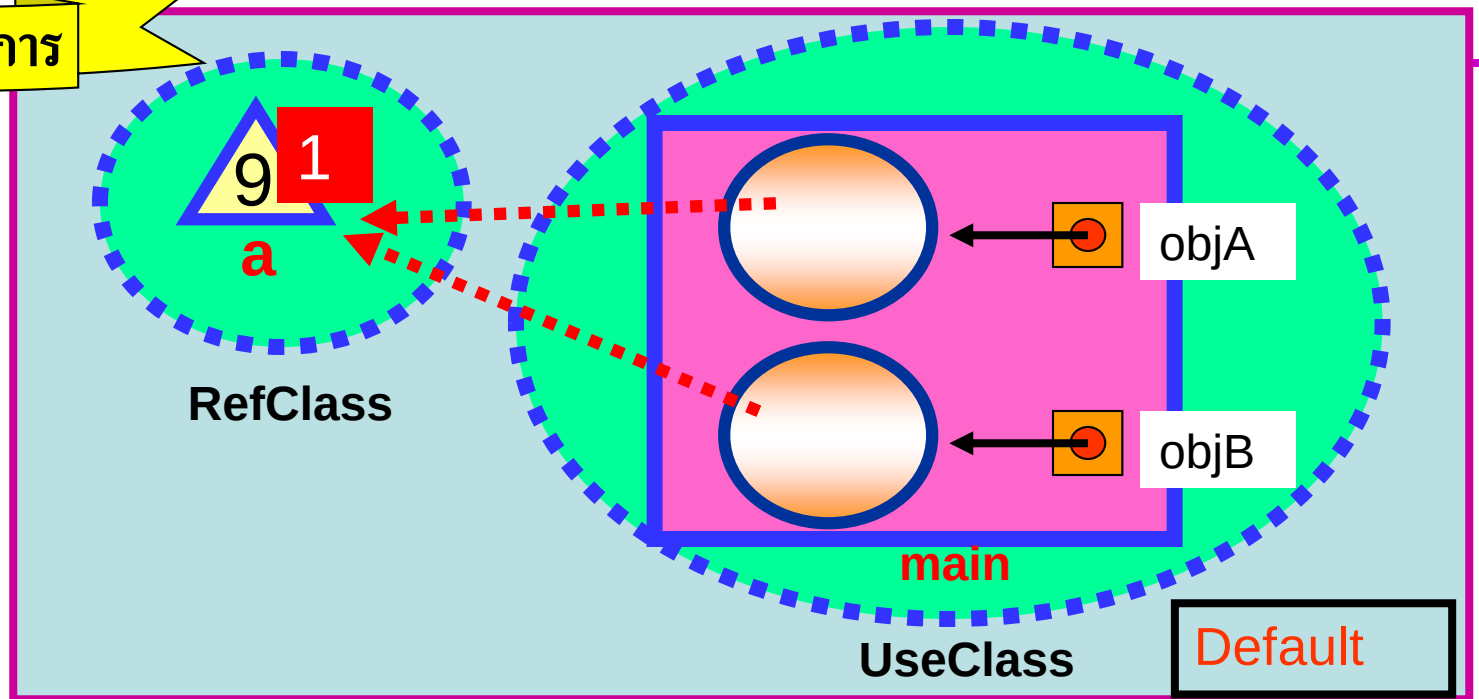
2.2 Method variable ซึ่งเป็นชนิดข้อมูล class การเข้าถึง static

```
1  public class RefClass{
2     static int a;
3  }
4  class UseClass {
5     public static void main(String args[]){
6        RefClass objA = new RefClass();
7        RefClass objB = new RefClass();
8        objA.a = 9;
9        objB.a = 1;
10       System.out.println(objA.a);
11       System.out.println(objB.a);
12    }
```

Method variable

1
1

จินตนาการ



ตัวอย่างทำความเข้าใจเกี่ยวกับแนวคิดของ class-Object

ของกรณีศึกษา Case A

Object->(attributes) , Object->(methods)

******หมายเหตุ ให้เขียน 1 Case / 1 แผ่นกระดาษ A4**

ตัวอย่าง
Case A