



បណ្ណបណ្ណាលក្នុង សម្បត្តិ អេច អ ឌី **Korea Software HRD Center**

Inheritance & Encapsulation & Polymorphism

ណែនាំដោយ : Dr. Kim Tae Kyung



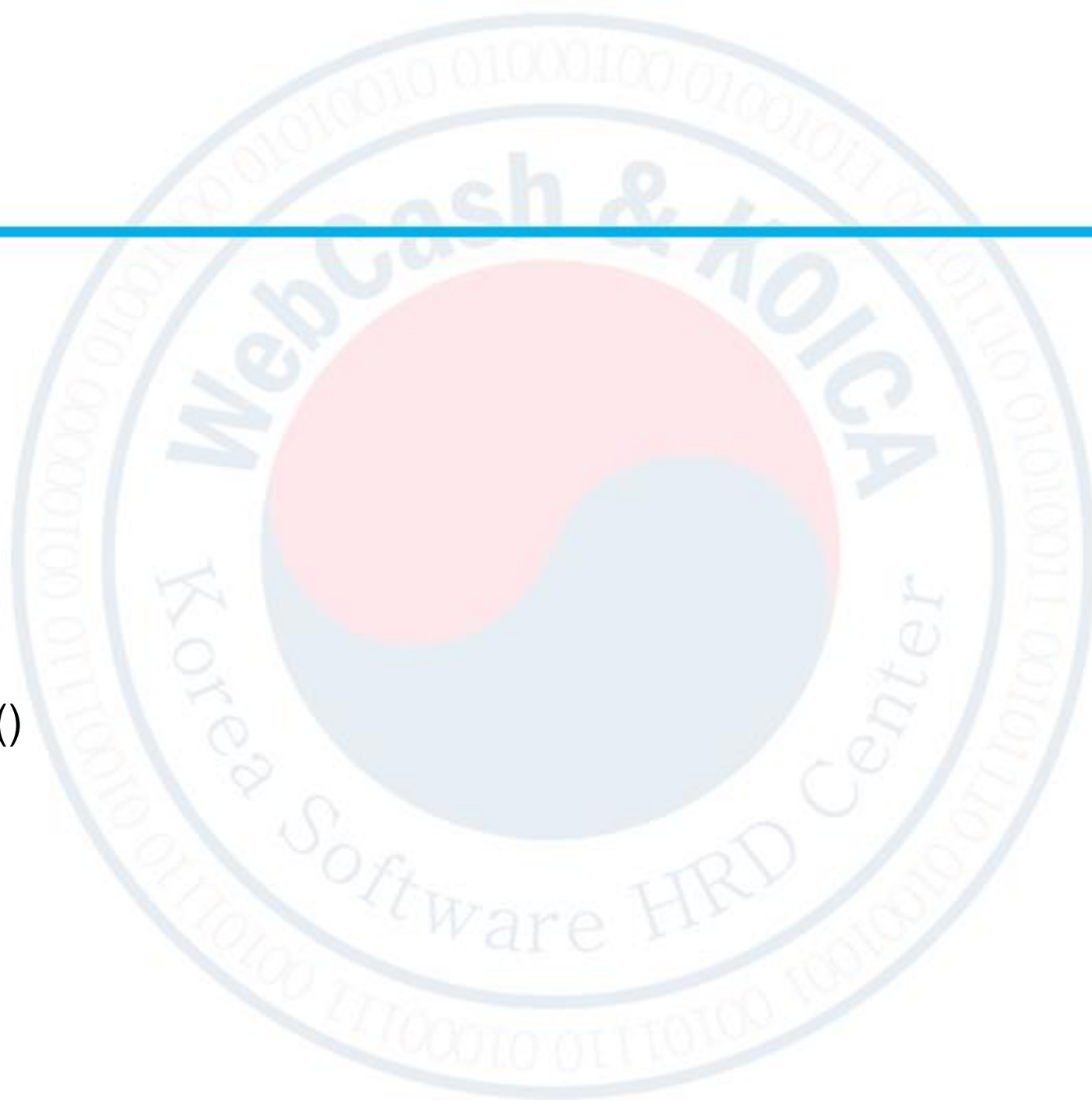
<http://www.kshrd.com.kh>

୭. Inheritance

୮. Encapsulation

୯. Polymorphism

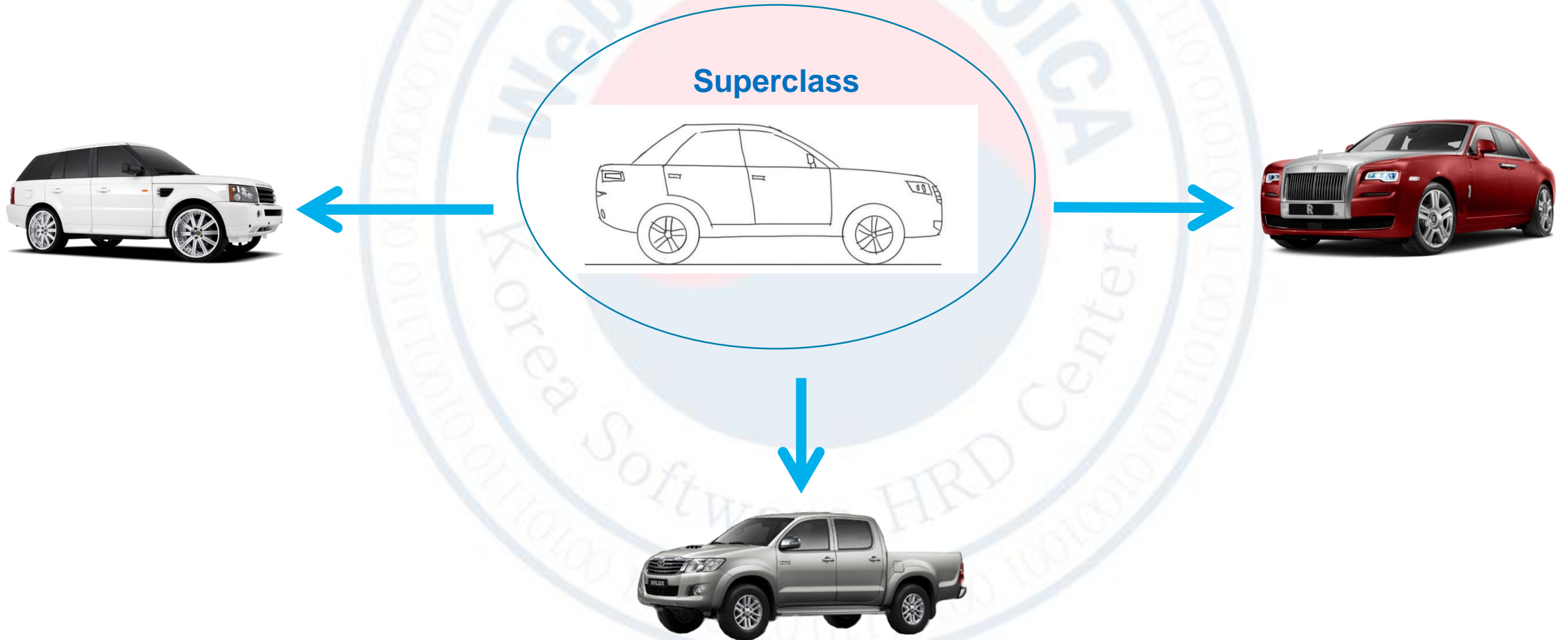
୧୦. Super & Super()



១. Inheritance

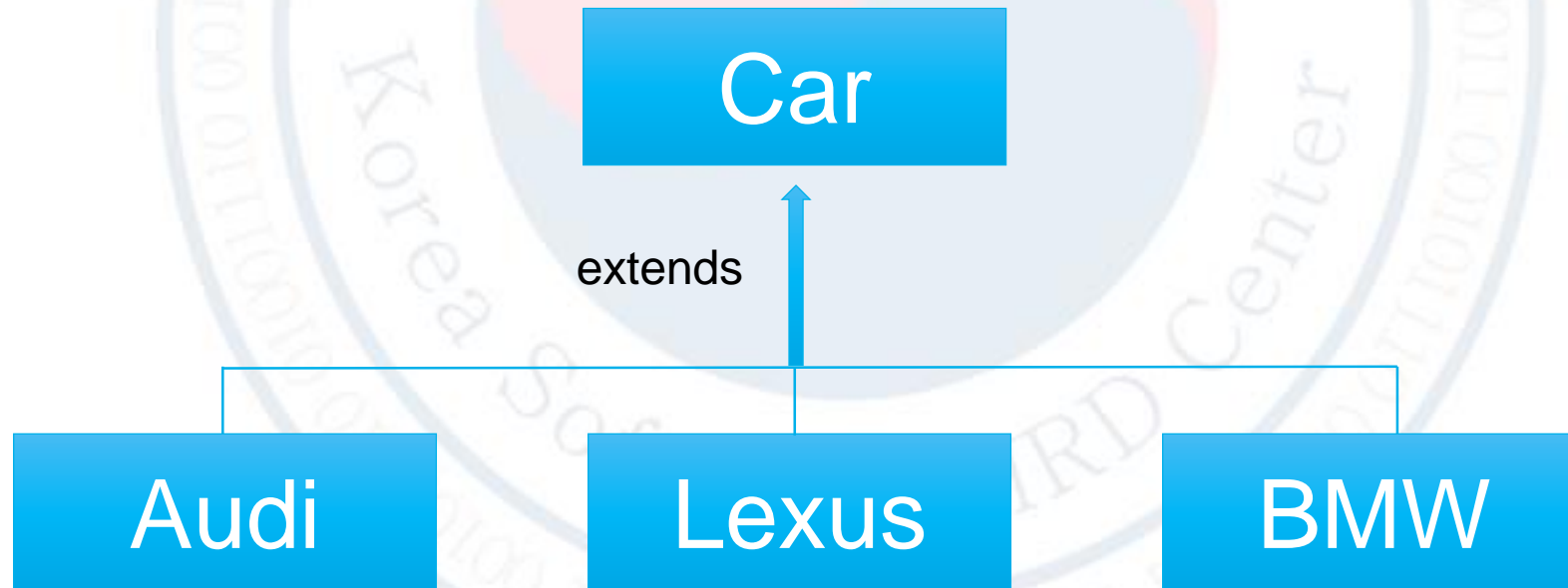
- Inheritance គឺជាការផ្ទេរ member របស់ Class មេ ទៅអោយ Class កូន
- នៅពេលដែល Class កូនទាំងឡាយមាននូវ member ដូចគ្នា យើងគួរតែបង្កើតវានូវក្នុង Class មេ
- នៅពេលដែលធ្វើការ Inheritance ហើយ, យើងក៏អាចបន្ថែមនូវ member ផ្សេងទៀតនៅក្នុង Class កូនផងដែរ
- Class មេ ហៅថា Super Class ឬ Base Class ឬ Parent Class
- Class កូន ហៅថា Sub Class ឬ Derived Class ឬ Child Class

9. Inheritance (𑌐)

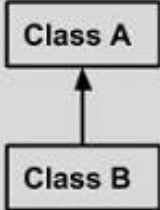
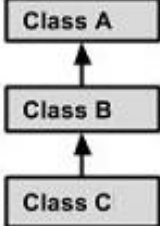
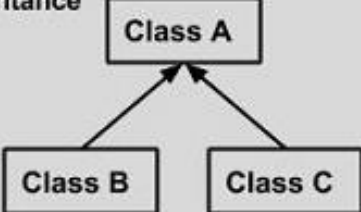
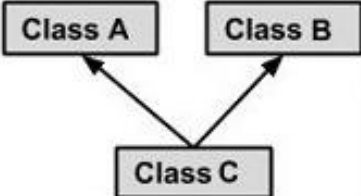


១. Inheritance (ឆ)

- ដើម្បីធ្វើ Inheritance យើងត្រូវ៖
 - បង្កើត Class ថ្មីមួយដោយធ្វើការ extends ចេញពី Class មួយទៀត



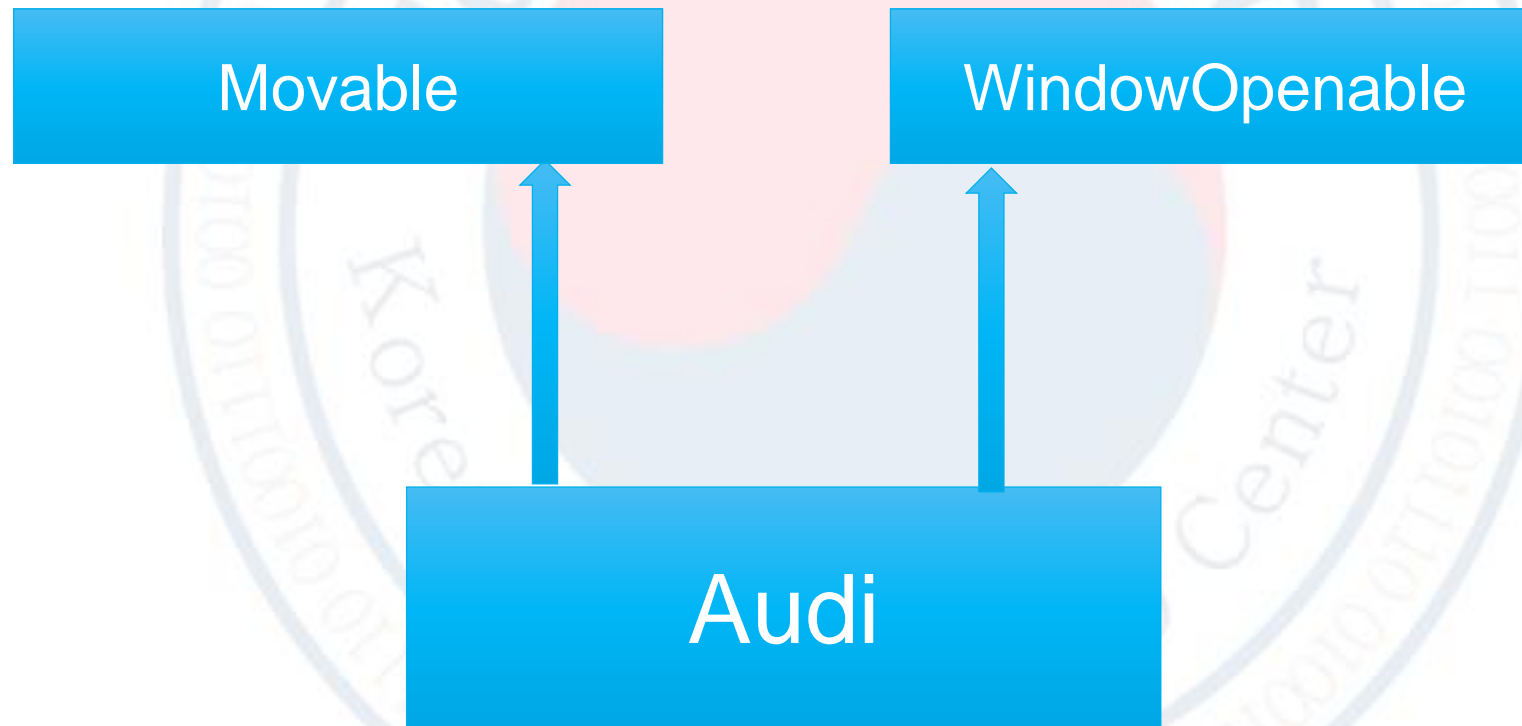
9. Inheritance (३)

Single Inheritance	 <pre>graph BT; B[Class B] --> A[Class A]</pre>	<pre>public class A { } public class B extends A { }</pre>
Multi Level Inheritance	 <pre>graph BT; C[Class C] --> B[Class B]; B --> A[Class A]</pre>	<pre>public class A {} public class B extends A {.....} public class C extends B {.....}</pre>
Hierarchical Inheritance	 <pre>graph BT; B[Class B] --> A[Class A]; C[Class C] --> A</pre>	<pre>public class A {} public class B extends A {.....} public class C extends A {.....}</pre>
Multiple Inheritance	 <pre>graph BT; C[Class C] --> A[Class A]; C --> B[Class B]</pre>	<pre>public class A {} public class B {.....} public class C extends A,B { } // Java does not support mutiple Inheritance</pre>

១. Inheritance (ឆ)

- ប្រសិនបើការធ្វើ Inheritance ចំពោះ Interface វិញ គឺប្រើនូវ Keyword implements
- នៅក្នុង Java ការធ្វើ Inheritance គឺអាច extends បានតែ Class មួយប៉ុណ្ណោះ ប៉ុន្តែអាច implements ច្រើន Interface

9. Inheritance (𑌕)



១. Inheritance (ឆ)

- Inheritance ចែកជា ២ ប្រភេទគឺ៖
 - IS-A Relationship
 - HAS-A Relationship (Composite)

9. Inheritance (𑌐)

- IS-A Relationship

Example:

// Class Car is a super Class

```
public class Audi extends Car {  
  
}
```

Note: Audi is a Car

Car is not an Audi

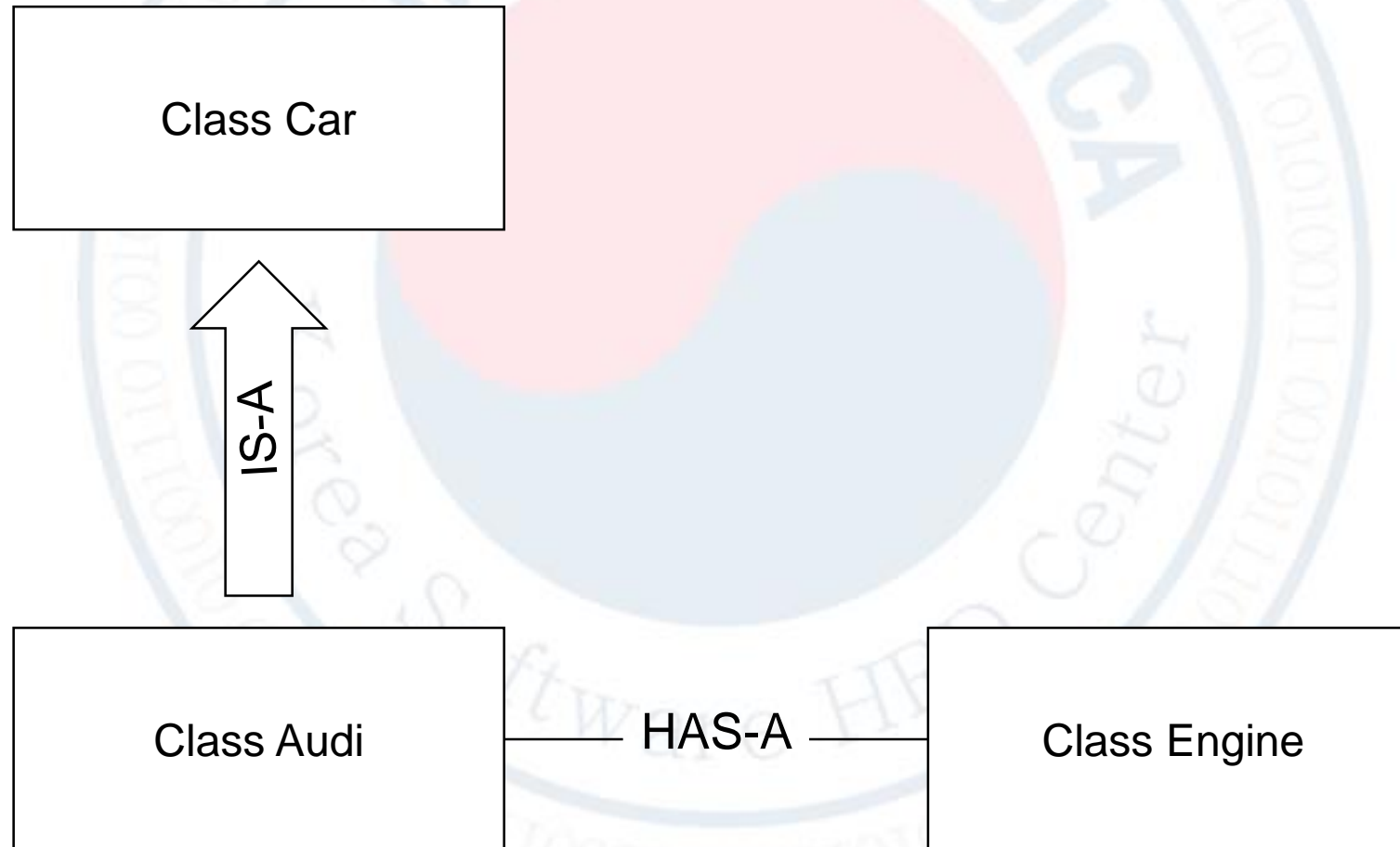
9. Inheritance (ㄹ)

- HAS-A Relationship (Composite)

Example:

```
public class Audi extends Car {  
    Engine e = new Engine();  
}  
  
class Engine {  
}
```

9. Inheritance (३)



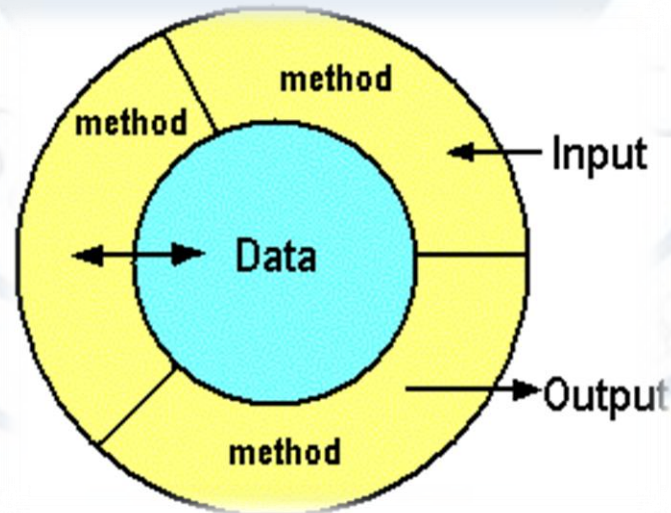
២. Encapsulation

- Encapsulation គឺជាការធ្វើអោយទិន្នន័យមាន សុវត្ថិភាព និង Data Hiding (មិនអោយ Class ខាងក្រៅចូលមកប្រើ Data members របស់ Class នេះបាន)។
- យើងត្រូវបង្កើតរាល់ Data members (Instance Variable) របស់វាទាំងអស់ជា Private
- ដើម្បីអោយ Class ខាងក្រៅចូលមកប្រើប្រាស់ Data Member នោះបានគឺយើងត្រូវបង្កើត public method Setter() និង Getter() ហើយអោយគេហៅ method នោះយកទៅប្រើជំនួសវិញ។

២. Encapsulation (ត)

អត្ថប្រយោជន៍ របស់ **Encapsulation**

- + គ្រប់គ្រងលើដំណើរការនៃ Data ក្នុងការទាញយក (get) ឬ បោះតម្លៃអោយ (set)
- + មាន Security ដោយ Data ត្រូវបានការពារដោយ Method។
- + យើងអាចកំណត់ Data Members របស់ Class ជា Read-Only ឬ Write-Only បាន។



៣. Polymorphism

- Polymorphism មានន័យថា ច្រើនទំរង់ ដែលវាអាចមាននូវ Method ច្រើនដែលមានឈ្មោះដូចគ្នា ប៉ុន្តែការងាររបស់វាផ្សេងៗពីគ្នា
- Polymorphism មាន ២ប្រភេទគឺ៖
 - Overloading : Method ២ ឬ ច្រើនដែលមានឈ្មោះដូចគ្នាប៉ុន្តែមានទំរង់ផ្សេងគ្នា
(Compile-Time Polymorphism)
 - Overriding : ការ Override Method ណាមួយដែលមានឈ្មោះនិងទំរង់ដូចទៅនឹងរបស់ Super Class នៅក្នុង Sub Class (Runtime Polymorphism) ដែល Compiler ជាអ្នកជ្រើសយក Method មួយណាមក execute

8. Polymorphism (8)

➤ Overloading

Example: print(int), print(double), print(boolean), print(String), etc.

```
void printResults() {  
    System.out.println("total = " + total + " average = " + average);  
}
```

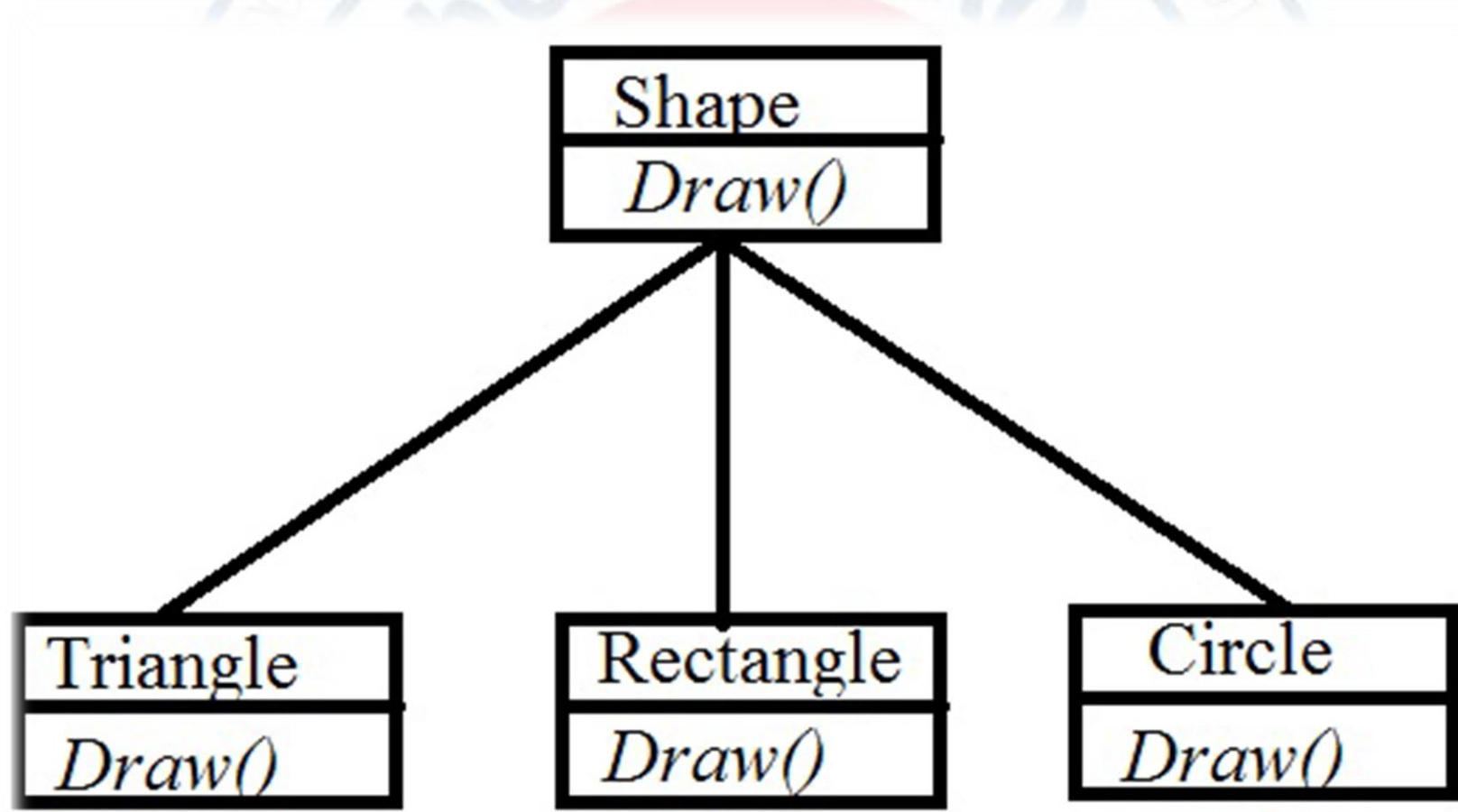
```
void printResult(String message) {  
    System.out.println(message + ": ");  
    printResults();  
}
```

៣. Polymorphism (ត)

➤ គោលការណ៍របស់ Overriding:

- ត្រូវតែជា IS-A relationship (inheritance) ដោយប្រើ extends ឬ implements
- Method ត្រូវមានឈ្មោះនិងទំរង់ដូចទៅនឹងរបស់ Super Class
- យើងមិនអាច Override static ឬក៏ final method បានទេ
- គួរតែប្រើនូវ *@Override annotation* នៅពីលើ Method ដែលបាន Override នៅក្នុង Sub Class

୩. Polymorphism (୩)



៤. Super & Super()

- `super`:
 - សំដៅទៅលើ Super Class នៃ Current Class
 - ប្រើដើម្បីហៅនូវ member របស់ Super Class
- `super()`:
 - ប្រើសំរាប់ហៅ Constructor របស់ Super Class

6. Super & Super()

```
class Shape {  
    protected int width, height;  
  
    public Shape(){  
        this(10, 5); //Referring to its overloaded  
    }  
  
    public Shape(int width, int height){  
        this.width=width;  
        this.height=height;  
    }  
}
```


6. Super & Super() (🌀)

```
class Rectangle extends Shape {  
    public Rectangle() {  
        super(); //Referring to superclass (Shape)'s constructor;  
        System.out.println("Area of Rectangle=" + (super.width * super.height));  
    }  
}  
  
public class MyClass {  
    public static void main(String[] args) {  
        Rectangle r = new Rectangle();  
    }  
}
```