មជ្ឈមណ្ឌលកូរ៉ែ សហ្វវែរ អេច អ ឌី

Korea Software HRD Center

ការស្វែងយល់ពី Abstract Class និង Interface ក្នុង Java

ណែនាំដោយ : Dr. Kim Tae Kyung

Webcash

KOICA
Korea International Cooperation Agency

មាតិកា

១. ការប្រើប្រាស់ខ្លួន abstract class

អ្វីទៅជា Abstraction?

- Abstraction គឺជា process នៃការ hiding implementation details ហើយបង្ហាញតែ functionality ទៅកាន់ User។

- មួយវិញទៀត វាធ្វើការបង្ហាញតែផ្នែកសំខាន់ទៅកាន់ User ប៉ុណ្ណោះ ហើយវាធ្វើការលាក់នូវពត៌មានខាងក្នុង។

១. ការប្រើប្រាស់ខ្លួន abstract class(ត)

- Abstraction ចែកចេញជា ២ គឺ៖
  - Abstract class
  - Interface

១. ការប្រើប្រាស់ខ្លួន abstract class

អ្វីទៅជា Abstract Class ?

- Abstract class គឺជា class ដែលត្រូវបានប្រកាសដោយប្រើប្រាស់នូវ keyword "abstract" ។
- Abstract class ត្រូវបានកំណត់ដោយ method មិនមាន body (undefined method) ហើយវាក៏អាចមាន concrete method មាន body(defined method) ផងដែរ។
- នៅក្នុងភាសា Java Programming undefined method ត្រូវបានគេដឹងថាជា un-Implemented ឬក៏ abstract method។

១. ការប្រើប្រាស់ខ្លួន abstract class(ត)

អ្វីទៅជា Abstract Class ?

- យើងមិនអាចយក abstract class មកបង្កើត object បាន។ គោលបំណងរបស់ abstract class គឺឲ្យ class ដទៃទៀតធ្វើការ extends ការងារមកធ្វើបន្ត។

- **ឧទាហរណ៍៖** នៅក្នុងក្រុមហ៊ុនមួយ គេបង្កើត abstract class ដើម្បីឲ្យអ្នកផ្សេង extends ការងារនោះមកធ្វើបន្ត(override)។

១. ការប្រើប្រាស់ឆ្លង abstract class(ត)

- Syntax

```
abstract class className {

        ..............

}
```

- **ឧទាហរណ៍៖** ប្រសិនបើ class មួយផ្ទុកនូវ abstract method នោះយើងត្រូវប្តូរ class នោះទៅជា abstract class។

```
class Vachile {

        abstract void Bike();

}
```

ប្តូរពី class ទៅ abstract class

```
abstract class Vachile {

        ......

}
```

១. ការប្រើប្រាស់ខ្លួន abstract class(ត)

អ្វីទៅជា Abstract Method ?

- Abstract method គឺជា method ដែលមានតែ declaration ឬក៏ prototype ប៉ុន្តែវាមិនមាន body គឺវាមិនមានសញ្ញា {} ឬក៏ definition ឡើយ។
- ដើម្បីបង្កើតនូវ undefined method ទៅជា abstract ដោយការប្រកាសរបស់វាត្រូវតែកំណត់ជាមុននូវ keyword "abstract" នៅពីមុខ method ។
- Syntax:

```
abstract ReturnType methodName(List of formal parameter);
// No braces{ }
```

១. ការប្រើប្រាស់ឆ្លង abstract class(ត)

- Example:

```
abstract void sum();
abstract void diff(int, int);
```

```
abstract class Vehicle {
        abstract void speed(); // abstract method
}

class Bike extends Vehicle {
        void speed() {
                System.out.println("Speed limit is 40 km/hr..");
        }

        public static void main(String args[]) {
                Vehicle obj = new Bike(); //indirect object creation
                obj.speed();
        }
}
```

Speed limit is 40 km/hr..

១. ការប្រើប្រាស់ខ្លួន abstract class(ត)

- ក្នុង abstract class អាចមាន៖

- data member (field របស់ class)
- Abstract method (method without body)
- Concrete Method (method with body)
- Constructor
- main method

```
public abstract class Vehicle {
        Vehicle() { // constructor
                System.out.println("Vehicle's detail!");
        }

        void turnOn() { // concrete method
                System.out.println(“Turn on machine.");
        }

        public abstract void speed(); // abstract method

        public static void main(String[] args) {
        // main()
                System.out.println("Honda");
        }
}
```

២. ការប្រើប្រាស់នូវ interface

អ្វីទៅជា interface?

- Interface គឺវាស្រដៀងទៅនឹង class ដែលតែវាមិនមែនជា class ទេ។ Interface ដែលប្រមូលផ្តុំនូវ abstract methods, constant variables។
- វាគឺជាការប្រមូលផ្តុំនូវ public static final variables (constants) ហើយនិង abstract methods ។
- Interface មិនមែនជា class ឡើយ ព្រោះ class អាចបង្កើត object បាន ប៉ុន្តែ interface គឺមិនអាចទេ។

២. ការប្រើប្រាស់ឆ្លង interface (ត)

អ្វីទៅជា interface?

- ការសរសេរកូដរបស់ interface មានលក្ខណៈស្រដៀងទៅនឹងការសរសេរនៅក្នុង class ប៉ុន្តែវាមានលក្ខណៈខុសគ្នា ២ គឺ៖
  - Class ពិពណ៌នាអំពី attributes និង behaviors របស់ object។
  - Interface ផ្តុំកន្លែ behaviors ដែលសម្រាប់ឲ្យ class implements។

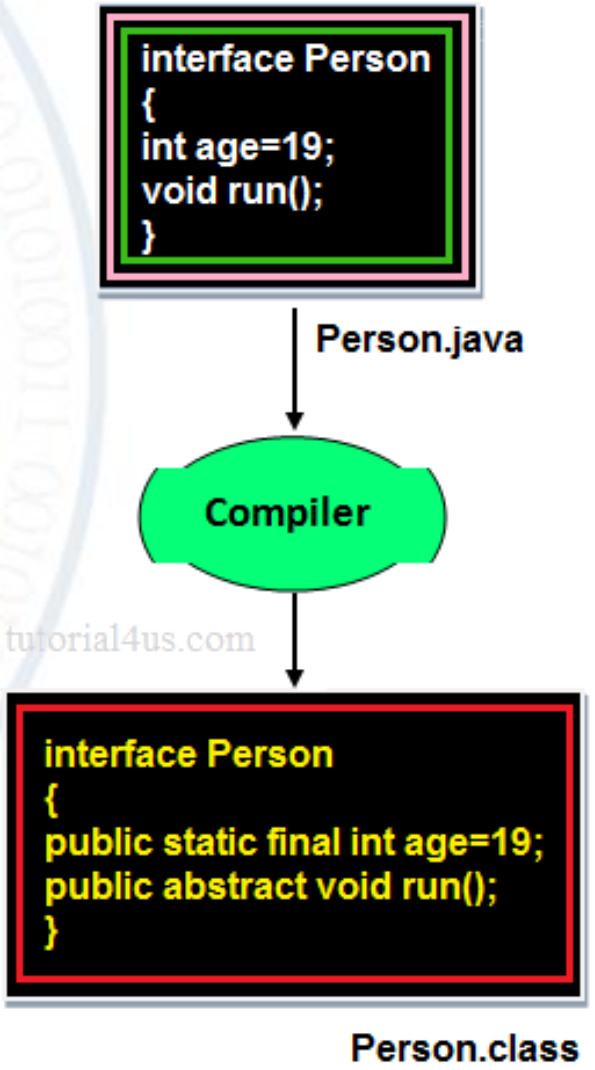
២. ការប្រើប្រាស់នូវ interface(ត)

Interface មានលក្ខណៈស្រដៀងទៅនឹង class ដូចតទៅ៖

- Interface អាចផ្ទុកនូវ methods បានច្រើន
- Interface ត្រូវបានសរសេរនៅក្នុង file ដែលមាន extension ".java" ហើយឈ្មោះរបស់វាគឺដូចទៅនឹងឈ្មោះរបស់ file
- Bytecode របស់វាគឺមាន extenstion ".class"

២. ការប្រើប្រាស់ឆ្លង interface(ត)

- ចំនុចសំខាន់ នៅពេលដែលយើងប្រកាសធម្មតា java compiler នឹងបន្ថែម public abstract keyword ទៅកាន់ interface method របស់យើង និង public static final ទៅកាន់ data member របស់វា។



២. ការប្រើប្រាស់នូវ interface(ត)

Interface មានលក្ខណៈខុសគ្នាទៅនឹង class ដូចតទៅ៖

- យើងមិនអាច instantiate ចេញពី Interface បានឡើយ។
- Interface មិនផ្ទុកនូវ constructor ឡើយ។
- រាល់ methods ទាំងអស់នៅក្នុង interface គឺជា abstract។
- Interface មិនត្រូវបាន extend ដោយ class ឡើយ ប៉ុន្តែវាត្រូវបាន implement ដោយ class។
- Interface អាចត្រូវបាន extend ពី multiple interfaces។
- Interface មិនផ្ទុកនូវ instance variable បានទេ ប៉ុន្តែវាអាចផ្ទុកនូវ public static final variables(Constant class variable) ។

២. ការប្រើប្រាស់ឆ្លង interface(ត)

- Syntax:

```
interface InterfaceName { …… }
```

- Example:

```
interface InterfaceName {
        datatype variableName=value;
        //Any number of final, static fields
        returntype methodName(list of parameters or no parameters)
        //Any number of abstract method declarations
 }
```

២. ការប្រើប្រាស់ឆ្លង interface(ត)

- របៀប Implement interface:
  - Class មួយប្រើនូវ keyword "implement" ដើម្បី implement នូវ interface។

```
/* Abstract method has
definition only */

public interface Movable {
        void moveLeft();
        void moveRight();

}
```

```
/* Subclass provide actual implementation */
public class MovablePoint implements Movable {
        public void moveLeft() {
                System.out.println("Left");
        }

        public void moveRight() {
                System.out.println("Right");
        }
}
```

២. ការប្រើប្រាស់ឆ្លង interface(ត)

- Multiple inheritance

```
interface Printable {
	void print();
}
interface Showable {
	void show();
}

class A7 implements Printable, Showable {
	public void print() {
		System.out.println("Hello");
	}

	public void show() {
		System.out.println("Welcome");
	}
}
```

២. ការប្រើប្រាស់ឆ្លង interface(ត)

- No Standard

```
public class LGRC {
        void turn_On() { }

        void turn_Off() { }
}

public class SamsungRC {
        void Turn_On() { }

        void Turn_Off() { }
}
```

```
public class Run {
        public static void main(String[] args) {
                LGRC lgrc = new LGRC();
                lgrc.turn_On();
                lgrc.turn_Off();

                SamsungRC samrc = new SamsungRC();
                samrc.Turn_On();
                samrc.Turn_Off();
        }
}
```

២. ការប្រើប្រាស់ឆ្លង interface(ត)

Communication Standard

```
public class SamsungRC implements Remote {
        public void turnOn() {
        }

        public void turnOff() {
        }
}
```

```
public class LGRC implements Remote {
        public void turnOn() {
        }

        public void turnOff() {
        }
}
```

```
public interface Remote {
        void turnOn();

        void turnOff();
}
```

```
public class Run {
        public static void main(String[] args) {
                LGRC lgrc = new LGRC();
                lgrc.turnOn();
                lgrc.turnOff();

                SamsungRC samrc = new SamsungRC();
                samrc.turnOn();
                samrc.turnOff();
        }
}
```

២. ការប្រើប្រាស់នូវ interface(ត)

- Interface ក៏អាចមាន Interface មួយទៀតនៅខាងក្នុងដែលគឺហៅថា nested interface។

```
public interface Mobile {
        public interface Battery{
        void charge();
    }
}
```

២. ការប្រើប្រាស់នូវ interface(ត)

- ចាប់ពី Java version 8 ទៅ Interface អាចមាន concrete method ដែល Declare ដោយចាប់ផ្តើមដោយ keyword default ឬ static ។ ចំនុចនេះហើយវាធ្វើអោយ Interface និង Abstract Class មិនមានភាពខុសគ្នាឆ្ងាយ ក្នុងការបន្ថែម Method ទៅអោយ Interface នាថ្ងៃមុខ ។
- Default method អាចអោយ class ដែល Implemented , override ឬ អត់ក៏បាន។

```
public interface Fly {
      default void display(){
            System.out.println("Default");
      }
}
```

២. ការប្រើប្រាស់ខ្លួន interface(ត)

- Static method ក្នុង Interface គឺដូចទៅនឹងការប្រើប្រាស់ static method ក្នុង Class ធម្មតាដែរ។

```
public interface Fly {
        static void test(){
                System.out.println("Static");
        }
}
```

```
public class Car implements Fly {
        public static void main(String[] args) {
                Fly.test();
        }
}
```

៣. ភាពខុសគ្នារវាង abstract class ហើយនិង interface

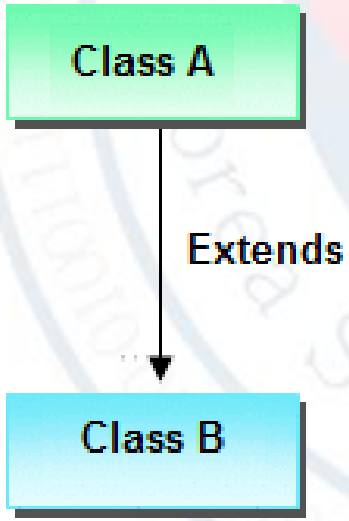
Abstract class	interface
Abstract class អាចត្រឹម inherit បានតែ ១ Class ឬ ច្រើន Interface	Interface មួយអាច Inherit Interface ច្រើន
Declare ដោយប្រើប្រាស់នូវ abstract keyword	Declare ដោយប្រើប្រាស់នូវ interface keyword
ប្រើ extends keyword សម្រាប់ implementation	ប្រើ implements keyword សម្រាប់ implementation
ផ្ទុកនូវតំលៃ fields ហើយនិង constant	ផ្ទុកតែតំលៃជា constant ប៉ុណ្ណោះ
អាចមាន constructor	មិនមាន constructor
Default access specifier នៃ abstract class methods គឺ default	Default access specifier នៃ interface method គឺ public

៣. ភាពខុសគ្នារវាង abstract class ហើយនិង interface(ត)

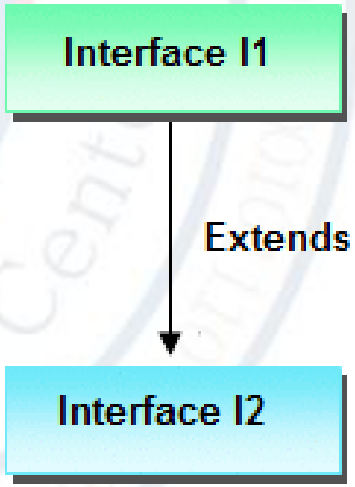
Abstract class	interface
យើងអាចមាន instance និង static block	យើងមិនអាចមាន instance និង static block
គ្មានការកំណត់សិទ្ធិក្នុងការប្រើប្រាស់ទៅលើ abstract class variable គឺវាអាចផ្ទុក non-public members	Variable មិនអាច declare ជា private, protected, transient, volatile គឺ Interface members ត្រូវបានកំណត់ជា public ដោយ default
អាចផ្ទុកនូវ Abstract Method ឬ Concrete method	គ្រប់ Method ជា Abstract method ដែលគ្មាន Body
Faster in performance	Slower in performance
Abstract class យើងអាចផ្តល់នូវ default implementation សម្រាប់ method ថ្មីបាន។ ហើយរាល់កូដទាំងអស់នៅដំណើរការដដែរ សម្រាប់ abstract class	ក្នុង interface បើយើងធ្វើការបន្ថែមនូវ method ថ្មី នោះ class ដែលធ្វើការ implementations ពី interface នេះត្រូវធ្វើការ override នូវ method ថ្មី។

៣. ភាពខុសគ្នារវាង abstract class ហើយនិង interface(ត)

- ទំនាក់ទំនងរវាង class និង interface:
  - Any class can extends another class

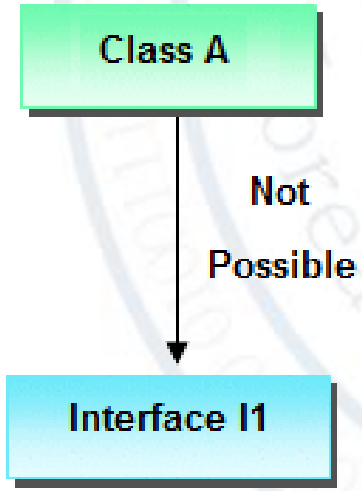


  - Any Interface can extends another Interface.

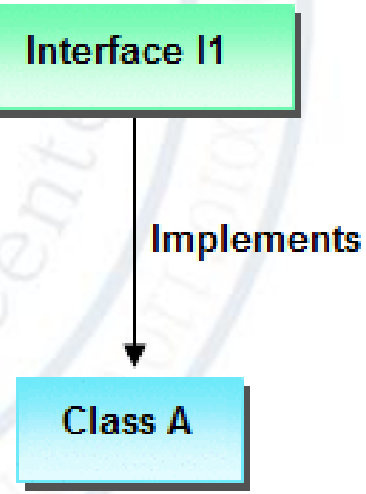


៣. ភាពខុសគ្នារវាង abstract class ហើយនិង interface(ត)

- ទំនាក់ទំនងរវាង class និង interface:
  - Any Interface can not extend or Implements any class.

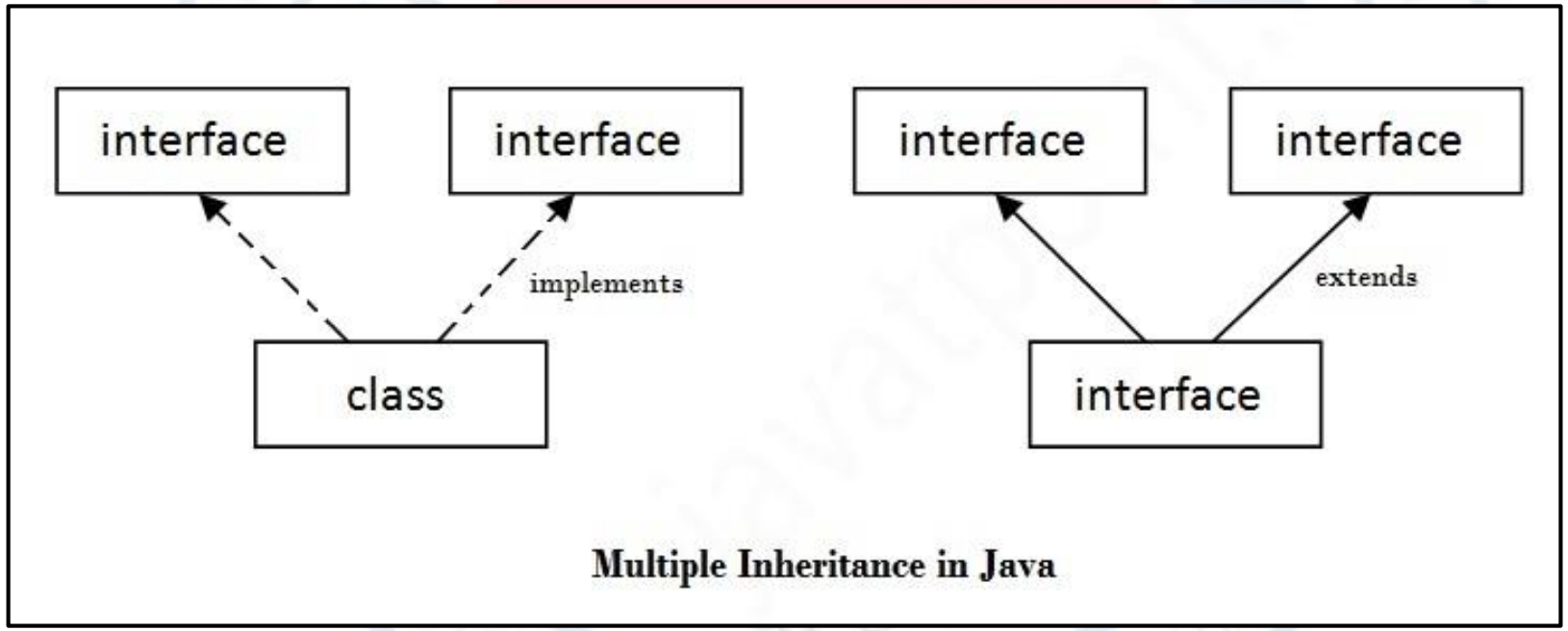


  - Any class can Implements another Interface



៣. ភាពខុសគ្នារវាង abstract class ហើយនិង interface(ត)

- Multiple inheritance with interface:



៣. ភាពខុសគ្នារវាង abstract class ហើយនិង interface(ត)

- តើពេលណាយើងប្រើ Abstract class?
  - ប្រសិនបើយើងនិយាយអំពី implementation ប៉ុន្តែមិនពេញលេញ (partially implemented) ដូច្នេះយើងប្រើនូវ abstract។
  - យើងប្រើវាក្នុងការចែករំលែក Code ទៅអោយ Class ដែលទាក់ទងនឹងវា ហើយឲ្យអ្នកដទៃធ្វើការបន្ថែមការងារមកលើវា ហើយ class ដែល extends ចេញពី abstract class គឺត្រូវធ្វើការ override ទៅលើ abstract method ទាំងឡាយណាដែលមាននៅក្នុង abstract class។
  - បើក្នុង abstract class មាន abstract method ផង និង method ធម្មតាផង ដូច្នេះយើងអាចធ្វើការ override តែ abstract method បានហើយ ចំពោះ concrete method មិនបាច់ override ក៏បានដែរ។ តែបើយើងចង់បន្ថែមនូវការងាររបស់យើង យើងអាចធ្វើការ override method ទាំងនោះបាន។

៣. ភាពខុសគ្នារវាង abstract class ហើយនិង interface(ត)

- តើពេលណាយើងប្រើ Abstract class?
  - Sub Class ដែល Inherit ពីវាមានប្រើប្រាស់នូវ Field & Method ដូចគ្នា ឬ ត្រូវការប្រើប្រាស់ Access Modifier ខុសពី Public ពីព្រោះនៅក្នុង Interface គ្រប់ Member ជា Public។
  - ប្រសិនបើនាពេលអនាគត យើងនឹងបន្ថែមនូវ Method មួយឬច្រើនទៅក្នុង Class បន្ថែមទៀត។

៣. ភាពខុសគ្នារវាង abstract class ហើយនិង interface(ត)

- តើពេលណាយើងប្រើ interface?
  - ប្រសិនបើយើងមិនដឹងអំពីការ implementation ទេ ដូច្នេះយើងប្រើនូវ interface ។
  - យើងប្រើវានៅពេល fully abstraction។
  - Interface support the functionality of multiple inheritances។
  - បង្កើតជា Communication Standard មួយរវាង Object ។
  - Method ទាំងឡាយមិនប្រែប្រួល ឬ មិនត្រូវការបន្ថែម Method អ្វីចូលបន្ថែមទៀត។