인터페이스 상속

인터페이스 상속

❖ 인터페이스간 상속 가능

```
public interface 하위인터페이스 extends 상위인터페이스1, 상위인터페이스2 { ... }
```

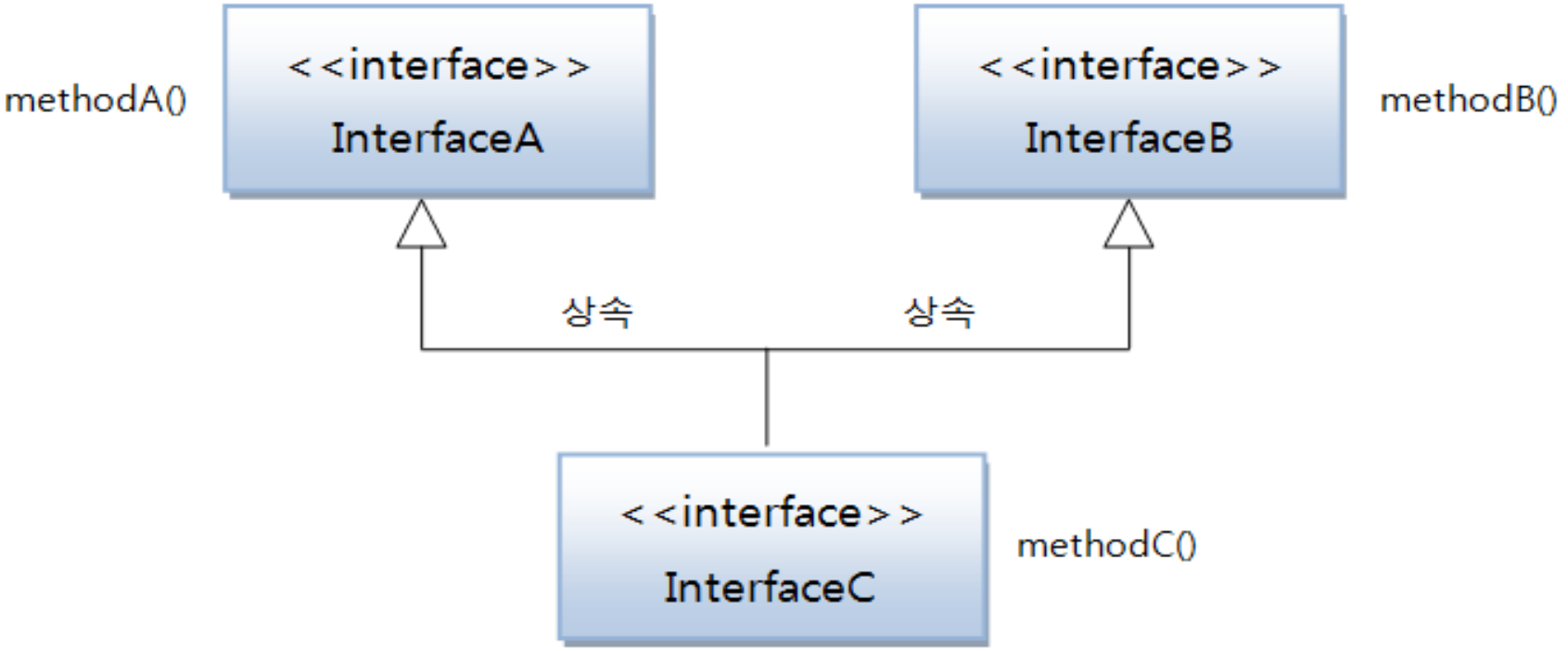
- 하위 인터페이스 구현 클래스는 아래 추상 메소드를 모두 재정의해야
  - 하위 인터페이스의 추상 메소드
  - 상위 인터페이스1의 추상 메소드
  - 상위 인터페이스2의 추상 메소드

```
하위인터페이스 변수 = new 구현클래스(...);
상위인터페이스1 변수 = new 구현클래스(...);
상위인터페이스2 변수 = new 구현클래스(...);
```

인터페이스 상속

❖ 인터페이스간 상속 가능

- 인터페이스 자동 타입 변환
  - 해당 타입의 인터페이스에 선언된 메소드만 호출 가능



인터페이스 상속

❖ 부모 인터페이스: InterfaceA.java

```
public interface InterfaceA {
    public void methodA();
}
```

❖ 부모 인터페이스: InterfaceB.java

```
public interface InterfaceB {
    public void methodB();
}
```

인터페이스 상속

❖ 하위 인터페이스: InterfaceC.java

```
public interface InterfaceC extends InterfaceA, InterfaceB {
    public void methodC();
}
```

❖ 하위 인터페이스 구현: ImplementationC.java

```
public class ImplementationC implements InterfaceC {
    public void methodA() {
        System.out.println("ImplementationC-methodA() 실행");
    }

    public void methodB() {
        System.out.println("ImplementationC-methodB() 실행");
    }

    public void methodC() {
        System.out.println("ImplementationC-methodC() 실행");
    }
}
```

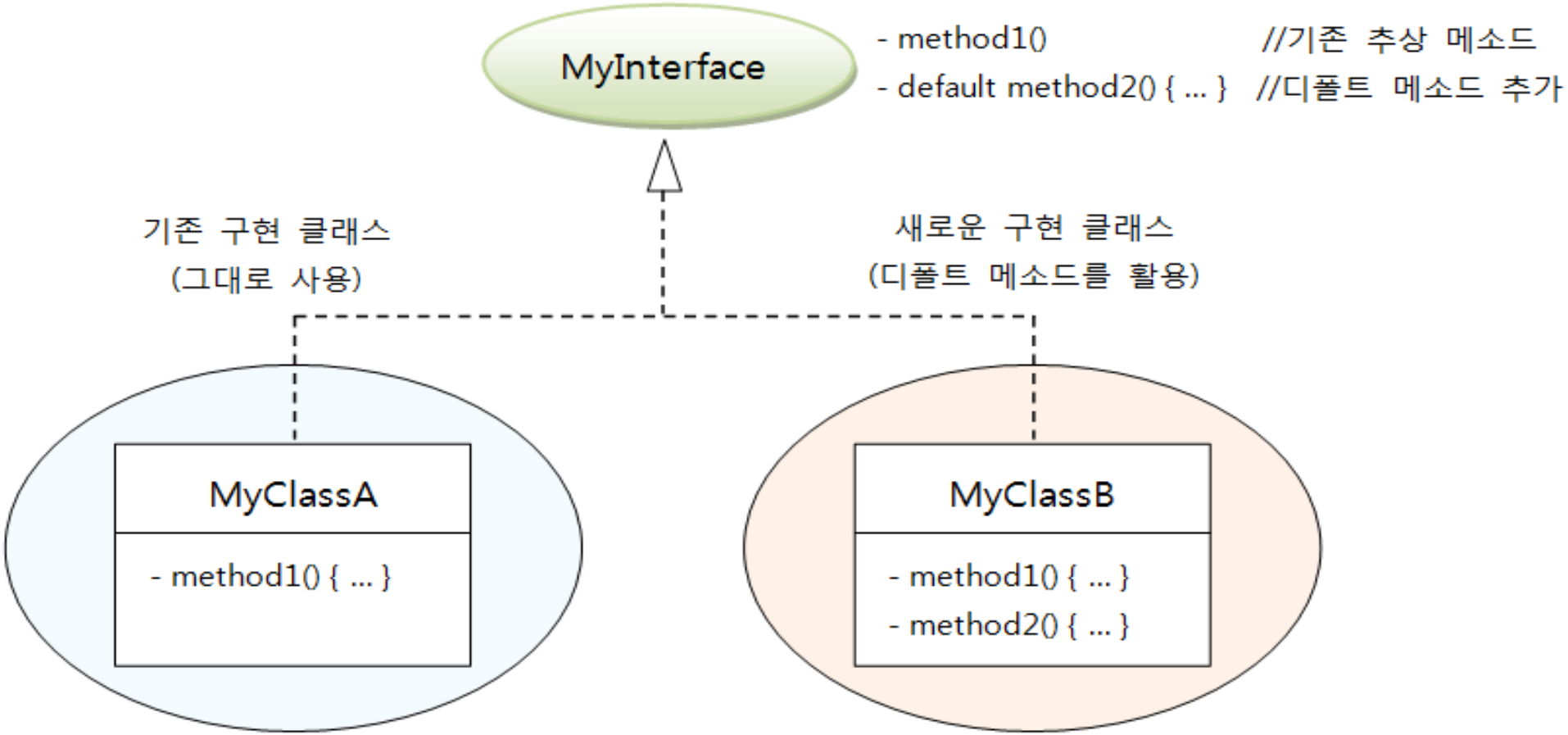
인터페이스 상속

❖ 호출 가능 인터페이스: Example.java

```
public class Example {
    public static void main(String[] args) {
        ImplementationC impl = new ImplementationC();

        InterfaceA ia = impl;
        ia.methodA();
        System.out.println();  // interfaceA 변수는 methoA()만 호출 가능

        InterfaceB ib = impl;
        ib.methodB();              // interfaceB 변수는 methoB()만 호출 가능
        System.out.println();

        InterfaceC ic = impl; // interfaceC 변수는 모든 메서드 호출 가능
        ic.methodA();
        ic.methodB();
        ic.methodC();
    }
}
```

인터페이스 상속

❖ 디폴트 메소드와 확장 메소드 사용하기



인터페이스 상속

❖ 인터페이스 : MyInterface.java

```
public interface MyInterface {
    public void method1();

    public default void method2() {
        System.out.println("MyInterface-method2 실행");
    }
}
```

인터페이스 상속

❖ 인터페이스 구현 : MyClassA.java

```
public class MyClassA implements MyInterface {
    @Override
    public void method1() {
        System.out.println("MyClassA-method1() 실행");
    }
}
```

❖ 인터페이스 구현 : MyClassB.java

```
public class MyClassB implements MyInterface {
    @Override
    public void method1() {
        System.out.println("MyClassB-method1() 실행");
    }

    @Override
    public void method2() {
        System.out.println("MyClassB-method2() 실행");
    }
}
```

❖ 디폴트 메서드 사용 : DefaultMethodExample.java

```
public class DefaultMethodExample {
    public static void main(String[] args) {
        MyInterface mi1 = new MyClassA();
        mi1.method1();
        mi1.method2();

        MyInterface mi2 = new MyClassB();
        mi2.method1();
        mi2.method2();
    }
}
```

인터페이스 상속

❖ 디폴트 메소드가 있는 인터페이스 상속

- 부모 인터페이스의 디폴트 메소드를 자식 인터페이스에서 활용 방법
  - 디폴트 메소드를 단순히 상속만 받음
  - 디폴트 메소드를 재정의(Override)해서 실행 내용을 변경
  - 디폴트 메소드를 추상 메소드로 재선언

인터페이스 상속

❖ 부모 인터페이스 : ParentInterface.java

```
public interface ParentInterface {
    public void method1();
    public default void method2() { /*실행문*/ }
}
```

❖ 자식 인터페이스 : ChildInterface1.java

```
public interface ChildInterface1 extends ParentInterface {
    public void method3();
}
```

```
ChildInterface1 ci1 = new ChildInterface1() {
    @Override
    public void method1() { /*실행문*/ }
    @Override
    public void method3() { /*실행문*/ }
};
```

```
ci1.method1();
ci1.method2();  //ParentInterface의 method2() 호출
ci1.method3();
```

인터페이스 상속

❖ 자식 인터페이스 : ChildInterface2.java

```
public interface ChildInterface2 extends ParentInterface {
    @Override
    public default void method2() { /*실행문*/ }  // 재정의

    public void method3();
}
```

```
ChildInterface2 ci2 = new ChildInterface2() {
    @Override
    public void method1() { /*실행문*/ }
    @Override
    public void method3() { /*실행문*/ }
};
```

```
ci2.method1();
ci2.method2();  //ChildInterface2의 method2() 호출
ci2.method3();
```

인터페이스 상속

❖ 자식 인터페이스 : ChildInterface2.java

```
public interface ChildInterface3 extends ParentInterface {
    @Override
    public void method2();
    public void method3();
}
```

```
ChildInterface3 ci3 = new ChildInterface3() {
    @Override
    public void method1() { /*실행문*/ }
    @Override
    public void method2() { /*실행문*/ }
    @Override
    public void method3() { /*실행문*/ }
};
```

```
ci3.method1();
ci3.method2();  //ChildInterface3 구현 객체의 method2() 호출
ci3.method3();
```

❖ 디폴트 메서드 사용 : DefaultMethodExtendsExample.java

```
public class DefaultMethodExtendsExample {
    public static void main(String[] args) {
        ChildInterface1 ci1 = new ChildInterface1() {
            @Override
            public void method1() {
                /* 실행문 */ }

            @Override
            public void method3() {
                /* 실행문 */ }
        };

        ci1.method1();
        ci1.method2(); // ParentInterface의 method2() 호출
        ci1.method3();

        // ------------------------------------------------------------
```

❖ 디폴트 메서드 사용 : DefaultMethodExtendsExample.java

```
ChildInterface2 ci2 = new ChildInterface2() {
   @Override
   public void method1() {
      /* 실행문 */ }

   @Override
   public void method3() {
      /* 실행문 */ }
};

ci2.method1();
ci2.method2(); // ChildInterface2의 method2() 호출
ci2.method3();

// ------------------------------------------------------------
```

❖ 디폴트 메서드 사용 : DefaultMethodExtendsExample.java

```
        ChildInterface3 ci3 = new ChildInterface3() {
            @Override
            public void method1() {
                /* 실행문 */ }

            @Override
            public void method2() {
                /* 실행문 */ }

            @Override
            public void method3() {
                /* 실행문 */ }
        };

        ci3.method1();
        ci3.method2(); // ChildInterface3 구현 객체의 method2() 호출
        ci3.method3();
    }
}
```