Class 클래스

❖ Class 클래스

- 클래스와 인터페이스의 메타 데이터 관리 / Reflection
- 메타데이터
  - 클래스의 이름, 생성자 정보, 필드 정보, 메소드 정보
- 런타임 때 클래스 정보 추출 및 이용

Class 클래스

❖ Class 객체 얻기(.class, getClass(), forName())

- Object.class
  - Object의 static 멤버
    - 모든 클래스는 static class 멤버를 가짐
- Object.getClass() 메서드
  - Object의 메서드
  - 인스턴스로부터 Class 객체 얻는 방법
- Class.forName()
  - 클래스명 문자열로부터 얻는 방법
  - 문자열에서 지정한 클래스가 존재하지 않는 경우 ClassNotFound 예외 발생

Class 클래스

❖ Car.java

```
public class Car {
}
```

Class 클래스

❖ ClassExample.java

```
public class ClassExample {
    public static void main(String[] args) {
        Car car = new Car();
        Class clazz1 = car.getClass();
        System.out.println(clazz1.getName());
        System.out.println(clazz1.getSimpleName());
        System.out.println(clazz1.getPackage().getName());
        System.out.println();

        try {
            Class clazz2 = Class.forName("sec06.exam01_class.Car");
            System.out.println(clazz2.getName());
            System.out.println(clazz2.getSimpleName());
            System.out.println(clazz2.getPackage().getName());
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

Class 클래스

❖ 리플렉션 (Reflection)

- 런타임 때 클래스의 생성자, 필드, 메소드 정보를 알아내는 것

❖ Car.java

```
public class Car {
    private String model;
    public String owner;

    // 생성자 추가
    // getter/setter 추가
}
```

❖ ReflectionExample.java

```
import java.lang.reflect.*;

public class ReflectionExample {
    public static void main(String[] args) throws Exception {
        Class clazz = Class.forName("Car");

        System.out.println("[클래스 이름]");
        System.out.println(clazz.getName());
        System.out.println();

        System.out.println("[생성자 정보]");
        Constructor[] constructors = clazz.getDeclaredConstructors();
        for (Constructor constructor : constructors) {
            System.out.print(constructor.getName() + "(");
            Class[] parameters = constructor.getParameterTypes();
            printParameters(parameters);
            System.out.println(")");
        }
        System.out.println();
```

❖ ReflectionExample.java

```
        System.out.println("[필드 정보]");
        Field[] fields = clazz.getDeclaredFields();
        for (Field field : fields) {
            System.out.println(field.getType().getSimpleName() + " " +
                    field.getName());
        }
        System.out.println();


        System.out.println("[메소드 정보]");
        Method[] methods = clazz.getDeclaredMethods();
        for (Method method : methods) {
            System.out.print(method.getName() + "(");
            Class[] parameters = method.getParameterTypes();
            printParameters(parameters);
            System.out.println(")");
        }
    }
```

❖ ReflectionExample.java

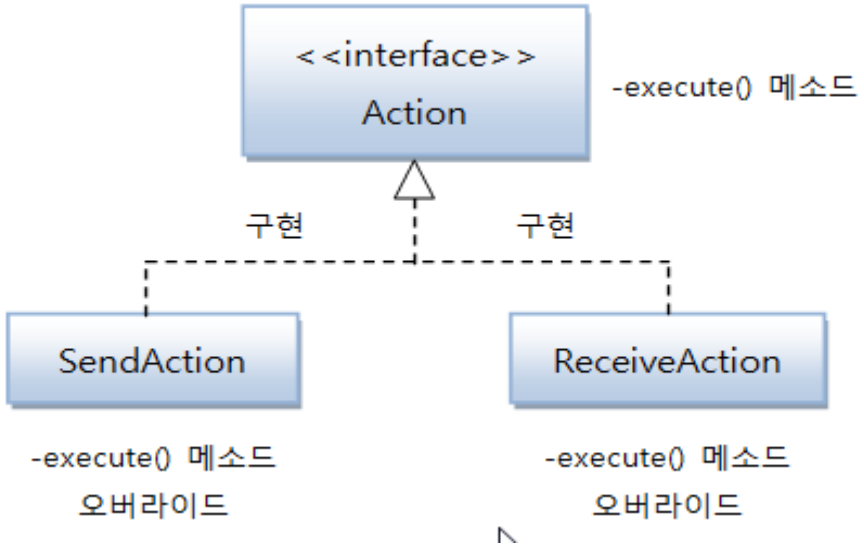
```
    private static void printParameters(Class[] parameters) {
        for (int i = 0; i < parameters.length; i++) {
            System.out.print(parameters[i].getName());
            if (i < (parameters.length - 1)) {
                System.out.print(",");
            }
        }
    }
}
```

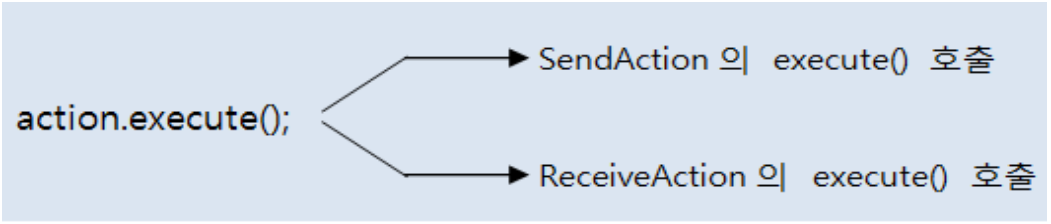
Class 클래스

❖ 동적 객체 생성(newInstance())

- ㅇ 실행 중 클래스 이름이 결정되어
  - ▪ 객체의 인스턴스를 생성

```
Class clazz = Class.forName("SendAction" 또는 "ReceiveAction");
Action action = (Action) clazz.newInstance();
action.execute();
```





Class 클래스

❖ 인터페이스 : Action.java

```
public interface Action {
    public void execute();
}
```

❖ 수신 클래스 : ReceiveAction.java

```
public class ReceiveAction implements Action {
    @Override
    public void execute() {
        System.out.println("데이터를 받습니다.");
    }
}
```

❖ 발신 클래스 : SendAction.java

```
public class SendAction implements Action {
    @Override
    public void execute() {
        System.out.println("데이터를 보냅니다.");
    }
}
```

❖ 동적 객체 생성 및 실행 : NewInstanceExample.java

```
public class NewInstanceExample {
    public static void main(String[] args) {
        try {
            Class clazz = Class.forName("SendAction");
            // Class clazz = Class.forName("ReceiveAction");

            Action action = (Action) clazz.newInstance();
            action.execute();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (InstantiationException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        }
    }
}
```