1.System 클래스

1-1. 개요

System 클래스는 유용한 필드들을 포함하고 있는데, 표준입력, 표준출력, 표준에러 스트림입니다. System 클래스는 final 로 선언되어 있어서 클래스를 인스턴스화 할 수 없습니다.(System 객체는 Static final로 이미 만들어져 있고 수정 불가능)

또한 System 클래스는 많은 수의 메소드들을 포함하고 있으며 그것들을 이용하여 외부에서 정의된 프로퍼티들이나 환경변수, 파일들과 라이브러리들을 읽거나 할 수 있습니다. 또한 System 클래스는 다른 배열로 배열의 일부를 복사하는 간단한 메소드가 포함되어 있습니다.

1-2. System class Fields

System.in

- 표준 입력 스트림

System.out

- 표준 출력 스트림

System.err

- 표준 오류 스트림

참조 : 위의 세가지 스트림들은 이미 열려 있으며 데이터에 접근할 준비가 되어있습니다.(static 이다)

1-3. System class Main Methods

1. void System.exit(종료상태값) : 종료상태값 으로 프로세스 종료.

2. void System.gc() : 쓰레기 수집기 실행요청.(but 바로 실행되지 않고, jvm이 한가해질 때 수행될 것)

3. long currentTimeMills() : 현재 시각 읽어서 long 타입 리턴.

4. String getProperty(String key) : 시스템 속성값(jvm 시작시 자동 설정되는 속성값) 문자열로 리턴.

4-1. key값의 종류 : java.version, java.home, os.name, user.name 등.

5. String getenv(String name) : 환경변수 읽기 (ex) System.getenv("JAVA_HOME");

2. Class 클래스

자바는 클래스와 인터페이스의 메타 데이터를 java.lang 패키지에 소속된 Class 클래스로 관리합니다. 여기서 메타 데이터는 클래스의 이름, 생성자 정보, 필드 정보, 메소드 정보를 말합니다.

Class 객체 얻기(getClass(), forName())

프로그램에서 Class 객체를 얻기 위해서는 Object 클래스가 가지고 있는 getClass() 메소드를 이용하면 됩니다. Object는 모든 클래스의 최상위 클래스이므로 모든 클래스에서 getClass() 메소드를 호출할 수 있습니다.

```
Class clasis = obj.getClass();

```

getClass() 메소드는 해당 클래스로 객체를 생성했을 때만 사용할 수 있는데, 객체를 생성하기 전에 직접 Class 객체를 얻을 수도 있습니다. Class는 생성자를 감추고 있기 때문에 new 연산자로 객체를 만들 수 없고, 정적 메소드인 forName()을 이용해야 합니다. forName() 메소드는 클래스 전체 이름(패키지가 포함된 이름)을 파라미터로 받고 Class 객체를 리턴합니다.

```
try {
    Class clasis = Class.forName(String className);
} catch( ClassNotFoundException e ) {

}
```


Class.forName() 메소드는 파라미터로 주어진 클래스를 찾지 못하면 ClassNotFoundException 예외를 발생시키기 때문에 예외 처리가 필요합니다. 다음은 두 가지 방법으로 Car 클래스의 Class 객체를 얻고, Class의 메소드를 이용해 클래스의 전체 이름과 간단한 이름 그리고 패키지 이름을 얻어 출력합니다.

*** ClassExam.java**

```
package api;


public class ClassExam {
    public static void main(String[] args) {
        Car car = new Car();
        Class<? extends Car> clasis1 = car.getClass();
        System.out.println(clasis1.getName());
        System.out.println(clasis1.getSimpleName());
        System.out.println(clasis1.getPackage().getName());
        System.out.println();
        try {
            Class<?> clasis2 = Class.forName("api.Car");

            System.out.println(clasis2.getName());
            System.out.println(clasis2.getSimpleName());
            System.out.println(clasis2.getPackage().getName());
        } catch(ClassNotFoundException e) {}
    }
}

class Car {

}


```


3. 리플렉션

Class 객체를 이용하면 클래스의 생성자, 필드, 메소드 정보를 알아낼 수 있습니다. 이것을 리플렉션(Reflection)이라고 합니다. Class 객체는 리플렉션을 위해 getDeclaredConstructors(), getDeclaredFields(), getDeclaredMethods() 를 제공하고 있습니다.

```
Constructor[] constructors = clasis.getDeclaredConstructors();
Field[] fields = clasis.getDeclaredFields();
Method[] methods = clasis.getDeclaredMethods();

```


Constructor, Field, Method 클래스는 모두 java.lang.reflect 패키지에 속해 있습니다. getDeclaredFields()와 getDeclaredMethods()는 클래스에 선언된 멤버만 가져오고 상속된 멤버는 가져오지 않습니다.

다음은 Car 클래스에서 선언된 생성자, 필드, 메소드의 정보를 얻고 출력합니다.

*** ReflectExam.java**

```cs
package api;


import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.Method;

public class ReflectExam {
    public static void main(String[] args) throws Exception {
        Class<?> clasis = Class.forName("api.Car");

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

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

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

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

    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(", ");
            }
        }
    }
}

```

```
53
54
55
```


실행 결과는 Car 클래스가 자체적으로 가지고 있는 public 멤버와 상위 클래스인 Object가 가지고 있는 public 멤버들이 모두 출력되고, private 멤버들은 출력되지 않는 것을 볼 수 있습니다.