열거형, 가변 인자, 어노테이션

열거형, 가변 인자, 어노테이션

- 자료형의 부여를 돕는 열거형
  - 열거형은 자바 5에서 추가된 자료형으로 "의미가 부여된 이름"을 갖는 "상수"의 선언

```
public enum Scale {
    DO, RE, MI, FA, SO, RA,TI
}
```

```
public class SimpleEnum {
    public static void main(String[] args) {
        Scale sc = Scale.DO;
        System.out.println(sc);
        switch (sc) {
            case DO:
                System.out.println("도~");
                break;
            case RE:
                System.out.println("레~");
                break;
            case MI:
                System.out.println("미~");
                break;
            default:
                System.out.println("파~솔~라~시~");
                break;
        }
    }
}
```

  - case문에서는 표현의 간결함을 위해 DO와 같이 열거형 값의 이름만 명시

열거형, 가변 인자, 어노테이션

- 클래스 내에 정의가 가능한 열거형의 정의
  - 특정 클래스 내에서만 사용하고자 하는 열거형 값이 있다면 클래스 내에 열거형을 정의할 수 있음

```
public class Customer {
    enum Gender {
        MALE, FEMALE
    }
    private String name;
    private Gender gen;
    public Customer(String name, String gen) {
        this.name = name;
        if (gen.equals("man")) {
            this.gen = Gender.MALE;
        }
        else {
            this.gen = Gender.FEMALE;
        }
    }
    @Override
    public String toString() {
        if (this.gen == Gender.MALE) {
            return "Thank you, Mr. " + this.name;
        } else {
            return "Thank you, Mrs. " + this.name;
        }
    }
}
```

```
public class InnerEnum {
    public static void main(String[] args) {
        Customer cus1 = new Customer("Brown", "man");
        Customer cus2 = new Customer("Jenny", "woman");
        System.out.println(cus1);
        System.out.println(cus2);
    }
}
```

열거형, 가변 인자, 어노테이션

- 매개변수의 가변 인자 선언과 호출
  - Set<E>에서 언급했던 "public static int hash(Object...values)" 메소드에서 "Object...values"가 가변 인자 선언
  - 가변 인자 선언 시 전달되는 인자의 수에 제한을 두지 않을 수 있음

```
public class showAll {
    public static void showAll(String...vargs) {
        System.out.println("LEN : " + vargs.length);
        for (String s : vargs) {
            System.out.println(s);
        }
    }
    public static void main(String[] args) {
        showAll("Box");
        showAll("Box", "Toy");
        showAll("Box", "Toy", "Apple");
    }
}
```

열거형, 가변 인자, 어노테이션

- 어노테이션 (Annotations)
  - 자바 컴파일러에게 메시지를 전달하는 목적의 메모
  - @Override, @Deprecated, @SuppressWarnings 어노테이션 타입들에 대한 소개

- @Override
  - 상위 클래스의 메소드 오버라이딩 또는 인터페이스에 선언된 추상 메소드의 구현

```
public interface Viewable {
    void showIt(String str);
}
```

```
public class Viewer implements Viewable {
    @Override
    public void showIt(String str) {
        System.out.println(str);
    }
}
```

```
public class AtOverride {
    public static void main(String[] args) {
        Viewable view = new Viewer();
        view.showIt("Hello Annotations");
    }
}
```

열거형, 가변 인자, 어노테이션

- @Deprecated
  - deprecated : 문제의 발생 소지가 있거나 개선된 기능의 다른 것으로 대체되어서 더 이상 필요 없게 되었음을 뜻함
  - 즉, 아직은 호환성 유지를 위해 존재하지만 이후에 사라질 수 있는 클래스 또는 메소드

```
public interface Viewable {
    void showIt(String str);
}
```

```
public class AtDeprecated {
    public static void main(String[] args) {
        Viewable view = new Viewer();
        view.showIt("Hello Annotations");
        view.brShowIt("Hello Annotations");
    }
}
```

```
public class Viewer implements Viewable {
    @Override
    public void showIt(String str) {
        System.out.println(str);
    }
    @Override
    public void brShowIt(String str) {
        System.out.println("[" + str + "]");
    }
}
```

  - 코드 컴파일 시 컴파일에 이상은 없지만 deprecated 된 무언가를 사용했음을 알리는 메시지 표시

열거형, 가변 인자, 어노테이션

- @SuppressWarnings
  - @Deprecated 어노테이션을 사용한 메소드를 대체할 수 있는 상황이 아니어서 당분간 그 메소드를 구현하고 호출해야 하는 상황에 @SuppressWarnings 어노테이션을 선언하여 경고 메시지를 지울 수 있음
  - Deprecated 된 메소드에 대한 경고를 보내지 말라고 요청하려면 "@SuppressWarnings("deprecation")" 사용

```
public interface Viewable {
    void showIt(String str);
}
```

```
public class AtSuppressWarnings {
    @SuppressWarnings("deprecation")
    public static void main(String[] args) {
        Viewable view = new Viewer();
        view.showIt("Hello Annotations");
        view.brShowIt("Hello Annotations");
    }
}
```

```
public class Viewer implements Viewable {
    @Override
    @SuppressWarnings("deprecation")
    public void showIt(String str) {
        System.out.println(str);
    }
    @Override
    public void brShowIt(String str) {
        System.out.println("[" + str + "]");
    }
}
```