람다(Lambda)의 소개

람다(Lambda)의 소개

- 람다의 이해
  - 람다를 사용하면 코드를 줄일 수 있으며 이렇게 만들어진 코드는 가독성도 뛰어나지만 람다에 익숙해져야 함

```
public interface Printable {
    void print(String s);
}
```

```
public class Printer implements Printable {
    @Override
    public void print(String s) {
        System.out.println(s);
    }
}
```

```
public class Lambda1 {
    public static void main(String[] args) {
        Printable prn = new Printer();
        prn.print("What is Lambda?");
    }
}
```

  - 위 코드에 익명 클래스를 적용하면 클래스의 정의가 하나 줄어듬

```
public interface Printable {
    void print(String s);
}
```

```
public class Lambda2 {
    public static void main(String[] args) {
        Printable prn = new Printable() {
            @Override
            public void print(String s) {
                System.out.println(s);
            }
        };
        prn.print("What is Lambda?");
    }
}
```

람다(Lambda)의 소개

- 익명 클래스의 정의를 람다식으로 변경(람다와 익명 클래스의 내부적인 동작 원리는 다름)

```
public interface Printable {
    void print(String s);
}
```

```
public class Lambda3 {
    public static void main(String[] args) {
        Printable prn = s -> { System.out.println(s); }
        prn.print("What is Lambda?");
    }
}
```

- 위 코드에서 대입 연산자의 오른쪽 "s -> System.out.println(s)"가 람다식(Lambda Expression)
- 람다와 익명클래스는 다른 방식으로 동작하지만 둘 다 인스턴스의 생성으로 이어지고, 람다식이 익명 클래스의 정의를 일부 대체하기 때문에 익명 클래스의 정의를 기반으로 람다식을 이해하는 것이 좋은 방법

람다(Lambda)의 소개

```
Printable prn = new Printable() {
    @Override
    public void print(String s) {
        System.out.println(s);
    }
};
```

- 위 문장에서 = 연산자 왼쪽이 Printable형 참조 변수이기 때문에 "new Printable()"이 없어도 유추 가능
- 또한 Printable 인터페이스에는 추상 메소드가 하나만 존재하므로 메소드 선언 부분인 "public void print(String s)"가 없어도 유추 가능

```
Printable prn = { System.out.println(s); }
```

- 위 문장에서 s가 매개변수라는 사실까지는 컴파일러가 유추하지 못함
- 따라서 매개변수에 대한 정보를 남기며 더불의 람다식의 구분을 위해 ->(람다 연산자)를 추가하여 람다식 표현

```
Printable prn = (String s) -> { System.out.println(s); }
```

- 위 문장에서 Printable 인터페이스의 추상 메소드를 통해 매개변수 s가 String이라는 것을 유추할 수 있으므로 이를 생략하여 최종 형태의 람다식 표현

```
Printable prn = s -> { System.out.println(s); }
```

람다(Lambda)의 소개

- 람다식의 인자 전달
  - "int n = 10;"과 같이 변수를 선언하고 초기화 할 수 있음
  - 이와 동일한 원리로 "method(10)"과 같이 인자를 전달하여 매개변수를 초기화할 수 있음
  - 마찬가지로 "Printable prn = s -> { System.out.println(s) }"와 같이 참조변수를 초기화 할 수 있음
  - 즉, "method(s -> System.out.println(s))"와 같이 람다식을 메소드의 인자로 전달할 수도 있음

```
public interface Printable {
    void print(String s);
}
```

```
public class Lambda4 {
    public static void ShowString(Printable p, String s) {
        p.print(s);
    }
    public static void main(String[] args) {
        ShowString(s -> { System.out.println(s); }, "What is
Lambda?");
    }
}
```

  - 람다식을 인자로 전할 때의 형태는 "Printable p = s -> { System.out.println(s); }"의 형태이므로 람다식을 인자로 전달 가능