제5장 : 람다식과 함수, 스트림

1급 객체

변수에 대입 가능한 객체를 1급 객체 (first class object) 라고 한다.

대표적인 1급 객체 : 값, 인스턴스, 배열

함수 (function)

Java8 에 추가된 개념으로 함수도 1급 객체로 취급 됨

입력을 처리하여 출력하는 것

$$y = 2x + 3$$

메서드를 변수에 대입하여 사용하는 예

입출력 타입만 같다면 메서드를 변수에 대입하는 것이 가능

```
class Main {
  public static int add(int x, int y) {
    return x + y;
  }

  public static void main(String[] args) {
    IntBinaryOperator func = Main::add;

    int result = func.applyAsInt(5, 3);
    System.out.println("5 + 3 = " + result);
  }
}
```

메서드와 함수의 차이

메서드는 클래스에 속하고 클래스를 조작하기 위한 일종의 함수

메서드는 이름이 있지만.

```
public static int add(int x, int y) {
  return x + y;
}
```

함수에게 이름은 중요치 않다

함수 저장용 인터페이스 선언

추상 메서드가 1개인(SAM : Single Abstract Method) 인터페이스는 함수 저장용으로 사용 가능

```
interface MyFunction {
  public abstract int call(int a, int b);
}

class Main {
  public static int add(int x, int y) {
    return x + y;
  }

  public static void main(String[] args) {
    MyFunction func = Main::add;

    int result = func.call(5, 3);
    System.out.println("5 + 3 = " + result);
  }
}
```

함수 저장용 범용 API

함수를 저장하기 위해 매번 SAM 인터페이스를 만드는 것은 귀찮음

따라서 Java에서는 자주 사용할 것 같은 함수 저장용 범용 API를 준비 해 놨음

대표적인 SAM 인터페이스 (함수형 인터페이스라고도 함)

함수형 인터페이스	Descriptor	Method명
Predicate<T>	T -> boolean	test()
Consumer<T>	T -> void	accept()
Supplier<T>	() -> T	accept()
Function<T,R>	T -> R	apply()
UnaryOperator<T>	T -> T	identity()

람다식 (lambda expression)

함수 내용을 바로바로 정의해서 사용하고 싶다~!!

```
interface MyFunction {
  public abstract int call(int a, int b);
}

class Main {

  public static void main(String[] args) {
    MyFunction func = (int a, int b) -> {
      return a + b;
    };

    int result = func.call(5, 3);
    System.out.println("5 + 3 = " + result);
  }
}
```

람다식의 표기법


```
IntToDoubleFunction func = (int a) -> {
  return a * 3.14;
};
```

```
IntToDoubleFunction func = (a) -> {
  return a * 3.14;
};
```

← 입력 타입 생략 가능

```
IntToDoubleFunction func = a -> {
  return a * 3.14;
};
```

← 입력값이 1개일 때 소괄호 생략 가능

```
IntToDoubleFunction func = a -> a * 3.14;
```

← 함수의 내용이 리턴문 밖에 없을 대 중괄호, return 키워드 생략 가능

함수를 값으로 취급할 때의 이점은??

Java8에 추가된 함수를 다루기 위한 범용 API인 Stream을 사용하면 좀 더 함수적인 사고를 가지고 개발할 수 있다.
함수형 프로그래밍

```
for (Integer i : list) {
  System.out.println(i);
}

list.stream().forEach(i -> System.out.println(i));

list.stream().forEach(System.out::println);
```

연습문제 5-1

다음 코드에 작성되어 있는 두 개의 static 메서드를 함수로서 변수에 담고, 그것을 호출하는 프로그램을 작성하시오.
이 때 함수를 대입하기 위해 필요한 Func1, Func2 인터페이스를 정의하시오.
메서드 이름이나 인수 이름은 자유롭게 정해도 됩니다.

```
1   public class Utils {
2
3     public static boolean isOdd(int n) {
4       return n % 2 == 1;
5     }
6
7     public static addNamePrefix(boolean male, String name) {
8       if (male == true) {
9         return "Mr." + name;
10      }
11      return "Ms." + name;
12    }
13  }
```

연습문제 5-2

연습문제 5-1의 Utils 클래스의 2가지 메서드와 동일한 내용을 람다식으로 표현하여 각각 Func1, Func2 타입 변수에 대입하는 문장을 작성하시오.