Lambda Expression 학습

학습내용

1. Lambda Expression의 개념
2. Lambda Expression의 활용

학습목표

- Lambda Expression의 개념을 설명할 수 있다.
- Lambda Expression의 기본적인 활용을 할 수 있다.

Lambda Expression

Java 8 이후의 **함수형 프로그래밍 지원**

수학적 함수를 표현하는 방법

기존과 다른 프로그래밍 패러다임을 지향

Lambda Expression

명령형 프로그래밍

연산 과정을 통해 프로그램 상태를 변경시키는 과정에 집중

함수형 프로그래밍

연산을 함수로 정의, 조합하는 것에 집중

Lambda Expression

명령형 프로그래밍 예시

```
if (!occurrence.containsKey(from)) {
        occurrence.put(from, 1);
} else {
        int previousValue = occurrence.get(from);
        occurrence.put(from, previousValue + 1);
}
```

from Key의 기존의 값을 가져와
1을 추가하여 삽입

Lambda Expression

함수형 프로그래밍 예시

```
occurrence.compute(from, (key,value) -> {
  if (value == null)
    return 1;
  else {
    return value + 1;
  }
});
```

occurrence update란 key에 대한
value를 1 증가시키는 것

Lambda Expression 구문

매개변수 화살표 함수 구현의 세 가지 구문으로 이뤄짐

Lambda Expression 구문

함수 1: 3개의 수 더한 결과 반환

```
(int x, int y, int z) -> {
        return x + y + z;
};
```

함수 2 : 3개의 수 콘솔 출력

```
(int x, int y, int z) -> {
        System.out.println(x+y+z);
};
```

Lambda Expression 사용법

```
(int x, int y, int z) -> {
            return x + y + z;
} (3,4,5);
```

[유사 LISP 구문]

Lambda Expression 사용법

함수형 인터페이스(Functional Interface)의 구현 1

```
ComputeThreeIntegers sum = (int x, int y, int z) -> {
            return x + y + z;
};
```

인터페이스 타입의 변수로 할당

함수형 인터페이스(Functional Interface)의 구현 2

```
public void print(ComputeThreeIntegers sum, int x, int y, int z) {
            sum.compute(x,y,z);
}

print((x, y, z) -> x + y + z, 3, 4, 5);
```

Method의 매개변수로 전달

[실습 1]
기존 방법

ComputeThreeIntegers Interface 설계

Interface를 구현한 Class 설계

Class의 instance 생성

Instance 활용

[실습 2]
Lambda Expression 활용 방법

ComputeThreeIntegers Functional Interface 설계

Lambda Expression으로 Interface 구현

Lambda Expression 사용법

다양한 축약이 가능

```
ComputeThreeIntegers sum = (int x, int y, int z) -> {
                return x + y + z;
};
```

매개변수 타입 생략(추론 가능)

```
ComputeThreeIntegers sum = (x, y, z) -> {
                return x + y + z;
};
```

단일 라인일 때 { } 생략 가능(반환 값)

```
ComputeThreeIntegers sum = (x, y, z) -> x + y + z;
```

Lambda Expression 사용법

다양한 축약이 가능

```
ConsumeThreeIntegers print = (int x, int y, int z) -> {
        System.out.println(x+y+z);
};
```

매개변수 타입 생략(추론 가능)

```
ConsumeThreeIntegers print = (x, y, z) -> {
        System.out.println(x+y+z);
};
```

단일 라인일 때 { } 생략 가능(수행 됨)

```
ConsumeThreeIntegers print = (x, y, z) -> System.out.println(x+y+z);
```

Lambda Expression 사용법

다양한 축약이 가능

```
(int r) -> {
            return 3.14f * r * r;
};
```

매개변수 타입 생략(추론 가능)

```
(r) -> {
            return 3.14f * r * r;
};
```

단일 라인일 때 { } 생략 가능(반환 됨)

```
(r) -> 3.14f * r * r;
```

단일 매개변수일 때 ( ) 생략 가능

```
r -> 3.14f * r * r;
```

Lambda Expression 사용법

빈 매개변수에는 ( ) 필요

```
() -> //DO or RETURN SOMETHING;
```

Lambda Expression

축약형

Remind

Lambda Expression