10주차 실습 보고서

강의명	객체지향프로그래밍및실습
담당교수	류기열
학과 학년	소프트웨어학과 2학년
학번	202220209
작성자	이욱준

2025.11. 6.

□ 1

(가)

```
interface Square{
    double sqr(double res);
}

double x;
Square s = (x) -> x*x;
```

(나)

```
import java.time.*;

interface Compare{
    boolean cmp(LocalDate date, int x);
}

LocalDate date;
double x;
Compare c = (date, x) -> date.getYear() < x;
```

(다)

```
interface Compare{
    String cmp(String date, String x);
}

String x;
String y;
Compare c = (x,y)->{
    if(x.length() > 5)
        return x.substring(0,5);
};
```

□ 2

(가)

```
lwj@lwj-code:~/workspace/JAVA/10$ javac EmployeeSortTest.java
EmployeeSortTest.java:17: error: no suitable method found for sort(List<Employee>)
        Collections.sort(elist);
                    ^
    method Collections.<T#1>sort(List<T#1>) is not applicable
      (inference variable T#1 has incompatible bounds
        equality constraints: Employee
        lower bounds: Comparable<? super T#1>)
    method Collections.<T#2>sort(List<T#2>,Comparator<? super T#2>) is not applicable
      (cannot infer type-variable(s) T#2
        (actual and formal argument lists differ in length))
  where T#1,T#2 are type-variables:
    T#1 extends Comparable<? super T#1> declared in method <T#1>sort(List<T#1>)
    T#2 extends Object declared in method <T#2>sort(List<T#2>,Comparator<? super T#2>)
1 error
```

Collections.sort()는 List<Employee>형에 대해 사용될 수 없기 때문에 오류가 발생한다. Employee 객체 간의 Equaility를 측정하는 방식이 정의되어있지 않아서 발생한다.

(나)

sort함수에 비교 조건을 lambda expression을 통해 추가한다.

JAVA > 10 > EmployeeSortTest.java > EmployeeSortTest > main(String[])

```java
import java.util.*;

// 2-(가): 아래 EmployeeSortTest 클래스에는 어떤 컴파일 문제가 있는지 설명하시오.
// ( EmployeeSortTest.java는 Employee 객체를 급여(salary) 오름차순으로 정렬하는 프로그램이다. )

// 2-(나): (가)번의 문제를 해결하기 위해 Comparator Interface를 구현하는 lambda expression을 사
// 어디를 어떻게 수정하면 되는지 코드를 첨부하시오. [ 단, Employee 클래스는 수정할 수 없다. ]

public class EmployeeSortTest {
    public static void main(String[] args) {
        List<Employee> elist = new ArrayList<>();
        elist.add(new Employee(n: "Lee", s: 10000));
        elist.add(new Employee(n: "Kim", s: 20000));
        elist.add(new Employee(n: "Park", s: 8000));
        elist.add(new Employee(n: "Han", s: 15000));

        Collections.sort(elist, (Employee e1, Employee e2)->e1.getSalary()-e2.getSalary());
        System.out.println(elist);
    }
}
```

PROBLEMS 54 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
lwj@lwj-code:~/workspace/JAVA/10$ javac EmployeeSortTest.java
lwj@lwj-code:~/workspace/JAVA/10$ java EmployeeSortTest
[Park:8000, Lee:10000, Han:15000, Kim:20000]
```

(다)

JAVA > 10 > HowManyEmployeeTest.java > HowManyEmployeeTest > main(String[])

```java

import java.util.function.Predicate;

public class HowManyEmployeeTest {
    public static void main(String[] args) {
        Employee[] emps = new Employee[3];

        emps[0] = new Employee(n: "Kim", s: 10000);
        emps[1] = new Employee(n: "Lee", s: 20000);
        emps[2] = new Employee(n: "Kim", s: 30000);

        int n = HowManyEmployeeTest.howMany(emps, e -> e.getSalary() >= 20000);
        int m =  HowManyEmployeeTest.howMany(emps, e -> e.getName() == "Kim");

        System.out.println("Number of High - salaried Employees = "+n);
        System.out.println("Number of Employees named Kim = "+m);
    }

```

PROBLEMS 52 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
lwj@lwj-code:~/workspace/JAVA/10$ javac HowManyEmployeeTest.java
lwj@lwj-code:~/workspace/JAVA/10$ java HowManyEmployeeTest
Number of High - salaried Employees = 2
Number of Employees named Kim = 2
```

3

JAVA > 10 > ExpressionTest.java > ExpressionTest > main(String[])

```
}

public class ExpressionTest {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        char op = in.next().charAt(index: 0);
        int x = in.nextInt();
        int y = in.nextInt();
        Expression exp = null;

        switch (op) {
            // 3-(A): op에 따라 exp 변수에 적절한 lambda expression을 지정하는 코드를 작성하시오.
            case '+': exp = (a,b)->a+b; break;
            case '-': exp = (a,b)->a-b; break;
            case '*': exp = (a,b)->a*b; break;
        }
        int result = f(exp, x,y);  // 3-(B): (A)의 exp을 연산하는 코드를 작성하여 연산 결과를 result에 반환하시오.
        System.out.println("result = "+ result);
    }
    public static int f(Expression e, int a, int b) {
        return e.eval(a, b);
    }
}
```

PROBLEMS 51 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
lwj@lwj-code:~/workspace/JAVA/10$ javac ExpressionTest.java
lwj@lwj-code:~/workspace/JAVA/10$ java ExpressionTest
+ 2 3
result = 5
lwj@lwj-code:~/workspace/JAVA/10$ java ExpressionTest
- 3 2
result = 1
lwj@lwj-code:~/workspace/JAVA/10$ java ExpressionTest
* 6 5
result = 30
lwj@lwj-code:~/workspace/JAVA/10$
```

□ 코드 전문

EmployeeSortTest.java

```
import java.util.*;

// 2-(가): 아래 EmployeeSortTest 클래스에는 어떤 컴파일 문제가 있는지 설명하시오.
// ( EmployeeSortTest.java 는 Employee 객체를 급여(salary) 오름차순으로 정렬하는 프로그램이다. )

// 2-(나): (가)번의 문제를 해결하기 위해 Comparator Interface 를 구현하는 lambda expression 을 사용할 수 있다.
// 어디를 어떻게 수정하면 되는지 코드를 첨부하시오. [ 단, Employee 클래스는 수정할 수 없다. ]

public class EmployeeSortTest {
    public static void main(String[] args) {
        List<Employee> elist = new ArrayList<>();
        elist.add(new Employee("Lee", 10000));
        elist.add(new Employee("Kim", 20000));
        elist.add(new Employee("Park", 8000));
        elist.add(new Employee("Han", 15000));

        Collections.sort(elist, (Employee e1, Employee e2)->e1.getSalary()-
e2.getSalary());
        System.out.println(elist);
    }
}
```

HowManyEmployeeTest.java

```
import java.util.function.Predicate;

public class HowManyEmployeeTest {
    public static void main(String[] args) {
        Employee[] emps = new Employee[3];

        emps[0] = new Employee("Kim", 10000);
        emps[1] = new Employee("Lee", 20000);
        emps[2] = new Employee("Kim", 30000);

        int n = HowManyEmployeeTest.howMany(emps, e -> e.getSalary() >=
20000);    // 2-(다)-(a): (emps, _____) 을 채우시오.
        int m =  HowManyEmployeeTest.howMany(emps, e -> e.getName() ==
"Kim");   // 2-(다)-(b): (emps, _____) 을 채우시오.
```

```
        System.out.println("Number of High - salaried Employees = "+n);
        System.out.println("Number of Employees named Kim = "+m);
    }

    public static int howMany(Employee[] emps, Predicate<Employee> t) {
    int n = 0;
    for (Employee e: emps) {
        if (t.test(e)) {
            n++;
        }
    }
    return n;
    }
}
```

ExpressionTest.java

```
import java.util.*;

interface Expression {
    int eval(int a, int b);
}

public class ExpressionTest {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        char op = in.next().charAt(0);
        int x = in.nextInt();
        int y = in.nextInt();
        Expression exp = null;

        switch (op) {
            // 3-(A): op 에 따라 exp 변수에 적절한 lambda expression 을 지정하는
코드를 작성하시오.
            case '+': exp = (a,b)->a+b; break;
            case '-': exp = (a,b)->a-b; break;
            case '*': exp = (a,b)->a*b; break;
        }
        int result = f(exp, x,y);  // 3-(B): (A)의 exp 을 연산하는 코드를 작성하여
연산 결과를 result 에 반환하시오.
        System.out.println("result = "+ result);
    }
    public static int f(Expression e, int a, int b) {
        return e.eval(a, b);
    }
}
```