

LEARNING OUTCOME

Adapt functional programs to solve computing problems using functional and logic programming language concepts. (P6, PLO3)

OVERVIEW

Programming paradigms can be defined as programming styles in problem solving. In software programming languages, there are distinct programming paradigms and a set of programming concepts used in the platforms. In practice, imperative programming like procedural or object-oriented paradigms are widely used in various programming platforms. Likewise, the declarative programming such as functional programming and logic programming paradigms can also be available in modern programming languages. In this assignment, therefore, focuses on programming paradigms primarily imperative- and declarative- programming paradigms for a specific problem solving.

REQUIREMENTS

Given the following computing solution code in Task 1 and Task 2. They are mostly implemented imperatively so you are required to test the program to understand how it works and rewrite them using declarative programming approach.

TASK 1:

Consider the following order information in the given file. Filename: `deliveryitem.txt`.

PARCELITEM\WEIGHT\DELIVERYTIME\RECIPIENT\POSTINGADDR\CONTACT\CASH PIZZAHUT\500GRAM\12:00PM\STEPHEN\APU AT TPM\0389900000\29.90 KFC\400GRAM\2:00PM\THOMAS\SRIPETALING\038998888\59.90 MACDONAL\300GRAM\8:00PM\MICKEY\SETAPAK\0341414141\59.80 CHICKKENCHOP\250GRAM\7:30PM\JOHNSON\BUKITJALIL\0387878787\19.80 STEAK\150GRAM\8:30PM\HELEN\SERDANG\0322223333\29.80

Analyse the imperative program code as shown in Listing 1. Adapt each of the programming methods using functional programming concepts. Your solution code should consider higher-order function, method references, functional interfaces or lambda expression, function pipelining, nullable object, collection of objects and etc.

Listing 1:- Imperative programming paradigm – code sample:

```

import java.io.IOException;
import java.nio.file.*;
import java.util.*;
import java.util.regex.Pattern;

public class OrderHandler {

    private static final String FILENAME = "data/deliveryitem.txt";
    List<String> data;

    public OrderHandler() {
        try {
            data = Files.readAllLines( Paths.get( FILENAME ) );
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    //display all order information except the heading text
    void print( List<List<String>> orderLst ) {
        for (int i = 0; i < orderLst.size(); i++) {
            System.out.println( orderLst.get(i) );
        }
    }

    //collect all orders and split them with '\'
    List<List<String>> collect(){
        List<List<String>> orderLst = new ArrayList();
        for (int i = 0; i < data.size(); i++) {
            if( !data.get(i).startsWith("PARCELITEM") ) {
                String[] split = data.get(i).split( Pattern.quote("\\") )
);
                orderLst.add( Arrays.asList( split ) );
            }
        }
        return orderLst;
    }

    //compute the total payment of order placed
    double computePaymnt( List<List<String>> orderLst ) {
        double sum = 0;
        for (int i = 0; i < orderLst.size(); i++) {
            int count = orderLst.get(i).size();
            double charge = Double.parseDouble( orderLst.get(i).get( count-1
) );
            sum += charge;
        }
        return sum;
    }

    //export to order object comprises name and charges only
    List<Item> populate( List<List<String>> orderLst ){
        List<Item> items = new ArrayList();
        for (int i = 0; i < orderLst.size(); i++) {
            int count = orderLst.get(i).size();
            String first = orderLst.get(i).get(0);
            String last = orderLst.get(i).get( count-1 );
            items.add( new Item( first, last ) );
        }
        return items;
    }
}

```

Marks Allocation:

Solution code	Marks
<i>print()</i> - Display all order information from the given file.	3 marks
<i>collect()</i> - Filter the heading text, collect all order records and split them using a character of '\ ' into a list of collections.	5 marks
<i>computePayment()</i> - Compute the total of payment for all orders.	5 marks
<i>populate()</i> - Obtain order item name and price only and store them in Item objects.	7 marks
Total:	20 marks

TASK 2:

Analyse the following Java code in Figure 1 and 2. Rewrite the given program code using logic programming paradigm, Prolog. Your solution should include recursive concept and demonstrate the query sample.

Part 1:

```
import java.util.*;
import java.util.stream.Collectors;

public class Filter {
    public static List<Number> apply(
        List<Number> lst, Double target){
        return lst.stream()
            .mapToDouble( Number::doubleValue )
            .filter( elem -> elem > target )
            .boxed()
            .collect( Collectors.toCollection(
                ArrayList::new ) );
    }
    public static void main(String[] args) {
//      Integer[] nums = new Integer[] {1,2,3,4,5,6,7,8,9};
      Double[] nums = new Double[] {2.1,3.2,4.3,5.4,6.5,7.6,8.7};
      System.out.println(
          Filter.apply( Arrays.asList(nums), 5.0 )
      );
    }
}
```

Figure 1: Filter the numeric elements based on target value

Part 2:

```

import java.util.*;
import java.util.stream.*;

public class Demo {
    public static void main(String[] args) {
        String[][] lst = {
            { "a", "b" }, { "c", "d", "e", "f" }
        };
        Stream<String[]> stream = Arrays.stream(lst);
        System.out.println( stream
            .flatMap( arr -> Arrays.stream(arr) )
            .collect( Collectors.toCollection( LinkedList::new ) )
        );
    }
}

```

Figure 2: Joining the elements from the array objects as a collection of objects

Marks Allocation:

Criteria	Marks
Part 1:	10 marks
- Base rule, recursive rule, list (head & tail), input argument list	8 marks
- Query and output sample	2 marks
Part 2:	10 marks
- Base rule, recursive rule, list (head & tail), appropriate condition	8 marks
- Query and output sample	2 marks
Total:	20 marks

DELIVERABLES

- A report comprising of the following elements:
 - i. Introduction
 - ii. Solution for Task 1 and 2
 - a. Source code
 - b. Screenshot of the running program
 - c. Explanation of techniques used in your solution
 - iii. Conclusion
 - iv. Reference
- All related program files (source code)

SUBMISSION INSTRUCTION

- You will be given a week to complete the assignment. The due date of the submission will be 7 days from the day that you received this assignment.
- Send a copy of your assignment to the lecturer and administrator in-charge of the redo assignment.

----- END OF DOCUMENT -----