วัตถุประสงค์

A. ทบทวนการอ่าน .csv ไฟล์

B. ทดลองเขียน java functional programming ด้วย java stream

กิจกรรมที่ 1

AAAAAAAA

jvm รับ parameter ต่อท้ายการเรียก lab10_xxyyyy (ให้ทำงาน) ในตัวแปรชื่อ args โดยเก็บเป็น String [ ]

BBBBBBBB

เราสามารถสกัดแต่ละคอลัมน์ของข้อมูล .csv จากไฟล์ด้วย Scanner และ .split()

(หมายเหตุ .trim() เอาไว้ตัด whitespace หน้าและหลังสตริง)

CCCCCCCC

.csv จริงๆแยกแต่ละคอลัมน์ไว้ด้วย “” ดังนั้นหาก String ใดมี , เป็นส่วนหนึ่งและเราไม่ต้องการให้ .spit() เก่งขึ้นตัดคอลัมน์ถูกต้อง ให้ใช้ regular expression ที่ให้ไป (จาก 4 tokens ซึ่งผิด เป็น 3 tokens ซึ่งถูกต้อง)

```
// java lab10_xxyyyy 5 haha
public static void main(String [] args) {
    //args[0] is "5" and args[1] is "haha"
    testbed(args);
}
private static void testBed(String[] args) {
    println("**AAAAAAAA**");
    // learn behavior of args
    print("command arguments are ");
    for (String s : args)
        println(s.trim());
    println();

    println("**BBBBBBBB**");
    // review openning .csv file using .split()
    String row;
    String [] tokens;
    try(Scanner input = new
Scanner(Paths.get("pack10_CSMovie/samples10.csv"))) {
      input.nextLine(); //skip header row
      while (input.hasNext()) {
          row = input.nextLine();
          tokens = row.split(",");
          for (String token : tokens)
                print(token + " ");
          println();
      }
    } catch (IOException e) {
       println("from IO error");
       e.printStackTrace();
    }
    println("**CCCCCCCC**");
    //test what if title contains comma(,)
    row = "\"This is, a sample title\", \"Horror\", \"10.0\"";
    tokens = row.split(",");
    println("There are " + tokens.length
                                  + " tokens");
    for (String token : tokens)
        println(token.trim() + " ");

    tokens =
row.split(",(?=(?:[^\"]*\"[^\"]*\")*[^\"]*$)", -1);
    println("There are " + tokens.length
                                   + " tokens");
    for (String token : tokens)
      println(token.trim() + " ");
}
```

กิจกรรมที่ 2

2.1

จาก dataset เราจะ skip คอลัมน์ released และ writer

และ เก็บ movie ที่ title ไม่ซ้ำ (เก็บเฉพาะ movie ที่ title ไม่ปรากฏ)

```
try(Scanner input = new Scanner(Paths.get(first:"pack10_CSMovie/movies.csv"))) {
    input.nextLine(); //skip header row
    while (input.hasNext()) {
        row = input.nextLine();
        rowCount++;
        String[] tokens = row.split(regex:",(?=(?:[^\"]*\"[^\"]*\")*[^\"]*$)", -1);
        if (tokens.length < 15) {
            incompleteCount++;
            System.out.println(rowCount + " " + incompleteCount + " is incompleted");
            continue; //skip this row
        }
        title = tokens[0];
        rating = tokens[1];
        genre = tokens[2];
        year = Integer.parseInt(tokens[3]);
        score = Double.parseDouble(parseDouble(tokens[5]));
        votes = (int) (Double.parseDouble(parseDouble(tokens[6])));
        director = tokens[7];
        star = tokens[9];
        country = tokens[10];
        budget = (int) (Double.parseDouble(parseDouble(tokens[11])));
        gross = (int) (Double.parseDouble(parseDouble(tokens[12])));
        company = tokens[13];
        runtime = (int) (Double.parseDouble(parseDouble(tokens[14])));
        if (!uniqueTitle.contains(title)) {
            mList.add(new CSMovie(title, rating, genre,
            year, score, votes,
            director, star, country,
            budget, gross, company,
            runtime));
        }
        uniqueTitle.add(title);
    }
    System.out.print("There are " + incompleteCount + " rows of incompleted data ");
    System.out.println("from " + rowCount + " rows. (" + uniqueTitle.size()+ ") unique titles.");
    System.out.println("list size is " + mList.size());
} catch (IOException e) {
    System.out.println(x:"from IO error");
    e.printStackTrace();
}
```

```
public class MovieCounter {
    ArrayList<CSMovie> mList = new ArrayList<>();
    Set<String> uniqueTitle = new HashSet<>();
    public MovieCounter() {
        String row;
        int rowCount = 1;
        int incompleteCount = 0;
        String title;
        String rating;
        String genre;
        Integer year;
        String skipped_released;
        Double score;
        Integer votes;
        String director;
        String skipped_writer;
        String star;
        String country;
        Integer budget;
        Integer gross;
        String company;
        Integer runtime;
```

และ ใช้ private static String parseDouble(String str) ในการใส่ค่า default กรณีที่ token นั้นๆเป็น missing data

```
private static String parseDouble(String str) {
    if (str.isEmpty())
        return ".0";
    return str;
}
```

2.2 q10 และ q11 ใช้ hint ดังต่อไปนี้

```
private static void hintQ10() {
    Map<String, Integer> unordered = new HashMap<>();
    unordered.put(key:"A",value:12);
    unordered.put(key:"C",value:7);
    unordered.put(key:"B",value:20);

    Map<String, Integer> orderByValueMap;
    orderByValueMap = unordered.entrySet().stream()
                                .sorted(Entry.comparingByValue())
                                .collect(Collectors.toMap(
                                    Entry::getKey, Entry::getValue,
                                    (e1,e2) -> e1, LinkedHashMap::new
                                ));
                                //add least value one by one, hence sorted
    for (Entry entry : orderByValueMap.entrySet())
        System.out.println(entry.getKey() + " " + entry.getValue());

    Map<String, Long> unorderedLong = new HashMap<>();
        unorderedLong.put(key:"D",value:12L);
        unorderedLong.put(key:"E",value:7L);
        unorderedLong.put(key:"F",value:20L);
    Map<String, Long> longAndReverseMap;
    longAndReverseMap = unorderedLong.entrySet().stream()
                                .sorted(Collections.reverseOrder(Entry.comparingByValue(Long::compareTo)))
                                .collect(Collectors.toMap(
                                  Entry::getKey, Entry::getValue,
                                  (e1,e2) -> e1, LinkedHashMap::new
                                ));
    for (Entry entry : longAndReverseMap.entrySet())
        System.out.println(entry.getKey() + " " + entry.getValue());
}
```

```
private static void hintQ11() {
    // count number of words in title
    Function<String, Integer> numWords = entry -> {
        String [] tokens = entry.split(regex:" ");
        return tokens.length;
    };

    List<String> data = Arrays.asList(...a:"one two three",
                                      "four five", "six seven eight nine");
    Optional<String> opt = data.stream()
                                .max(Comparator.comparing(numWords));
    System.out.println(opt.get());
}
```

กิจกรรมที่ 3

Q1 แสดง average score

Q2 แสดง movie ที่ votes > 1_900_000

Q3 แสดง movie ที่ได้ gross สูงที่สุด

Q4 แสดง genre ที่มีใน dataset

Q5 ใช้ .map(e -> String.format("%-55s --> %s",e.getTitle(), e.getRuntime()))

แสดง movie title และ runtime ที่มี runtime น้อยที่สุด 5 อันดับ

Q6 แสดง title และ budget ของหนังที่ใช้ budget มากที่สุด และ น้อยที่สุด

Q7 แสดง top 3 movie by genre โดยเรียงตาม score (descending (มากขึ้นก่อน))

Q8 ใช้ .reversed() และ .thenComparing()

แสดง top 3 action movie เรียงตาม score (descending) หากเสมอกันให้ใช้ title (ascending)

Q9 ใช้ .summingLong()

แสดงผลรวมของรายได้ แยกตาม genre

Q10 แสดง 10 อันดับ company ที่ผลิต movie มากที่สุด พร้อมจำนวน movies

Q11 แสดงหนังที่ชื่อมีอักษร ‘a’ มากที่สุด

<u>คำสั่ง</u>

ส่ง **Lab**10**_MovieCounter.java**

กำหนดส่ง **TBA**