

M10 Final Project

CS 143

Kyosuke Miyoshi

Jun 8 2023

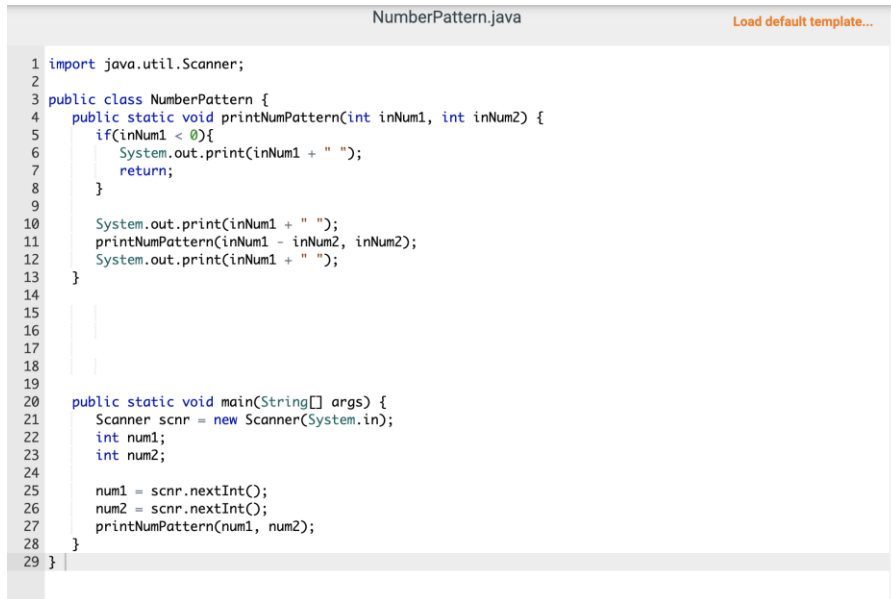
These objectives are refers to an advanced version of the previous class. The high knowledge is needed to work on these objectives. These objectives are elements of Java features and programming applications. It is important to identify each topic and use it correctly, so it is possible to develop an efficient and reliable program.

1) Objective 1: Java provides automatic memory management. Developers don't need to explicitly allocate or free memory, the garbage collector automatically reclaims unnecessary objects. This helps avoid problems with memory leaks and misuse of memory. The main advantage of linked lists is that data can be inserted and deleted relatively efficiently. When inserting new elements, programmers only need to change the references between existing nodes. While each individual operation may seem simple, the task of reading data from a file and storing it in a linked list can become challenging as the code itself becomes more complex.

```
1 import java.util.Scanner;
2
3 public class MileageTrackerLinkedList {
4     public static void main (String[] args) {
5         Scanner scnr = new Scanner(System.in);
6
7         // References for MileageTrackerNode objects
8         MileageTrackerNode headNode = null;
9         MileageTrackerNode currNode = null;
10        MileageTrackerNode lastNode = null;
11
12        double miles;
13        String date;
14        int i;
15
16
17        // Front of nodes list
18        headNode = new MileageTrackerNode();
19        lastNode = headNode;
20
21        int count = scnr.nextInt();
22
23        for(i=0;i<count;i++) {
24            miles = scnr.nextDouble();
25            date = scnr.next();
26            currNode = new MileageTrackerNode(miles, date);
27
28            lastNode.insertAfter(currNode);
29            lastNode = currNode;
30        }
31
32        currNode = headNode.getNext();
33        while (currNode != null) {
34            currNode.printNodeData();
35            currNode = currNode.getNext();
36        }
37    }
38 }
```

LAB ACTIVITY: 5.8.1: LAB: Mileage tracker for a runner

2) Objective 2: Recursion is an approach that solves the problem by calling itself. Recursion is the technique of dividing a given problem into simpler subproblems and solving them to get the final solution. In Java, recursion can be used to efficiently solve complex problems. Having a good stopping condition is important when using recursion. When implementing a recursive function, it is necessary to accurately define the base case (the condition in which the recursion ends) and the recursive case (the condition in which the recursive call is made). to infinite loops where the recursion does not terminate.

The image shows a screenshot of a Java code editor with a file named 'NumberPattern.java'. The code implements a recursive method 'printNumPattern' that prints a pattern of numbers. The base case is when 'inNum1' is less than or equal to 0, in which case it prints 'inNum1' and returns. The recursive case prints 'inNum1', then calls 'printNumPattern' with 'inNum1 - inNum2' and 'inNum2', and finally prints 'inNum1' again. The 'main' method uses a 'Scanner' to take two integers as input and calls the recursive method.

```
1 import java.util.Scanner;
2
3 public class NumberPattern {
4     public static void printNumPattern(int inNum1, int inNum2) {
5         if(inNum1 <= 0){
6             System.out.print(inNum1 + " ");
7             return;
8         }
9
10        System.out.print(inNum1 + " ");
11        printNumPattern(inNum1 - inNum2, inNum2);
12        System.out.print(inNum1 + " ");
13    }
14
15
16
17
18
19
20    public static void main(String[] args) {
21        Scanner scnr = new Scanner(System.in);
22        int num1;
23        int num2;
24
25        num1 = scnr.nextInt();
26        num2 = scnr.nextInt();
27        printNumPattern(num1, num2);
28    }
29 }
```

LAB ACTIVITY: 9.12.1: LAB: Number pattern

3) Objective 3: Inheritance is a key feature of object-oriented programming in Java. Inheritance allows programmers to base a new class on an existing class. Inheritance makes it easier to reuse code and create hierarchical data structures.

Java's collections framework contains convenient classes and interfaces for storing and managing data. Efficiently handles various data structures such as lists, sets, maps, etc.

Generics is one of the powerful features of Java. Generics allow programmers to write generic code while maintaining type safety. They can parameterize classes and methods to create reusable code for different data types.

File is marked as read only

Current file: [CourseInformation.java](#) ▾

```
2
3 public class CourseInformation {
4     public static void main(String[] args) {
5         Scanner scnr = new Scanner(System.in);
6
7         Course myCourse = new Course();
8         OfferedCourse myOfferedCourse = new OfferedCourse();
9
10        String courseNumber, courseTitle;
11        String oCourseNumber, oCourseTitle, instructorName, location, classTime;
12
13        courseNumber = scnr.nextLine();
14        courseTitle = scnr.nextLine();
15
16        oCourseNumber = scnr.nextLine();
17        oCourseTitle = scnr.nextLine();
18        instructorName = scnr.nextLine();
19        location = scnr.nextLine();
20        classTime = scnr.nextLine();
21
22        myCourse.setCourseNumber(courseNumber);
23        myCourse.setCourseTitle(courseTitle);
24        myCourse.printInfo();
25
26        myOfferedCourse.setCourseNumber(oCourseNumber);
27        myOfferedCourse.setCourseTitle(oCourseTitle);
28        myOfferedCourse.setInstructorName(instructorName);
29        myOfferedCourse.setLocation(location);
30        myOfferedCourse.setClassTime(classTime);
31        myOfferedCourse.printInfo();
32
33        System.out.println("    Instructor Name: " + myOfferedCourse.getInstructorName());
34        System.out.println("    Location: " + myOfferedCourse.getLocation());
35        System.out.println("    Class Time: " + myOfferedCourse.getClassTime());
36    }
37 }
```

Current file: [Course.java](#) ▾

[Load default template...](#)

```
1 public class Course{
2     // TODO: Declare private fields
3     private String courseNumber, courseTitle;
4
5     public void setCourseNumber(String number){
6         courseNumber = number;
7     }
8
9     public String getCourseNumber(){
10        return courseNumber;
11    }
12
13    public void setCourseTitle(String title){
14        courseTitle = title;
15    }
16
17    public String getCourseTitle(){
18        return courseTitle;
19    }
20
21    public void printInfo() {
22        System.out.println("Course Information:");
23        System.out.println("    Course Number: " + courseNumber);
24        System.out.println("    Course Title: " + courseTitle);
25    }
26
27
28
29    // TODO: Define mutator methods -
30    //     setCourseNumber(), setCourseTitle()
31
32
33    // TODO: Define accessor methods -
34    //     getCourseNumber(), getCourseTitle()
35
36
37 }
```

```

Current file: OfferedCourse.java Load default template...
1 public class OfferedCourse extends Course {
2     private String oCourseNumber, oCourseTitle, instructorName, location, classTime;
3
4     public void setInstructorName(String instructor){
5         instructorName = instructor;
6     }
7
8     public String getInstructorName(){
9         return instructorName;
10    }
11
12    public void setLocation(String place){
13        location = place;
14    }
15
16    public String getLocation(){
17        return location;
18    }
19
20    public void setClassTime(String time){
21        classTime = time;
22    }
23
24    public String getClassTime(){
25        return classTime;
26    }
27    // TODO: Declare private fields
28
29    // TODO: Define mutator methods -
30    //     setInstructorName(), setLocation(), setClassTime()
31
32    // TODO: Define accessor methods -
33    //     getInstructorName(), getLocation(), getClassTime()
34
35
36

```

LAB ACTIVITY: 2.16.1: LAB: Course Information (derived classes)

```

7 public static void main (String[] args) {
8     Scanner scnr = new Scanner(System.in);
9     String personName = "";
10    int counter = 0;
11    int youPosition = 0;
12
13    Queue<String> peopleInQueue = new LinkedList<String>();
14
15    personName = scnr.nextLine();
16    while (!personName.equals("-1")) {
17        peopleInQueue.add(personName);
18        youPosition++;
19        if (personName.equals("You")) {
20            break;
21        }
22        // TODO: Add personName to peopleInQueue and
23        //     determine position of "You" (youPosition)
24        personName = scnr.nextLine();
25    }
26
27
28    System.out.println("Welcome to the ticketing service... ");
29    System.out.println("You are number " + youPosition + " in the queue.");
30
31    while (!peopleInQueue.isEmpty()) {
32        String headPerson = peopleInQueue.remove();
33        if (headPerson.equals("You")) {
34            break;
35        }
36        System.out.println(headPerson + " has purchased a ticket.");
37        counter++;
38        System.out.println("You are now number " + (youPosition - counter));
39    }
40
41    System.out.println("You can now purchase your ticket!");
42
43

```

LAB ACTIVITY: 6.9.1: LAB: Ticketing service (Queue)

Current file: **LabProgram.java** ▾[Load default template...](#)

```
1 import java.util.Scanner;
2
3 public class LabProgram {
4     public static Pair<Integer> readIntegerPair(Scanner scnr){
5         return new Pair<>(scnr.nextInt(),scnr.nextInt());
6     }
7     public static Pair<Double> readDoublePair(Scanner scnr){
8         return new Pair<>(scnr.nextDouble(),scnr.nextDouble());
9     }
10    public static Pair<String> readStringPair(Scanner scnr){
11        return new Pair<>(scnr.next(),scnr.next());
12    }
13    public static void main(String[] args) {
14        Scanner scnr = new Scanner(System.in);
15        Pair<Integer> integerPair1 = readIntegerPair(scnr);
16        Pair<Integer> integerPair2 = readIntegerPair(scnr);
17
18        Pair<Double> doublePair1 = readDoublePair(scnr);
19        Pair<Double> doublePair2 = readDoublePair(scnr);
20
21        Pair<String> stringPair1 = readStringPair(scnr);
22        Pair<String> stringPair2 = readStringPair(scnr);
23
24        System.out.println(integerPair1+" "+integerPair1.comparisonSymbol(integerPair2)+" "+integerPair2);
25        System.out.println(doublePair1+" "+doublePair1.comparisonSymbol(doublePair2)+" "+doublePair2);
26        System.out.println(stringPair1+" "+stringPair1.comparisonSymbol(stringPair2)+" "+stringPair2);
27    }
28 }
```

Current file: **Pair.java** ▾[Load default template...](#)

```
1 public class Pair <TheType extends Comparable<TheType>> {
2     private TheType firstVal;
3     private TheType secondVal;
4
5     public Pair(TheType aVal, TheType bVal) {
6         this.firstVal = aVal;
7         this.secondVal = bVal;
8     }
9
10    public String toString(){
11        return "["+firstVal+", "+secondVal+"]";
12    }
13    public int compareTo(Pair<TheType> otherPair){
14        if(firstVal.compareTo(otherPair.firstVal)>0)
15            return 1;
16        if(firstVal.compareTo(otherPair.firstVal)<0)
17            return -1;
18        if(secondVal.compareTo(otherPair.secondVal)>0)
19            return 1;
20        if(secondVal.compareTo(otherPair.secondVal)<0)
21            return -1;
22        return 0;
23    }
24
25    public char comparisonSymbol(Pair<TheType> otherPair){
26        if(this.compareTo(otherPair)==1)
27            return '>';
28        if(this.compareTo(otherPair)==-1)
29            return '<';
30        return '=';
31    }
32 }
```

LAB ACTIVITY: 7.7.1: LAB: Pairs (generic classes)

4) Objective 4: Java provides various methods and classes for implementing algorithms such as searching and sorting. This enables efficient data retrieval and data sorting. It was difficult for

me to understand the system of searching and sorting. I felt it was very important to understand the basics of how the code works.

Algorithm analysis is a technique for evaluating the efficiency of algorithms in terms of execution time, memory usage, etc. Java provides tools and techniques for designing programs with algorithmic efficiency in mind. It took time for me to understand the role of O notation.

```
8 public static int[] readNums() {
9     Scanner scnr = new Scanner(System.in);
10    int size = scnr.nextInt();           // Read array size
11    int[] nums = new int[size];          // Create array
12    for (int j = 0; j < size; ++j) {      // Read the numbers
13        nums[j] = scnr.nextInt();
14    }
15    return nums;                         // Return the array
16 }
17
18 // Output the numbers in nums, separated by spaces.
19 // No space or newline will be output before the first number or after the last.
20 public static void printNums(int[] nums) {
21     for (int i = 0; i < nums.length; i++) {
22         System.out.print(nums[i]);
23         if (i < nums.length) {
24             System.out.print(" ");
25         }
26     }
27 }
28
29 public static void merge(int[] numbers, int i, int j, int k) {
30     int mergedSize = k - i + 1;
31     int mergedNumbers[] = new int[mergedSize];
32     int mergePos;
33     int leftPos;
34     int rightPos;
35
36     mergePos = 0;
37     leftPos = i;
38     rightPos = j + 1;
39
40     while (leftPos <= j && rightPos <= k) {
41         comparisons++;
42         if (numbers[leftPos] < numbers[rightPos]) {
43             mergedNumbers[mergePos] = numbers[leftPos];
44             ++leftPos;
45         }
46         else {
```

```

45     }
46     else {
47         mergedNumbers[mergePos] = numbers[rightPos];
48         ++rightPos;
49     }
50     ++mergePos;
51 }
52
53 while (leftPos <= j) {
54     mergedNumbers[mergePos] = numbers[leftPos];
55     ++leftPos;
56     ++mergePos;
57 }
58
59 while (rightPos <= k) {
60     mergedNumbers[mergePos] = numbers[rightPos];
61     ++rightPos;
62     ++mergePos;
63 }
64
65 for (mergePos = 0; mergePos < mergedSize; ++mergePos) {
66     numbers[i + mergePos] = mergedNumbers[mergePos];
67 }
68 }
69
70 public static void mergeSort(int numbers[], int i, int k) {
71     int j;
72
73     if (i < k) {
74         j = (i + k) / 2;
75         /* Trace output added to code in book */
76         System.out.printf("%d %d | %d %d\n", i, j, j+1, k);
77
78         mergeSort(numbers, i, j);
79         mergeSort(numbers, j + 1, k);
80
81         merge(numbers, i, j, k);
82     }
83 }

```

```

63     }
64
65     for (mergePos = 0; mergePos < mergedSize; ++mergePos) {
66         numbers[i + mergePos] = mergedNumbers[mergePos];
67     }
68 }
69
70 public static void mergeSort(int numbers[], int i, int k) {
71     int j;
72
73     if (i < k) {
74         j = (i + k) / 2;
75         /* Trace output added to code in book */
76         System.out.printf("%d %d | %d %d\n", i, j, j+1, k);
77
78         mergeSort(numbers, i, j);
79         mergeSort(numbers, j + 1, k);
80
81         merge(numbers, i, j, k);
82     }
83 }
84
85 public static void main(String[] args) {
86     int[] numbers = readNums();
87
88     System.out.print("unsorted: ");
89     printNums(numbers);
90     System.out.println("\n");
91
92     mergeSort(numbers, 0, numbers.length - 1);
93
94     System.out.print("\nsorted: ");
95     printNums(numbers);
96     System.out.println();
97     System.out.println("comparisons: " + comparisons);
98 }
99 }

```

LAB ACTIVITY: 3.11.1: LAB: Merge sort

PARTICIPATION
ACTIVITY

4.2.6: Nested loops.

Determine the big-O worst-case runtime for each algorithm.

1)

```
for (i = 0; i < N; i++) {  
  for (j = 0; j < N; j++) {  
    if (numbers[i] < numbers[j]) {  
      ++eqPerms;  
    }  
    else {  
      ++neqPerms;  
    }  
  }  
}
```

☐ $O(N)$
☐ $O(N^2)$

2)

```
for (i = 0; i < N; i++) {  
  for (j = 0; j < (N - 1); j++) {  
    if (numbers[j + 1] < numbers[j]) {  
      temp = numbers[j];  
      numbers[j] = numbers[j + 1];  
      numbers[j + 1] = temp;  
    }  
  }  
}
```

☐ $O(N)$
☐ $O(N^2)$

3)

```
for (i = 0; i < N; i = i + 2) {  
  for (j = 0; j < N; j = j + 2) {  
    cVals[i][j] = inVals[i] * j;  
  }  
}
```

☐ $O(N)$
☐ $O(N^2)$

PARTICIPATION ACTIVITY: 4.2.6: Nested loops

This course was very tough for me to understand the system, and it took a lot of time. I learned a lot of things in Java, but it does not mean I can use Java because I think it is not enough for me to do output. I completed my tool in Python. This tool uses selenium, and this tool is for making work easier. When I made this tool, I did not know a lot of things, so I searched for a lot of things while I was making this tool. And I felt output is the only way to learn programming. I will finish this course, but I have to keep learning Java and doing the output.
(<https://github.com/repezendelivery/Tool>) This is my code of the tool.