INSTITUT TEKNOLOGI SEPULUH NOPEMBER
Faculty of Intelligent Electrical, Electronic, and Informatics Engineering
Department of Information Systems
Bachelor of Computer Science Program in Information Systems
SEMESTER II, FINAL EXAMINATION, 2023/2024

ES234211 – Programming Fundamental

Course Convenors
Faizal Johan Atletiko
Renny Pradina Kusumawardani
Ahmad Muklason

Answer and Marking Scheme

19 June 2024 TIME ALLOWED: 2.5 HOURS

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **SIX (6)** questions
2. Answer ALL Questions. The marks for each question are indicated at the beginning of each question.
3. Answer the questions in any order.
4. This is **CLOSED BOOK** exam.
5. When justification is required, you **MUST** systematically write down the steps of your answer.
6. Please work independently, and **Good Luck!**

Question 1. (10 marks)

Consider the following method.

```
public void secretMethod(int [][] a){
    for(int i=1;i<a.length;i++){
        for(int j=i;j<a[0].length-1;j++){
            if (a[i][j+1] > a[i][j]){
                a[i][j] = a[i][j+1];
                a[i][j+1] = a[i][j]%1;
            }
        }
    }
}
```

For each two-dimensional array listed below, write the final array state that would result if the given array were passed as a parameter to the method. Briefly justify your answer!

(a) `int b1[][]={{0,2,4},{2,3,5},{7,8,9}}`

(b) `int b2[][]={{0,1,0,1},{1,1,1,1},{0,0,0,0}}`

Answer

(a)

```
0 2 4
2 5 0
7 8 9
```

Iteration (i)	Iteration (j)	Comparison	Action	Array State
Initial	-	-	-	{ {0, 2, 4}, {2, 3, 5}, {7, 8, 9} }
1	1	`a[1][2] > a[1][1]`	`a[1][1] = a[1][2]`	{ {0, 2, 4}, {2, 5, 5}, {7, 8, 9} }
		`5 > 3`	`a[1][2] = a[1][1] % 1`	{ {0, 2, 4}, {2, 5, 0}, {7, 8, 9} }
2	2	`a[2][3] > a[2][2]`	No action (condition false)	{ {0, 2, 4}, {2, 5, 0}, {7, 8, 9} }
		`9 > 9`	-	-

(b)

```
0 1 0 1
1 1 1 1
0 0 0 0
```

Iteration (i)	Iteration (j)	Comparison	Action	Array State
Initial	-	-	-	{ {0, 1, 0, 1}, {1, 1, 1, 1}, {0, 0, 0, 0} }
1	1	`a[1][2] > a[1][1]`	No action (condition false)	{ {0, 1, 0, 1}, {1, 1, 1, 1}, {0, 0, 0, 0} }
	2	`a[1][3] > a[1][2]`	No action (condition false)	{ {0, 1, 0, 1}, {1, 1, 1, 1}, {0, 0, 0, 0} }
2	2	`a[2][3] > a[2][2]`	No action (condition false)	{ {0, 1, 0, 1}, {1, 1, 1, 1}, {0, 0, 0, 0} }

Marking Scheme

- Each fully correct row counts for **1.5 out of 10 points** and no mark for partially correct row.
- 1 point for any answer
- halve the marks if no justification is provided
- round up the total mark.

Question 2. (20 marks)

Write a method called ticTacToe that accepts a 3x3 array of chars ('O' or'X') as a parameter and print the board and the winner of the game, either 'O' or 'X'. If no winner print "Draw". To determine the winner of a 3x3 Tic-Tac-Toe game, you need to check all possible lines (rows, columns, and diagonals) where a player could have placed three of their marks in a row. Example input parameter:

```
input                  | output
=======================================
char t[][]={           | X O O
    {'X','O','O'},     | X X X
    {'X','X','X'},     | O X O
    {'O','X','O'},     |
}                      | The winner : X
---------------------------------------
char t[][]={           | O X O
    {'O','X','O'},     | X O X
    {'X','O','X'},     | X O O
    {'X','O','O'},     |
```

```
}                    | The winner : O
--------------------------------------
char t[][]={          | O X O
    {'O','X','O'},   | O O X
    {'O','O','X'},   | X O X
    {'X','O','X'},   |
}                    | The winner : Draw
--------------------------------------
```

Hints: Steps to Determine the Winner:

1. Check Rows: Loop through each row to see if all elements are the same.
2. Check Columns: Loop through each column to see if all elements are the same.
3. Check Diagonals: Check the two diagonals to see if all elements are the same.
4. Check for Draw: If all cells are filled and there's no winner, it's a draw.

Answer

One of alternative answer:

```
public static void ticTacToe(char t[][]){
    String winner="Draw";
    for(int i=0;i<t.length;i++){
        for(int j=0;j<t[0].length;j++){
            System.out.print(t[i][j]+" ");
        }
        System.out.println();

        //cheking rows
        if(t[i][0]==t[i][1] && t[i][1]==t[i][2]) {
            winner=""+t[i][0];
        }
        //cheking columns
        if(t[0][i]==t[1][i] && t[1][i]==t[2][i]) {
            winner =""+t[0][i];
```

```
            }
        }
        //checking diagonal
        if(t[0][0] == t[1][1] && t[1][1]==t[2][2]) {
            winner =""+t[0][0];
        }
        if(t[0][2] == t[1][1] && t[1][1]==t[2][0]) {
            winner =""+t[0][2];
        }
        System.out.println("the winner : "+winner);
    }
```

Marking Scheme

- Correct printing the board counts **4 out of 20 points**.
- Correct checking rows counts **4 out of 20 points**.
- Correct checking columns counts **4 out of 20 points**.
- Correct checking first diagonal counts **3 out of 20 points**.
- Correct checking second diagonal counts **3 out of 20 points**.
- Correct checking draw condition counts **2 out of 20 points**.

Question 3. (10 marks)

A recursive method is defined as follow:

```
public static int mystery(int n) {
    if (n < 10) {
        return n;
    } else {
        int a = n / 10;
        int b = n % 10;
```

```
            return mystery(a + b);
        }
    }
```

What is the output from the following method call?Briefly justify your answer!

(a) `mystery(678)`

(b) `mystery(555)`

Answer

(a) `mystery(678)`

Call	n	Cond	a	b	Rec Call	Res
1	678	n>=10	67	8	mystery(75)	
2	75	n>=10	7	5	mystery(12)	
3	12	n>=10	1	2	mystery(3)	
4	3	n<10	-	-	-	3

(b) `mystery(555)`

Call	n	Cond	a	b	Rec Call	Res
1	555	n>=10	55	5	mystery(60)	
2	60	n>=10	6	0	mystery(6)	
3	6	n<10	-	-	-	6

Marking Scheme

- Part (a) counts for **5 out of 10 points.**
- Part (b) counts for **5 out of 10 points.**
- For any partially correct answer, please mark evenly.
- Halve the mark if no justification is provided (i.e. showing the output only)
- If the total mark contains fractions, please round up to the nearest integer.

Question 4. (20 marks)

Complete the following **recursive** method:

```
public static void zap(int n){
    if(n<1) throw new IllegalArgumentException();
    //your code goes here

}
```

that prints out n characters as follows. The middle character of the output should always be an asterisk ("*"). If you are asked to write out an even number of characters, then there will be two asterisks in the middle ("**"). Before the asterisk(s) you should write out less-than characters ("<"). After the asterisk(s) you should write out greater-than characters (">"). For example, the following calls produce the following output:

```
Example:
method call      | expected output
-----------------------------------------
zap(1)           | *
zap(2)           | **
zap(3)           | <*>
zap(4)           | <**>
```

```
zap(5)          | <<*>>
zap(6)          | <<**>>
zap(7)          | <<<*>>>
zap(8)          | <<<**>>>
```

Hints : within the recursive case decrease the parameter by 2.
Note : You are **NOT ALLOWED** to use any looping.

Answer
One of alternative answer:

```
public static void zap(int n){
    if(n<1) throw new IllegalArgumentException();
    else{
        //checking point 1
        if(n==1){
            System.out.print("*");
        }
        //checking point 2
        else if(n==2){
             System.out.print("**");
        }
        else{
            //checking point 3
            System.out.print("<");
            zap(n-2);
            System.out.print(">");
        }
    }
}
```

Marking Scheme

- Checking point 1 counts **5 out of 20 points**.
- Checking point 2 counts **5 out of 20 points**.

- Checking point 3 counts **10 out of 20 points**.

Question 5. (15 marks)
The following is the top five paperback non-fiction on the New York Time's list of bestsellers in the third week of June, 2024:

1. The Body Keeps the Score
2. Killers of the Flower Moon
3. The Backyard Bird Chronicles
4. Braiding Sweet grass
5. The Boys in the Boat

These books have been on the list for 294, 171, 7, 217, and 165 weeks respectively. Create a simple Java program which operates solely from a public static void main method (i.e. you do not need to create classes or instances of the book), which:

- Stores the names of the books in an array
- Stores the duration on the list in another array
- Enables querying of a book and its duration on the list based on its rank (1 to 5)
- Is able to handle run time errors caused by users entering non-number input or out-of-scope numbers (less than 1 or more than 5).

Example input and outputs:

```
Input : Enter rank on the list: 0
Output: Rank is invalid, must be from 1 to 5

Input : Enter rank on the list: best
Output: Input is invalid, must be a number from 1 to 5

Input : Enter rank on the list: 10
```

```
Output: Rank is invalid, must be from 1 to 5

Input : Enter rank on the list: 2
Output: Killers of the Flower Moon, has been on
        the list for 171 weeks.
```

Hints: Use `ArrayIndexOutOfBoundsException` and `NumberFormatException` or `InputMismatchException`

Answer

One of alternative answer:

```
import java.util.Scanner;
public class BestsellerList {
public static void main(String[] args) {
//checking point 1
// Store the names of the books in an array
String[] books = {
"The Body Keeps the Score",
"Killers of the Flower Moon",
"The Backyard Bird Chronicles",
"Braiding Sweetgrass",
"The Boys in the Boat"
};

// Store the duration on the list in another array
int[] weeksOnList = {294, 171, 7, 217, 165};

//checking point 2
// Scanner to read user input
Scanner scanner = new Scanner(System.in);
System.out.print("Enter rank on the list: ");


while(true) { // Non-obligatory, but nice :)
//checking point 3
```

```
try {
// Reads the next input as an Integer
// Alternative syntaxes may also be used,
so long they give // similar results
int rank =Integer.parseInt(scanner.nextLine());
// Print the book and its duration on the list
System.out.println(books[rank-1] + ", has
been on the list for " + weeksOnList[rank-1] + " weeks.");
break; // Non-obligatory

//checking point 4
} catch (NumberFormatException e) {
// when reading input using Integer.parseInt()
System.out.println("Input is invalid, must
be a number from 1 to 5.");

//checking point 5
} catch (ArrayIndexOutOfBoundsException e) {
System.out.println("Rank is invalid, must be from 1 to 5.");


} finally { // Non-obligatory
System.out.print("Enter rank on the list: ");
}
}
// Close the scanner
scanner.close();
}
}
```

Marking Scheme

- Each checking point counts **3 out of 15 points**.

- Mark evenly for partially correct answer.

Question 6. (25 marks)

A vet clinic, "Fauna Medika", currently focuses on the treatment of cats. You are asked to design a simple Vet Clinic Management System for the clinic using object-oriented programming principles. This system will manage various types of cats receiving treatment and the veterinarians at the clinic. At its current capacity, the clinic can treat at most 20 cats at a time, and has at most 5 practicing veterinarians. The classes and requirements for this system are as follows:

```
Class Cat:
---------------------------
Fields:
String name
int age
boolean isShortHaired
String color
String breed

Methods:
- Constructor to initialize all fields.
- void displayDetails(): Displays the details (fields) of the cat.

Class Veterinarian:
---------------------------
Fields:
String name
String vetId
Methods:
- Constructor to initialize all fields.
- void displayVetDetails(): Displays the details
  (fields) of the veterinarian.

Class ClinicManagement:
---------------------------
```

```
Fields:
Cat[] catList
Vet[] vetList
Methods:
- Constructor to initialize the lists capacity,
  ClinicManagement(int numCats, int numVets).
- void addCat(Cat cat): Adds a cat to the clinic
  management system and print message (see expected output).
- void addVet(Veterinarian vet): Adds a
  veterinarian to the clinic management system
  and print message (see expected output).
- void displayAllCats(): Displays all cats in the clinic.
- void displayAllVeterinarians(): Displays all
  veterinarians in the clinic.

Class Main:
---------------------------
Contains the public static void main(String[] args)
for the system, which does the following:

- Create a new Clinic Management object that can
  have a maximum of 20 patients and 5 vets
  (hint: these numbers are the array capacity).
- Create two cat objects:
   Name: "Butter", age: 2 year, long coat/fur,
   color: cream, breed: "Persian"
   Name: "Telo", age: 1 year, short coat/fur,
   color: orange, breed: "British Shorthair"
- Add the cats to the clinic management object's catList
- Create one veterinarian:
  Name: drh. Agus Wardana, id: 201032
- Add the veterinarian to the clinic management object's vetList
- Display all cat(s) and all veterinarian(s).
```

Expected Output:

```
Adding a Cat to the Vet Clinic Management System...

Name: Butter
```

```
Age: 2
Short Haired: No
Breed: Persian

Adding a Cat to the Vet Clinic Management System...

Name: Telo
Age: 1
Short Haired: Yes
Breed: British Shorthair


Adding Veterinarian to the Vet Clinic Management System...

Name: drh. Agus Wardana
Vet ID: 201032

Displaying All Cats in the Clinic:

Name: Butter
Age: 2
Short Haired: No
Breed: Persian

Name: Telo
Age: 1
Short Haired: Yes
Breed: British Shorthair

Displaying All Veterinarians in the Clinic:

Name: drh. Agus Wardana
Vet ID: 201032
```

Answer

One of alternative answer:

```
//Class representing a Cat
class Cat {
String name;
int age;
boolean isShortHaired;
String color;
String breed;
// Constructor to initialize all fields
public Cat(String name, int age, boolean isShortHaired, String color,
this.name = name;
this.age = age;
this.isShortHaired = isShortHaired;
this.color = color;
this.breed = breed;
}
// Method to display the details of the cat
public void displayDetails() {
System.out.println("Name: " + name);
System.out.println("Age: " + age);
System.out.println("Short Haired: " + (isShortHaired ? "Yes" : "No"));
System.out.println("Breed: " + breed);
}
}
//Class representing a Veterinarian
class Veterinarian {
String name;
String vetId;
// Constructor to initialize all fields
public Veterinarian(String name, String vetId) {
this.name = name;
this.vetId = vetId;
}
// Method to display the details of the veterinarian
public void displayVetDetails() {
System.out.println("Name: " + name);
System.out.println("Vet ID: " + vetId);
}
}
```

```
//Class to manage the clinic operations
class ClinicManagement {
Cat[] catList;
Veterinarian[] vetList;
int catCount;
int vetCount;
// Constructor to initialize the lists' capacity
public ClinicManagement(int numCats, int numVets) {
catList = new Cat[numCats];
vetList = new Veterinarian[numVets];
catCount = 0;
vetCount = 0;
}
// Method to add a cat to the clinic management system
public void addCat(Cat cat) {
if (catCount < catList.length) {
catList[catCount++] = cat;
System.out.println("Adding a Cat to the Vet
Clinic Management System...\n");
cat.displayDetails();
System.out.println();
} else {
System.out.println("Cannot add more cats. Capacity full.");
}
}
// Method to add a veterinarian to the clinic management system
public void addVet(Veterinarian vet) {
if (vetCount < vetList.length) {
vetList[vetCount++] = vet;
System.out.println("Adding Veterinarian to
the Vet Clinic Management System...\n");
vet.displayVetDetails();
System.out.println();
} else {
System.out.println("Cannot add more veterinarians. Capacity full.");
}
}
// Method to display all cats in the clinic
```

```
public void displayAllCats() {
System.out.println("Displaying All Cats in the Clinic:\n");
for (int i = 0; i < catCount; i++) {
catList[i].displayDetails();
System.out.println();
}
}
// Method to display all veterinarians in the clinic
public void displayAllVeterinarians() {
System.out.println("Displaying All Veterinarians in the Clinic:\n");
for (int i = 0; i < vetCount; i++) {
vetList[i].displayVetDetails();
System.out.println();
}
}
}


//Main class to run the program
public class Main {
public static void main(String[] args) {
// Create a new Clinic Management object
//with a capacity of 20 cats and 5 vets
ClinicManagement clinic = new ClinicManagement(20, 5);
// Create two cat objects
Cat cat1 = new Cat("Butter", 2, false, "cream", "Persian");
Cat cat2 = new Cat("Telo", 1, true, "orange", "British Shorthair");
// Add the cats to the clinic management object
clinic.addCat(cat1);
clinic.addCat(cat2);
// Create one veterinarian
Veterinarian vet = new Veterinarian("drh. Agus Wardana", "201032");
// Add the veterinarian to the clinic management object
clinic.addVet(vet);
// Display all cats and veterinarians in the clinic
clinic.displayAllCats();
clinic.displayAllVeterinarians();
}
```

```
}
```

Marking Scheme

- Cat Class counts **5 out of 25 points**.
- Veterinarian Class counts **5 out of 25 points**.
- ClinicManagement class counts **7 out of 25 points**.
- Main Class counts **8 out of 25 points**.
- Mark evenly for partially correct answer.