

FIRST SEMESTER SESSION 2022/2023 (A221)

STIA2024 DATA STRUCTURES AND ALGORITHM ANALYSIS

UUM COLLEGE OF ARTS AND SCIENCES

================================================================

ASSIGNMENT 2 REPORT

================================================================

*Submitted to:*

Dr. Sharhida Zawani Binti Saad

*Prepared by:*

|  |  |  |
| --- | --- | --- |
|  | NAME | MATRIC.NO |
| 1 | VINCENT BEH HUA EIK | 279018 |
| 2 | POON WAI KIT | 279021 |

Data Structure Concept Applied

The data structure implemented for this assignment is the Contiguous List by using an array. Essentially, we use an array to store objects in sequential order as a way to implement the list. As we have already decided the size of the list (5), the number of data is small and the operation of deletion happens rarely for the type of program created. The implementation of contiguous list using an array to store data is suitable.

Operations of Contiguous List in the Program

There are a number of operations that the program can perform namely creating the list, finding if the list is empty or full, adding a new element to the list, updating an element in the list, removing an element from the list, searching for an element on the list and printing the content of the list. The element stated is the Vacation objects which will be used in the program.

Initializing the Object

Before we can implement the List in the program, we would need to create a class to initialize the object needed which are Vacation objects. This is because we are using array of objects to implement the contiguous list. The Vacation class will contain the elements to the object, the constructor, the accessors, mutators and the toString() method to allow us a way manipulate how the output of the program will be.

Creating the List

With the Vacation Class initialized, we are now ready to implement the List using array of objects.



As you can see from the code snippet above, the List is initialized as an array of object Vacation with the size of 5 objects in the main class. This will allow us to perform other operations by manipulating the Vacation objects.

Find if the List is empty or full

To find if the List is empty or full, we create 2 methods which will allow the program to return whether the list is empty or full. This will then be implemented in other methods which require them. The code snippet below is the 2 methods.

Text

Description automatically generated

Adding a new element to the List

Now we proceed with the main functions of the program. Adding a new element or adding a new Vacation object to the List will mean that we are adding objects to the array. That action will be in sequential order where if the list is empty, the first added object will be in index 0 while the following object will be in order of index 1, 2, 3, 4 respectively. The code snippet below is the implementation of the operation.

Text

Description automatically generated

As you can see, the isFull() is implemented within the add() method to ensure no additional objects is added within the limited size of the initialized array which will cause an ArrayOutOfBounds error to occur if it happens.

Updating an element in the List

In the operation of updating a Vacation object, what the program essentially does is allowing the user to create a new object with their desired information and choose which already created object within the list they would want the new object to replace. This operation is mainly for use when the list is full or if the user would want to make changes to the already added objects. For this operation, the isEmpty() method is implemented as no objects can be updated if the list is empty. That can be seen as the code snippet attached below.

Text

Description automatically generated

Removing an element from the List

Now moving to removing or deleting an object from the List, this operation will be slightly more complicated and intensive on computing power. This is because when an object is chosen to be removed, that action is not sequential as the object that is desired to be deleted is not necessarily at the end of the list. If the object is in the middle of the list, that removed object will leave a gap in the array. To solve that issue, the code for the operation will need to consider that possibility and shift the objects at the back to the front in order to prevent the gap. The snippet of the code below will include the isEmpty() method as there will be nothing to remove if the list is empty while the mentioned method to shift the objects is also included.

Text

Description automatically generated

Searching for an element on the List

To search for the specific object on the list, we would need to first establish what elements of the object we are using to search for said object. In this case, we are using the location to search for the Vacation object. When the option to search is chosen, the program will ask the user to input the location which they would like to search for. The input will then be passed through the search method to allow the program to search for the object.

Text

Description automatically generated

The input is then used to compare to the objects in the list through a for loop to look for a matching location. If there is a match, the program will display the information of the object while the program will print out “No vacations found” if the input does not match any locations of the objects stored in the list. As usual, a isEmpty() method is implemented as the program cannot search through anything if the list is empty.

Text

Description automatically generated

Printing the content of the List

Finally, the last operation is printing or displaying the content of the list. This method will include the formatting to ensure the output is neat. The method will have a for loop to cycle through all the added objects for printing and it will also include a isEmpty() method as printing without any objects in the list is pointless. The snippet of the method is as below.

Text

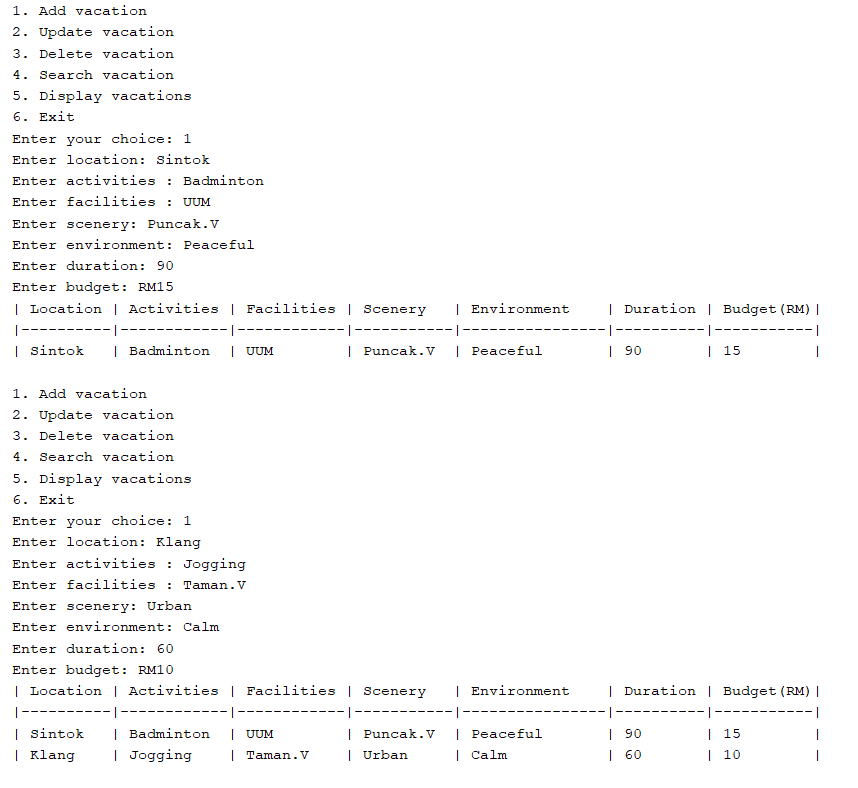
Description automatically generated

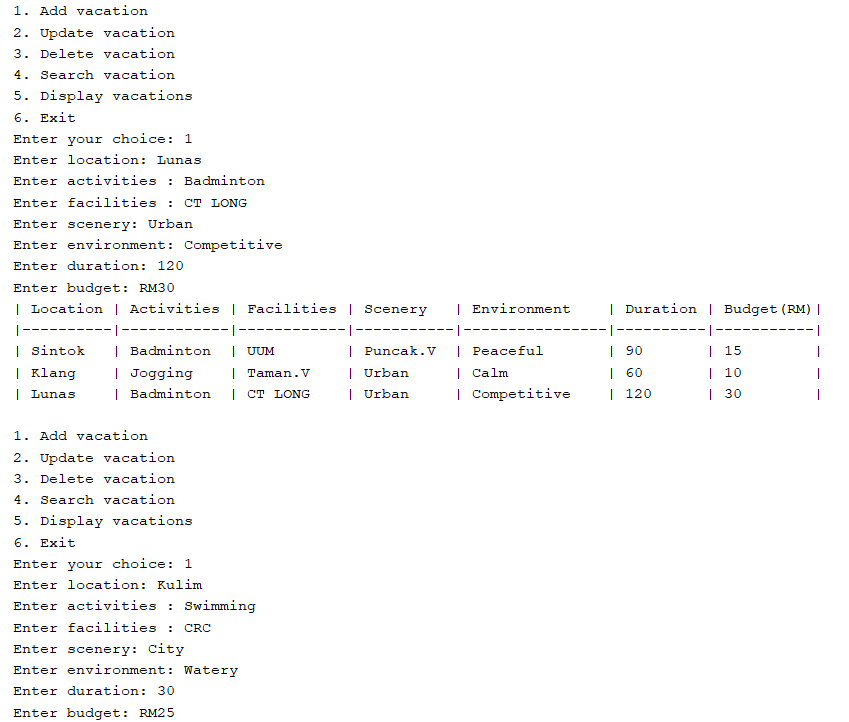
Sample Output

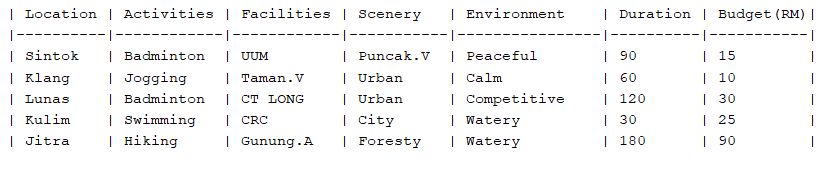


First running the program try choice2,3,4and 5 to show the list is empty and can continue run. Choosing these choices will show the List is empty.

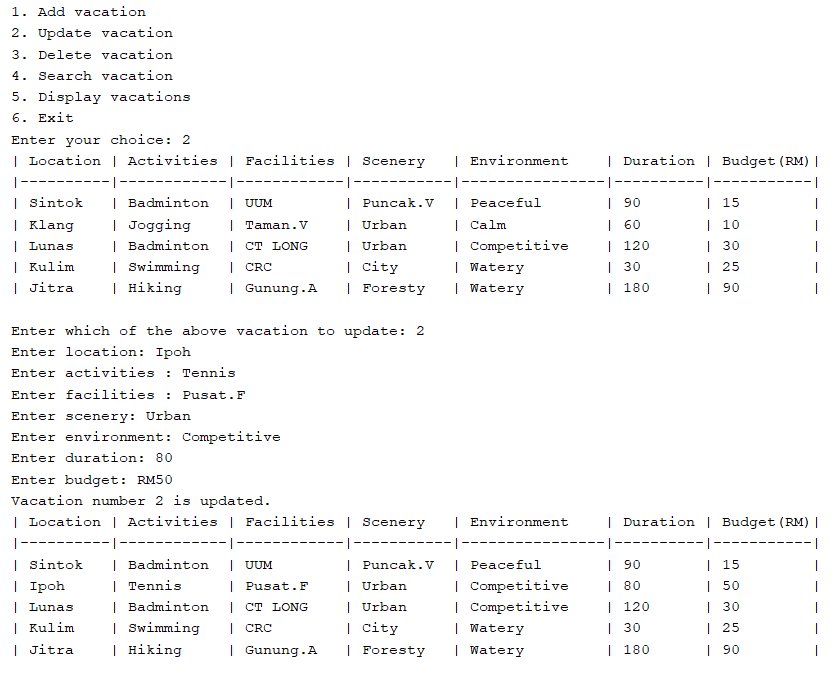
Add data method Output:



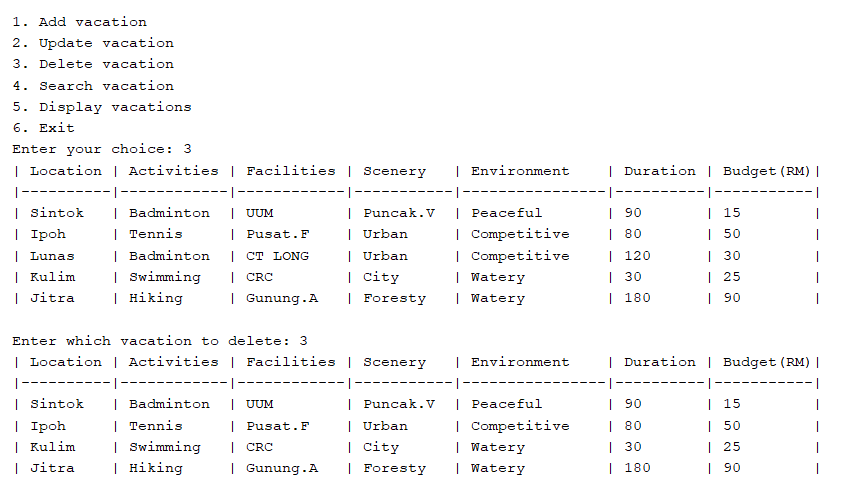




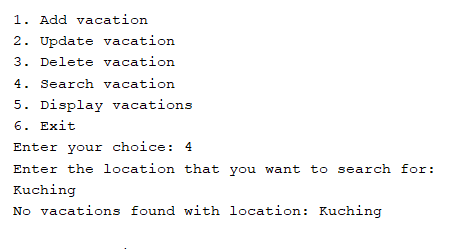
Delete data Method Sample output:

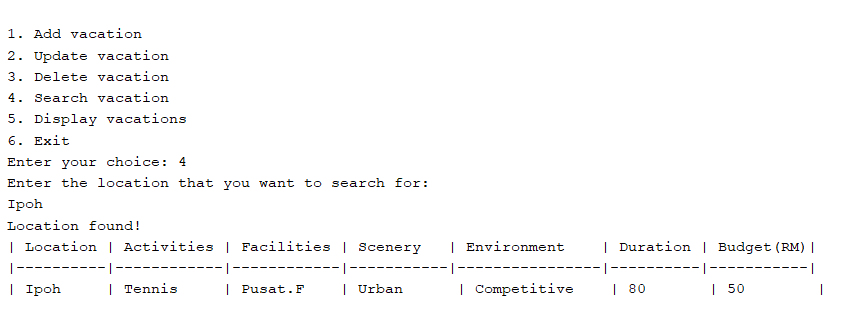


Delete Data Method Sample Output:

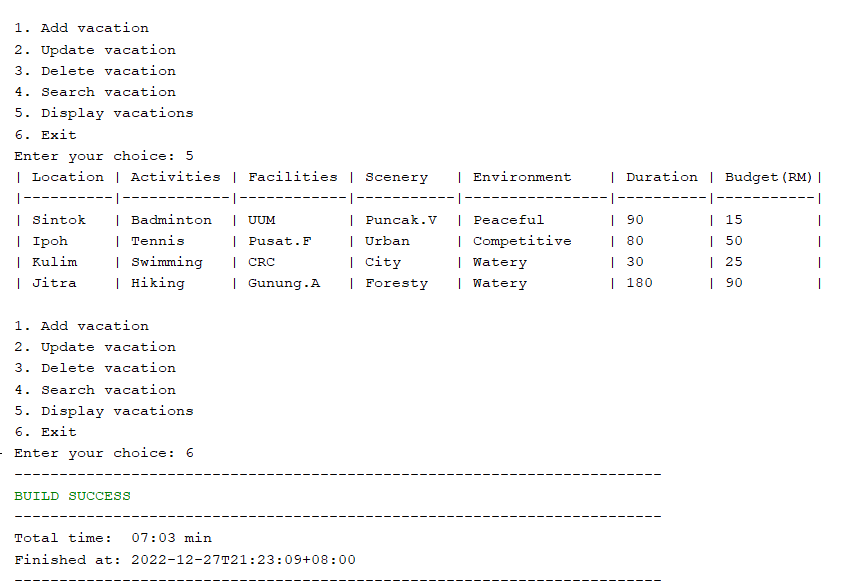


Search Data Method:





Display Data Method Sample Output:



Source Code(Main Class)

import java.util.Arrays;

import java.util.Scanner;

public class VacationList {

private static Vacation[] vacations = new Vacation[5];

private static int size = 0;

public static void main(String[] args) {

VacationList vacationList = new VacationList();

Scanner scanner = new Scanner(System.in);

OUTER:

while (true) {

System.out.println("1. Add vacation");

System.out.println("2. Update vacation");

System.out.println("3. Delete vacation");

System.out.println("4. Search vacation");

System.out.println("5. Display vacations");

System.out.println("6. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

add();

break;

case 2:

update();

break;

case 3:

delete();

break;

case 4:

System.out.println("Enter the location that you want to search for: ");

String location = scanner.nextLine();

search(location);

break;

case 5:

display();

break;

case 6:

break OUTER;

default:

break;

}

}

}

// Adds a new vacation to the list

public static void add() {

Scanner scanner = new Scanner(System.in);

// Check if the array is full and resize if necessary

if (isFull())

System.out.println("Vacation List is full!");

else {

System.out.print("Enter location: ");

String location = scanner.nextLine();

System.out.print("Enter activities : ");

String activities = scanner.nextLine();

System.out.print("Enter facilities : ");

String facilities = scanner.nextLine();

System.out.print("Enter scenery: ");

String scenery = scanner.nextLine();

System.out.print("Enter environment: ");

String environment = scanner.nextLine();

System.out.print("Enter duration: ");

int duration = scanner.nextInt();

System.out.print("Enter budget: RM");

int budget = scanner.nextInt();

scanner.nextLine();

vacations[size] = new Vacation(location, activities, facilities, scenery, environment, duration, budget);

size++;

display();

}

}

// Updates an existing vacation in the list

public static void update() {

if (isEmpty())

System.out.println("Vacation List is empty!");

else {

display();

Scanner scanner = new Scanner(System.in);

System.out.print("Enter which of the above vacation to update: ");

int index = scanner.nextInt();

scanner.nextLine();

System.out.print("Enter location: ");

String location = scanner.nextLine();

System.out.print("Enter activities : ");

String activities = scanner.nextLine();

System.out.print("Enter facilities : ");

String facilities = scanner.nextLine();

System.out.print("Enter scenery: ");

String scenery = scanner.nextLine();

System.out.print("Enter environment: ");

String environment = scanner.nextLine();

System.out.print("Enter duration: ");

int duration = scanner.nextInt();

System.out.print("Enter budget: RM");

int budget = scanner.nextInt();

scanner.nextLine();

vacations[index-1] = new Vacation(location, activities, facilities, scenery, environment, duration, budget);

System.out.println("Vacation number " + index + " is updated.");

display();

}

}

// Deletes a vacation from the list

public static void delete() {

Scanner scanner = new Scanner(System.in);

if (isEmpty())

System.out.println("Vacation List is empty!");

else {

display();

System.out.print("Enter which vacation to delete: ");

int index = scanner.nextInt();

scanner.nextLine();

index = index -1;

if (index < 0 || index > size-1) {

System.out.println("Error: Invalid index.");

}

else{

for (int x = index; x<size-1 ; x++){

vacations[x] = vacations[x+1];

}

size--;

display();

}

}

}

// Searches the list for a vacation with a specific location

public static void search(String location) {

if (isEmpty())

System.out.println("Vacation List is empty!\n");

else {

boolean found = false;

for (int i = 0; i < size; i++) {

if (vacations[i].getLocation().equalsIgnoreCase(location)) {

found = true;

System.out.println("Location found!");

System.out.println("| Location | Activities | Facilities | Scenery | Environment | Duration | Budget(RM)|");

System.out.println("|----------|------------|------------|------------|----------------|----------|-----------|");

Vacation vacation = vacations[i];

System.out.printf("| %-8s | %-10s | %-10s | %-7s | %-14s | %-8d | %-9d |%n",

vacation.getLocation(),

vacation.getActivities(),

vacation.getFacilities(),

vacation.getScenery(),

vacation.getEnvironment(),

vacation.getDuration(),

vacation.getBudget());

System.out.println("");

}

}

if (!found) {

System.out.println("No vacations found with location: " + location);

System.out.println("");

}

}

}

// Displays all vacations in the list

public static void display() {

if (isEmpty())

System.out.println("Vacation List is empty!");

else {

// Print table header

System.out.println("| Location | Activities | Facilities | Scenery | Environment | Duration | Budget(RM)|");

System.out.println("|----------|------------|------------|--------------|----------------|----------|-----------|");

// Print details of each vacation

for (int i = 0; i < size; i++) {

Vacation vacation = vacations[i];

System.out.printf("| %-8s | %-10s | %-10s | %-7s | %-14s | %-8d | %-9d |%n",

vacation.getLocation(),

vacation.getActivities(),

vacation.getFacilities(),

vacation.getScenery(),

vacation.getEnvironment(),

vacation.getDuration(),

vacation.getBudget());

}

System.out.println("");

}

}

public static boolean isEmpty( ) {

if(size == 0)

return true;

else

return false;

}

public static boolean isFull( )

{

if(size == vacations.length)

return true;

else

return false;

}

}

Source Code(Sub Class)

public class Vacation {

private String location;

private String activities;

private String facilities;

private String scenery;

private String environment;

private int duration;

private int budget;

public Vacation(String location, String activities, String facilities, String scenery, String environment, int duration, int budget) {

this.location = location;

this.activities = activities;

this.facilities = facilities;

this.scenery = scenery;

this.environment = environment;

this.duration = duration;

this.budget = budget;

}

public String getLocation() {

return location;

}

public String getActivities() {

return activities;

}

public String getFacilities() {

return facilities;

}

public String getScenery() {

return scenery;

}

public String getEnvironment() {

return environment;

}

public int getDuration() {

return duration;

}

public int getBudget() {

return budget;

}

@Override

public String toString() {

return "Location: " + location + ", Activities: " + activities + ", Facilities: " + facilities + ", Scenery: " + scenery + ", Environment: " + environment + ", Duration: " + duration + ", Budget: " + budget;

}

}