PRACTICAL ASSIGNMENT - MARKING REPORT

1. PERSONAL DATA

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Group number :** | | | | |
| No | Name | ID | Programme | Total Marks |
| 1. |  |  |  |  |
| 2. |  |  |  |  |
| 3. |  |  |  |  |
| 4. |  |  |  |  |

1. SUBMISSION STATUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No soft copy/ Upload wrong file(s) | Late submission of softcopy | No hardcopy | Late submission of hardcopy | No issue |
|  |  |  |  |  |

1. COMPILATION AND RUNNING

|  |  |  |
| --- | --- | --- |
| Does not compile/Bytecode & batch file do not work | Compile but no output/ wrong output/ run-time error | Compile and produce output |
|  |  |  |

1. PRESENTATION OF SOURCE CODES(3%)
2. Indent Style (1.5%) Poor Inconsistent Good
3. Identifier names (1.5%) Poor choice Meaningful Meaningful and good naming convention
4. PROGRAM COMPONENT (57% + 3%)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Program Components | Missing/ Does not work | Major errors | Minor errors | Not robust | No issue/ Excellent design | Max marks | Marks obtained |
| Framework Design (Use of interfaces and abstract classes) |  |  |  |  |  | 10 |  |
| Classes for storing objects (data structures/containers) |  |  |  |  |  | 12 |  |
| Bin Packing Algorithms (at least 2) |  |  |  |  |  | 16 |  |
| Test program (main program, set of bins and set of objects) |  |  |  |  |  | 14 |  |
| Exception and error handling |  |  |  |  |  | 5 |  |
| Presentation of source codes |  |  |  |  |  | 3 |  |
|  |  |  |  |  | Total | 60 |  |

1. REPORT AND OTHER COMPONENT (40%)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Components | Missing | Poor | Average | Good | Excellent | Max marks | Marks obtained |
| The proposed solution and design (data structures and algorithms) |  |  |  |  |  | 8 |  |
| Discussion (efficiency and complexities) |  |  |  |  |  | 12 |  |
| Flowchart |  |  |  |  |  | 5 |  |
| UML Diagram |  |  |  |  |  | 5 |  |
| Sample input and test cases |  |  |  |  |  | 5 |  |
| Screenshots |  |  |  |  |  | 5 |  |
|  |  |  |  |  | Total | 40 |  |

Table of Contents

[1.0 Proposed Solution 2](#_Toc196771664)

[1.1 People (Items) 2](#_Toc196771665)

[1.2 Elevator (Container) 2](#_Toc196771666)

[1.3 Bin Packing Algorithms (Strategy) 3](#_Toc196771667)

[2.0 Discussion and Complexity Analysis 4](#_Toc196771668)

[2.1 Efficiency of Algorithms 4](#_Toc196771669)

[2.2 Complexity of Algorithms 4](#_Toc196771670)

[3.0 Flowchart of Bin Packing Algorithms 5](#_Toc196771671)

[4.0 UML Diagram of Java Program 6](#_Toc196771672)

[5.0 Sample of Input Data and Test Cases 7](#_Toc196771673)

[6.0 Sample Output(s) 8](#_Toc196771674)

[7.0 Print Out of Java Program 9](#_Toc196771675)

# 1.0 Proposed Solution

In our daily life, clearing a queue that is using the elevator is a common headache due to unoptimized packing of people in the elevator. For this bin-packing assignment, the main goal is to minimize the number of elevators needed to compensate a queue of people, where people will be optimally packed into elevator. In this case, the people are the items, and the elevator is the container. The solution to achieve said goal are strategies that implements bin-packing algorithms.

## 1.1 People (Items)

A person class declared in “Person.java” outlines the attribute of a person. The attributes of a person are as follows:

|  |  |
| --- | --- |
| **Attribute** | **Representation** |
| area (m2) | The area occupied when standing in an elevator |
| weight (Kg) | The load added on elevator |

A group of persons is identified as people, where the queue data structure declared in “ElevatorQueue.java” is used to store each individual person. The reason for picking queue as the suitable data is to simulate real world scenario where people queue up to enter the elevators on a first come first serve basis.

## 1.2 Elevator (Container)

An elevator class declared in “ElevatorBin.java” outlines the attribute of an elevator. The attributes of an elevator are as follows:

|  |  |
| --- | --- |
| **Attribute** | **Representation** |
| bin | The people that are currently in the elevator |
| FULL\_LOAD (Kg) | The max load of people that the elevator can handle |
| FULL\_AREA (m2) | The dimension of the elevator’s floor to stand on |
| currentLoad (Kg) | The current load of people in the elevator |
| currentArea (m2) | The current area occupied by people in the elevator |

The elevator class is designed like a stack to capture the nature of first in last out when people are entering and exiting the elevator. A group of elevator is identified as elevators, where the array list data structure declared in “ElevatorArrayList.java” is used to store the elevator used. The reason for picking array list as the suitable data structure is because of the constant time indexing operation, which is commonly used in bin-packing algorithms.

## 1.3 Bin Packing Algorithms (Strategy)

To pack the people into the elevators efficiently, four popular approximation bin packing algorithms will be used. The algorithms implemented are:

|  |  |
| --- | --- |
| **Algorithms** | **Strategy** |
| First Fit (FF) | People enter the left-most elevator that can handle them. |
| Best Fit (BF) | People enter the elevator with the least area available left. |
| Next Fit (NF) | People only enter the current elevator. |
| Worst Fit (WF) | People enter the elevator with the most area available left. |

These algorithms serve as an experimental societal norm of which elevator people should enter when queuing. Multiple algorithms are implemented to better showcase the difference in efficiency and complexity.

# 2.0 Discussion and Complexity Analysis

## 2.1 Efficiency of Algorithms

Each of the algorithms are tested with the same queue of people. The test is performed three separate times with different number of people in the queue. The documented results are as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Number of People** | **Algorithms** | **First Fit** | **Best Fit** | **Next Fit** | **Worst Fit** |
| 5000 | **Elevators Used** | 461 | 461 | 472 | 462 |
| 50 000 | **Elevators Used** | 4605 | 4603 | 4721 | 4609 |
| 500 000 | **Elevators Used** |  |  |  |  |

## 2.2 Complexity of Algorithms

Blah blah

# 3.0 Flowchart of Bin Packing Algorithms

# 4.0 UML Diagram of Java Program

# 5.0 Sample of Input Data and Test Cases

# 6.0 Sample Output(s)

# 7.0 Print Out of Java Program