**SW2 – Project Evaluation Form**

* **Each team must submit the following Documentation that contains:**

**- Project Description in detail.**

**- Class Diagram. And Database Schema.**

* **Each team must submit the project via GitHub:**
* **Source Code.**
* **Video Demo for running ( 2 – 5 Minutes ).**
* **Documentation and Evaluation Form**
* **The Evaluation will start with giving all teams 30 marks then check the following criteria:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Violation Level** | **Full** | **Medium** | **Small** | **Grade** |
| **Documentation** | **-5** | **-2** | **-1** |  |
| **Not Apply MVC (it does not Separate Business logic from GUI).**  **Example of violation: write the implantation for a method such as an inset item into the database inside the Button Action method)** | **-6** | **- 3** | **-1** |  |
| **Violate clean code – Variables** | **-2** | **-1** | **-.05** |  |
| **Violate clean code – Functions** | **-2** | **-1** | **-.05** |  |
| **Violate Single-responsibility Principle** | **-2** | **-1** | **-.05** |  |
| **Violate Open-closed Principle** | **-2** | **-1** | **-.05** |  |
| **Violate the Liskov Substitution Principle** | **-2** | **-1** | **-.05** |  |
| **Violate Interface Segregation Principle** | **-2** | **-1** | **-.05** |  |
| **Violate Dependency Inversion Principle** | **-2** | **-1** | **-.05** |  |
| **Not Upload code to GitHub** | **-1** | | |  |
| **Only One Branch Without Merge (GitHub)** | **-2** | | |  |
| **Only One Contribution (GitHub)** | **-2** | | |  |
| **Total Minus from Grade** |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Design Pattern Bounce** | **+4** |  |
| **Bounce on Overall Work** | **+2** |  |
| **Total Team Grade / 30** |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name (Arabic)** | **ID** | **Individual Bounce +2** | **Grade** | **Grade** |
| سامح وليد محمد الشافعي | 201900335 |  |  |  |
| سعيد مليجي سيد مليجي | 201900341 |  |  |  |
| سعيد عماد ماهر محمد | 201900339 |  |  |  |
| **زياد احمد عبدالتواب سعيد** | 201900322 |  |  |  |
| **ستيف عادل فاروق حبيب** | 201900337 |  |  |  |
| **محمد شريف فتحي محمد سالم** | 201900688 |  |  |  |

Project details

Make a square with size 4X4 by using 4 or 5 pieces. The pieces can be rotated or flipped and all pieces should be used to form a square. Example sets of pieces. There may be more than one possible solution for a set of pieces, and not every arrangement will work even with a set for which a solution can be found. Examples using the above set of pieces... Rotate piece D 90 degree then flip horizontal {R 90 + F H} \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Input: The first line contains number of pieces. Each piece is then specified by listing a single line with two integers, the number of rows and columns in the piece, followed by one or more lines which specify the shape of the piece. The shape specification consists of 0 or 1 characters, with the 1 character indicating the solid shape of the puzzle (the 0 characters are merely placeholders). For example, piece A above would be specified as follows:

2 3

111

101

Output: Your program should report all solution, in the format shown by the examples below. A 4-row by 4-column square should be created, with each piece occupying its location in the solution. The solid portions of piece #1 should be replaced with 1' characters, of piece #2 with 2' characters. Sample output that represents the figure above could be:

1 1 1 2

1 4 1 2

3 4 2 2

3 4 4 2

For cases which have no possible solution simply report "No solution possible". You must provide many sample inputs (many text files) during the discussion time, to test your project on many samples. Each text file represents one problem to be solved.

* Get all probability of shapes order

Shape1 Shape2 Shape3 Shape4

Shape2 Shape1 Shape3 Shape4

Shape3 Shape1 Shape2 Shape4

…

Shape3 Shape2 Shape4 Shape1

Shape2 Shape3 Shape4 Shape1

If we have 4 shapes , then we have 24 possibilities 🡺 !4

* Rotate the possibilities

We rotate the first shape and then you get all the possibilities again .

Shape1ROTATED Shape2 Shape3 Shape4

Shape2 Shape1ROTATED Shape3 Shape4

…

Shape4 Shape2 Shape3 Shape1ROTATED

Shape3 Shape2 Shape4 Shape1ROTATED

Shape2 Shape3 Shape4 Shape1ROTATED

We rotate the second shape and then you get all the possibilities again .

Shape1ROTATED Shape2ROTATED Shape3 Shape4

Shape2ROTATED Shape1ROTATED Shape3 Shape4

…

Shape2ROTATED Shape3 Shape4 Shape1ROTATED

The same way to get all the possibilities for the third and fourth

We start making more than one square of these possibilities

* Put the shape in the box

A picture containing shoji, building

Description automatically generated

Move step by step inside the box , When we reach a free space, we start by examining the places corresponding to the area of the shape in the square

Chart, box and whisker chart

Description automatically generated

ex.

1. We want use this shape in the previous square

A picture containing shoji, crossword puzzle, building, shrimp

Description automatically generatedWe find that the first cell in the box is empty , We specify the places that the shape will occupy in the box, starting with the cell we found .

A picture containing building

Description automatically generatedIf we find that the shape will be completely added, we add it .

2. We want use this shape in the previous square

A red and white flag

Description automatically generated with medium confidenceWe will find that the first cell in the box is occupied, so we will move to the next

A picture containing shoji, building, clipart

Description automatically generatedWe find that the second cell in the box is empty , and we specify the places that the shape will occupy in the box, starting with the cell we found .

A red and white flag

Description automatically generated with low confidence

We find that the shape will not be added because of the last cell .

A picture containing shoji, building

Description automatically generated

Now we need to move to a new cell, which is the third cell, which is empty .

we find that the shape will be completely added , So this is the right place .

We repeat this process with all the shapes until the square is filled. If it is not filled, we try other ordinal possibilities until it is filled. If it is not filled, we find that the given set of shapes cannot be a 4 x 4 square of them.

* Gui Description

A picture containing graphical user interface

Description automatically generated

add shapes id in every owns field and add name of square

then press add , it will be add to database

Chart

Description automatically generated

fill the field with id of shapes that you want to test , to know If there is a square 4×4 formed of them

You can choose a pre-registered square shape to test it.

Just click on the list and choose the name of a square from them and then click on “Result”.

Diagram

Description automatically generatedClass diagram

s

Diagram

Description automatically generated

Diagram

Description automatically generated

Graphical user interface

Description automatically generatedDatabase schema

Design pattern

Name : singleton pattern

Singleton pattern is one of the simplest design patterns in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

This pattern involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.

Diagram

Description automatically generatedImplementation

We're going to create a *SingleObject* class. *SingleObject* class have its constructor as private and have a static instance of itself.

*SingleObject* class provides a static method to get its static instance to outside world. *SingletonPatternDemo*, our demo class will use *SingleObject* class to get a *SingleObject* object.

* **It used**

In class shape we use it when we make two private constructor and we created a two methods that return object of the class type

**Text

Description automatically generated**



# Name : **Prototype**

Prototype pattern refers to creating duplicate object while keeping performance in mind. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

This pattern involves implementing a prototype interface which tells to create a clone of the current object. This pattern is used when creation of object directly is costly. For example, an object is to be created after a costly database operation. We can cache the object, returns its clone on next request and update the database as and when needed thus reducing database calls.

Implementation

We're going to create an abstract class *Shape* and concrete classes extending the *Shape* class. A class *Shape Cache* is defined as a next step which stores shape objects in a *Hashtable* and returns their clone when requested.

*PrototypPatternDemo*, our demo class will use *ShapeCache* class to get a *Shape* object.

Diagram

Description automatically generated

* **It works**

**Instead of put this function with shape class , we put it with shape former to create objects separated in every shape**

**Not only one**