1 About the Project

1.1 Objectives of the assignment

The objective of this project is for you to combine all the things we have seen this semester into a complete application. In many ways, it is similar to a 3-session laboratory assignment, but you will have much more freedom on what to do and on the way you organize your time. You will be asked to hand in stage reports presenting the work you have accomplished so far, the choices you have made, and the difficulties you have encountered. These stage reports will be graded and their grades will contribute to the global grade for the project. This will be explained in more details at the end of this document.

1.2 How the game is played

Rather than attempting to write a lengthy description of the rules, I will simply show some screen dumps of an ongoing game and explain what is going on.

At the beginning of a game, the window is filled with colored squares. The number of different colors and the dimensions of the grid are user-selectable. Figure 1 shows an example of an initial
configuration. Note that the window’s left pane displays the number of remaining squares of each color (plus other information that I will explain shortly). When the mouse’s cursor passes over a square that is connected horizontally or vertically to at least one square with the same color, then all theses connected squares get highlighted. For example, in Figure 2, the cursor is above a blue square that is connected to two other blue squares. As a consequence, the three squares get highlighted. Note that the widow’s left pane displays the number of highlighted squares as well as a potential score for this selection.

![Figure 2: Three connected squares get highlighted.](image)

The location of the block of squares and their color have no effect on the potential score. It only depends on the number of squares in the block.

![Figure 3: Six connected squares get highlighted.](image)

Two block are worth 2 points and 3 blocks are worth 3 points, but 4 blocks are worth 6 points, 5 blocks bring 11 points, 6 blocks bring 18 points, 7 blocks bring 27 points…The number of points
increases a lot faster than the number of connected blocks.

![Figure 4: Seven connected squares get highlighted.](image1)

In fact, this is the objective of the game: The player will attempt to get the highest possible number of points by removing blocks as large as possible. Now, how can a player remove a block of squares? Simply by clicking on one of its elements. For example, let us consider the configuration show in Figure 5, as the player is about to remove a block of three green squares.

![Figure 5: The board before the player removes a block of 3 squares.](image2)
When the player clicks on the block, all of the block’s squares are removed from the board, thus creating an empty space, as shown in Figure 6.

![Figure 6: Intermediate (invisible) state between Figures 5 and 7.](image)

The squares that are above this empty space simply slide down—presumably under the effect of gravity—into it to replace the squares that were removed. As shown in Figure 6, some squares will go down by only one position while others will go down by two. The new state is as shown in Figure 7. Notice the empty squares on top.

![Figure 7: The board after the player has removed the three squares.](image)
One consequence of the removal of the green block is that the red block the cursor points at in Figure 7 now counts 8 squares, one more than before the move (check it in Figure 6). We can improve the score for this red block by connecting it with the small block of 2 red squares that is to its lower right. The way to do this is by removing the black block the cursor points at in Figure 8.

Figure 8: The board before the player removes a block of 3 back squares.

After the black block has been removed, our red block now counts 9 squares, and we can bring this number to 10 by removing the three highlighted green squares in Figure 9.

Figure 9: The board after the player has removed the three black squares.
There is not much more that can be done to increase that red block. By removing it, we create a block of yellow squares.

![Figure 10: The board after the player has removed the three green squares.](image)

That block of yellow squares itself can be increased by removing two black squares that stand in the way (Figure 11).

![Figure 11: The board after the player has removed the three black squares.](image)
1.3 Handouts

The folder “Project - Part 1” contains the text of this assignment as a .pdf file and a source file named “LayoutDemo.java” that demonstrates the use of a grid layout and of Java buttons.

2 What to Do

2.1 Create a project file

Although you are probably going to re-use parts of the code you developed for Labs 07 through 10, you are probably better off starting from a new project. Keep the two targets of the project: you will need them both (and probably a couple more) before the end. If you want to use files from past labs for this project, remember to copy the file into your project folder before you add it to the project. Otherwise you will be adding an incorrect “File Path” to your project and will get penalized for that when we grade your assignment.

2.2 You’re on your own

Well, not really. I am going to give a rough breakdown of the tasks you should try to tackle first, and then it will be up to you to decide how you are going to solve the subproblems, and in which order.

- identifying blocks of connected squares that have the same color is probably the the toughest part of the assignment. I recommend that you keep it for next week. For the time being, you should probably concentrate on the other aspects of the project.

- You need to decide how you are going to implement your grid of squares. We have already seen a way to do that, by drawing squares in the frame. An alternative way of creating a grid of rectangular objects is through the use of a GridLayout in which you display JButton objects. The code of the handout shows to do this. Both approaches have their drawbacks.

- In the end you will probably have to implement a Square class. If you decide to use JButton objects, then the Square class will be derived from JButton (we will discuss that in class tomorrow).

- You should probably as well create a class SquareBlock that will store (most likely in an array, but linked lists are fine too) references to squares that form a block together.

2.3 About the partial evaluations

What are you expected to complete by next Tuesday? Obviously not the entire project. On the other hand, I would expect you to have completed the design of your classes. This means that you should have determined what information they store (what is the state of an object of each class) and what their methods are (what is the behavior of an object of each class). You should have implemented, at least partially, some of your classes.

Next week, I will give you more details on the algorithm to determine all the elements in a block. You will also be able to use class inheritance in a more efficient way.
3 Extra Credit

You can do a number of things for extra credit for this project. Here are a couple of examples.

3.1 Read and write the state to a file

A few weeks back we learned about file I/O. Now we can put this to use to write the state of your grid and recover it from a file. You may also consider saving a list of best scores.

3.2 Add sound to the game

This is not very hard. You can find lots of demo code on the web doing just that.

4 How You Will Be Evaluated

4.1 Point distribution

The maximum number of points for the project is 100, but extra points will be awarded for excellent aspects of the project or report. The point distribution for the entire project is as follows:

<table>
<thead>
<tr>
<th>Phase 1 evaluation</th>
<th>10 pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>End-of-session evaluation</td>
<td></td>
</tr>
<tr>
<td>Phase 1 report in</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Phase 2 evaluation</th>
<th>10 pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>End-of-session evaluation</td>
<td></td>
</tr>
<tr>
<td>Phase 2 report in</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Execution of the Complete Project</th>
<th>25 pts</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Final Source Code</th>
<th>5 pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good indentation and general readability</td>
<td></td>
</tr>
<tr>
<td>Javadoc headers</td>
<td>5 pts</td>
</tr>
<tr>
<td>Judicious comments well positioned in the code</td>
<td>5 pts</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Final Report</th>
<th>10 pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discussion of important aspects of the project</td>
<td></td>
</tr>
<tr>
<td>General quality of the writing and presentation</td>
<td>10 pts</td>
</tr>
</tbody>
</table>

4.2 Various point penalties

Hopefully we won’t have to apply many of these:
Project left accessible on the workstation -5 pts
Project folder incomplete or not properly cleaned up -5 pts
Incorrectly configured project file -5 pts
Report file missing from the project folder -5 pts

**Late penalties**

Project folder (uploaded to EnVision server), per day late  -10%

If you submit a project late, then it is your responsibility to notify the TA (with CC. to the instructor) that the project is finally available for download on the EnVision server. If you fail to do so, then the “late penalty clock” will keep ticking until the TA gets around to checking your folder on the EnVision server and notices your project. Unless specifically asked to do so, do not mail your project folder as an attachment.