|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Comments** | **Mark** |
| 587554 | M. Li | A rather crude class diagram. Not clear about the functionality of the Game class. The description of the shuffle algorithm is OK but use of pseudo code would be better. I wasn’t sure what the Grid class was. Some details as to how sub blocks within the Bitmap image were to be handled would have been useful. This is a key consideration in implementing the game.  No functionality as program crashes before it can load a picture. Some code submitted which has reasonable commenting | 13/25  10/75  Total 23/100 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Comments** | **Mark** |
| 1206791 | C. Yan | All your functionality is in a single class – NewGame. A much more object oriented solution is to think about how you represent the image as a group of sub-image blocks which can be shuffled. It’s better to delegate this functionality to a separate class. No detailed design of the shuffle algorithm but you have the correct idea on how to shuffle an image so it can be successfully re-shuffled. A pseudo-code description of how to achieve this would have been helpful in your report. | 12 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Comments** | **Mark (/25)** |
| 1202636 | B. Cao | No class diagram produced. Your discussion about moving image blocks doesn’t contain any idea about how to shuffle the image so that the unshuffled image can be restored by exchanging adjacent image blocks. This is important to think about. Your flow chart on the last page doesn’t convey any meaningful information. | 11 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Comments** | **Mark (/25)** |
| 1020922 | S. Albuasais | A good effort with the report. The idea of having a separate image tile class is a good one. You didn’t show any class diagram showing how your classes interact. No need for an event timer as the game is played under user interaction. You haven’t discussed how to shuffle an image so that the unshuffled image can be restored by exchanging adjacent image blocks. This is important to think about. | 15 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Comments** | **Mark (/25)** |
| 1206792 | Y. Shen | Your class diagram is not clear. What is swapping\_pieces? It looks more like a class method than a class. You have the idea about numbering image blocks but you haven’t discussed how to shuffle them in such a way that they can be unshuffled by swapping adjacent blocks. This is an important consideration. Your flow chart doesn’t convey any meaningful information. Your report lacks a lot of detail as to how you are going to go about designing and implementing your program. | 9 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Comments** | **Mark (/25)** |
| 1103270 | N. Khnouf | Use-case diagram not really necessary. Not a detailed enough class diagram. No clue as to how the game class is structured. The key issue of representing an image as a set of sub-images has not been addressed. No need for a state diagram. You have missed out details of the shuffle algorithm which is key to implementing a working program. | 9 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Comments** | **Mark (/25)** |
| 1133166 | Z. Chen | The flow diagram on the first page doesn’t convey any useful information. Your discussion about shuffling misses out the key point about how a shuffle algorithm can be implemented so that it can be reversed by swapping adjacent image sub-blocks. Representing an image sub-block by a cell class is quite sensible although your design diagram doesn’t specify the functionality of this class. There seems to be mixture of classes and methods in this diagram. | 12 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Comments** | **Mark (/25)** |
| 1167952 | Dk Noralam | No complete design diagram but sensible to have an image tile class. The relationship to neighboring tile objects is important to think about. Also you haven’t mentioned your shuffle algorithm. The key point is how a shuffle algorithm can be implemented so that it can be reversed by swapping adjacent image sub-blocks. | 13 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Comments** | **Mark (/25)** |
|  |  |  |  |