Team	Project Grade	Feedback
s182081	17	The program works as expected. The purpose of the class PlacedPiece is unclear, as it acts as an encapsulation for Piece but give complete access to the underlying Piece anyways. Working with the class Piece directly would have work and was a simpler option. Otherwise the code is well structured, modular, easy to read.
s182498	19	The program works as expected and the code is clear, modular, and easy to read.
s194923	19	Program works as expected and performs quite efficiently. The code is modular and well structured, although the overly verbose comments makes it hard to read.
s200933	16	The classes were not part of the expected package "be.uliege.montefiore.oop". Otherwise the program works as expected. The code is modular and well structured.
s214826	15	The structure of the archive does not match the requirements of the statement. It should be organized with a package structure of the form "be.uliege.montefiore.oop" so that it can be compiled and run with the provided commands. Your program fails to detect when the input files contains too many pieces. But it works fine otherwise. The code is clear and well structured.
s218277	18	The program works as expected and the code is clear and modular.
s220065	13	The structure of the archive does not match the requirements of the statement. It should be organized with a package structure of the form "be.uliege.montefiore.oop" so that it can be compiled and run with the provided commands. Your program crashes on input file with invalid header, but works well otherwise. The code is clear and has some structured. The parsing part could have been isolated from the main method for better modularity.

s222125-s2401880	14	Your program fails to detect when too many pieces are given in the input file. Also, when it finds a solution, the display does not match the requirements of the statement. The code lacks modularity. You should not place in the main class the parser and the solver. These two components of the project could be made much more independent. It is a detail, but the method Board.sidesMatch() considers that two FLAT inside the puzzle is a valid configuration, while the statement only considered BUMP and PITS inside the puzzle.
s222271	17	The code is clear, modular and well commented.
s222601	14	Your archive does not have the expected structure, and your main class does not compile directly (it missed a ';'). Always test your code from the archive before submitting it. Otherwise your program works as expected. The code is clear and well commented. The parsing part could have been isolated from the main class for even better modularity.
s223067-s213326	8	The archive does not match the package structure implied by the statement. Your program fails to properly detect many invalid input files. It also fails to solve puzzle where at least one of the dimension is equal to 1. On other valid instances, it works as expected. The code is not modular at all. Most of the work is performed in the main class in a static approach. Your project does not really respect the principles of Object Oriented Programming. The parser and the solver should at least be in separated classes.
s225049	18	Your program works as expected. The code is clear and modular, but some commenting would have been welcomed.
s225306	15	Your classes should belong to the package be.uliege.montefiore.oop in order to be compiled using the command provided in the statement. Your program fails to detect many invalid input files, and when it does, the message is not very precise. On valid puzzle, your program works well and is quite efficient. The code is clear and well structured.

s225656	17	The program works as expected except when the puzzles have one of their dimension equal to 1. The code is modular and well commented. It is however a bit complicated, especially with these classes containing one inner class. I believe you could simply merge the inner class with the container one to make it simpler.
s226111-s2306987	13	Although your classes belong to the expected package, your archive is not structured accordingly to the package definition, which prevents the expected compilation command to work directly. The program fails to correctly detect some invalid input files (empty file or invalid header for instance), and wrongly consider puzzles with one of the dimension equal to 1 as invalid. The code is clear and well structured.
s228617	14	Although your classes belong to the expected package, your archive is not structured accordingly to the package definition, which prevents the expected compilation command to work directly. Your program works as expected, except that it finds a solution to a square puzzle with a piece containing 3 FLAT edges. The code is easy to read and has good modularity.
s2300387	17	The program crashes on puzzle where one of the dimension is equal to 1. Otherwise, it works as expected. The code is clear and well structured.
s2300563-s2306687	17	The code is clear and well structured, although it could have been even better to isolate the solver part more clearly from the main Puzzle class.
s2300717	18	The program works as expected, and the code is clear and well structured.
s2300983-s226324	12	Your program fails to detect properly most invalid input files. The final display of a solution is not exactly the one required. Also, your program prints a lot of debug info for no good reason on a submitted version. Clean your project and make it work as asked in the statement before submitting it! It also fails to properly parse puzzle when one dimension is greater or equal to 10 (it only expects one digit for the height/width). The code is quite easy to read and has good structured. The idea behind the Group class would be very relevant on big puzzles, it is unfortunate your code does not parse such big puzzles...

s2301633-s2401154	13	Your program fails to detect many invalid input files, and when it does, it still throws an exception instead of cleanly exiting the program. It works as expected on valid input files. Your code is easy to read, but lack modularity. The class Puzzle does almost all the work.
s2301643	18	Your program works well, except that it considers puzzles where one of the dimension is equal to 1 as invalid. The code is clear and modular. I would only argue that in the class Board, instead of having two methods "canPlacePiece()" and "placePiece()", you could merge them as one method, which returns a Boolean to say whether the piece has been placed or not. In your code, you give the possibility to place a piece at an invalid position.
s2301654	16	Your program wrongly detects puzzles where one of the dimension is equal to 1 as invalid. It otherwise works as expected. The code is clear and has good modularity.
s2301801-s2300530	17	Your program fails to detect invalid header or empty files. It also fails on puzzle where one of the dimension is equal to 1. It works well on other valid instances. The code is clear and modular. Some real effort have been made on the comments and documentation.
s2301809-s2301186	14	The structure of the archive does not match the package organization of your project. Your program somehow finds a solution to a 4x4 puzzle with a piece containing 3 FLAT edges. There is also a bug in your display method (the height and width should be inverted) which prevents the entire solution to be displayed when the height and the width are not equal. The code is otherwise clear and simple, with good modularity.
s2301812-s2403408	15	The program does not detect when too many pieces are provided in the input file. It otherwise works as expected. The code is clear and well structured, although some commenting would be appreciated. The method Board.place() could first call Board.canPlace() to forbid the user from placing a Piece at an invalid position, to ensure the integrity of the Board.
s2302003	18	The program works as expected. The code is a bit overly complex, but is very well documented, and shows good usage of many OOP principles.

s2302095	15	The program does not detect when too many pieces are provided in the input file. It otherwise works as expected. The code is clear and well structured, although some commenting would be appreciated. The method Board.setPiece() could first call Board.isValidPlacement() to forbid the user from placing a Piece at an invalid position, to ensure the integrity of the Board.
s2303071-s2301832	19	The program works as expected and the code is clear and is well structured.
s2304697-s2301282	11	Your program fails to properly detect many invalid input files, and also wrongly consider puzzle where one of the dimension is equal to 1 as invalid. The code lacks modularity. Most of the work is done by the class Puzzle alone.
s2304788	16	The program works as expected on valid inputs. It detects correctly invalid input files, but it would have been better to not propagate the DataError up to the main method, but exit the program cleanly simply with the error message printed. The code is well structured, but some commenting would have been welcomed, especially for the solver part.
s2305323	17	Your archive contains way too many unnecessary files, only submit what is needed. Also your classes do not belong to the expected package (it misses the prefix 'be'). Otherwise, the program works as expected. The code is a bit complex, but well structured, and very well documented.
s2305452	18	The program works as expected, and the code is clear and well structured.
s2306665	15	Your program fails to detect most invalid input files. It otherwise works correctly on valid input. The code is clear and well structured. I would argue that the method Board.insertPiece() should call first the method Board.canPlace() so as to forbid a Board to be in an invalid state.
s2400109	18	The program works as expected. The code is clear and well structured.
s2400176-s2400739	17	The program works well and is quite efficient. It however wrongfully finds a solution to a 4x4 puzzle containing a piece with 3 FLAT edges. The code is clear and has good modularity.

s2400691	15	Your program fails to solve puzzle where one of the dimension is equal to 1. The error messages are sometimes unclear (e.g., "Error: null" or "Error: 2"). It works well on other valid input. The code is modular, but sometimes overly complex, without many comments to ease the reading.
s2400880-s2405314	16	The program works as expected, although it would have been better on error to not exit with an exception but terminate the program normally and simply display the error message (the internal use of exception is a good thing). The code is a bit complicated, but has good modularity and the comments help the reading.
s2400887-s2302116	12	Your program fails to detect many invalid input files. The code is very clear, but lacks modularity. It is best to separate the major components into different classes, instead of having the class Puzzle do all the work.
S2400964	12	The archive does not respect the expected package structure. The code cannot be compiled using the command provided in the statement. Your program fails to detect many invalid input files. It works fine on valid inputs. Your code is clear and modular, although some commenting would have been welcomed. Also, why are the instance variables in the Solver class protected instead of private?
s2401814	17	Your program works as expected. The code is clear and has good modularity.
s2401900	20	The program works as expected and is quite efficient. The code is well structured and well commented.
s2401908-s2401901	6	Your program simply does not work. It sometimes loops infinitely on unsolvable puzzles (printing the same error message again and again). It does not manage to solve simple puzzle and consider them unsolvable, or even sometimes crash on them. The code is way too complex. I think you tried to solve this problem very efficiently, without having a simple working solution first. You tried to use as many OOP concept as possible, which could be a good thing, but not if it comes at the price of completely breaking your program, and making the code much harder to debug. The overall structure is a good basis, but should be made simpler, and there is a real effort on commenting.
s2401941	20	The program works well and the code is clear and well structured.

s2402075	16	Your program detects most types of invalid input files, but still crashes afterwards instead of cleanly exiting the program. It works well on solvable instances, but it would have been better to explicitly print something when your program detects that a puzzle cannot be solved. The code is easy to read and has good modularity.
s2402267-s2401536	15	Your program fails to detect some invalid input files (invalid header or too many pieces in the description). It also fails to solve 4x1 and 1x4 puzzles for instance (but solves the 1x1). It works well on other valid instances. The code is clear, well documented and well structured. I would only argue that the method Board.place() should first call Board.canPlace() to ensure that the board cannot contain adjacent pieces that do not match together.
s2402388	17	Your program works as expected, except for valid puzzles of size Nx1 or 1xN (it works for the 1x1 puzzle though). The code is simple, well commented and has good modularity.
s2402409-s2400966	11	Your program fails to properly handle some invalid input (no input file, empty file, or invalid header). It otherwise works as expected. Your code is clear and easy to read, but lacks modularity. Almost all the work is done by the class Puzzle in a *static* approach, which does not respect the OOP principles.
s2402496-s2401300	5	Your archive does not match the required package structure. Furthermore, your program simply does not work. It detects many invalid input files, but still crashes with a NullPointerException instead of cleanly exiting. Overall it crashes over most valid input files (regardless whether they are solvable or not). Your code is way too complex. The Solveur.solve() method is 700+ lines long. This makes it too hard to read and to debug. You probably tried to have a fancy solution directly, without first designing a simple solution that works. Try to decompose the problem into smaller parts (different shorter methods in the class Solveur) that can be tested individually. The overall approach respects the principle of OOP, but is too ambitious.
s2402498	20	The program works well and fast. The code is clear, well commented and has good modularity.
s2402855-s2400965	14	The program works well on valid inputs, but does not handle all invalid inputs with an error message and a clean exit. The code is simpler but lacks modularity. Most of the work is done by the class Puzzle alone.

s2403101-s2400353	16	Your program fails to properly detect some invalid input files. It works as expected on valid input. The code is clear and modular.
s2403372	13	Your program fails to detect most invalid input files, prints an incorrect reason, or gives the correct reason but still crashes afterwards. It also fails on puzzles where one of the dimension is equal to 1. It works well on other valid inputs, although the indices of the pieces should start at 1 and not 0. The code is clear, but not really modular. Almost all the work is done by the class Puzzle alone.
s2404244-s2401958	16	Your program crashes on empty input files, and when the header is invalid. It works as expected otherwise. The code is clear, some work is delegated to the different Piece classes, but too much is done by Puzzle alone. At least try to separate the parser from the solver to improve the modularity of your code.
s2404442	16	Your program works well, except that it considers puzzles where one of the dimension is equal to 1 as invalid. The code is easy to read and has good modularity. It would have been better to ensure the integrity of the class Board by enforcing that the public method Board.place() fails if Board.canPlace() returns false on the same arguments.
s2404678-s2400866	18	Your program works well, except that it fails to solve puzzles where one of the dimension is equal to 1. The code is clear, modular, and very well documented.
s2404805	16	The program works as expected, except that it fails to detect invalid files where the number of pieces does not match the dimension of the puzzle given in the header. The code has good modularity. Some instance and class variables have incoherent visibility: the public final variables in the private inner class PiecePlacement, the package private variable in the class Solver, or the public and final variable PuzzleResult.data. For the latter, be especially careful because the final keyword does not protect the content of the array data.
s2404824	16	Your program fails to properly detect some invalid input files, and also fails to solve puzzles where one of the dimension is equal to 1. It works well on other valid inputs. The code is clear and modular. The design of the class Board can be improved to ensure its integrity: the method Board.place() should first call Board.canPlace() to make sure you don't create a clash between placed pieces on your board.

s2404848-s2402742	14	Your archive does not match the required package structure. The program correctly detects invalid input files. However it fails to solve a 1x4 puzzle (but solves a 4x1). It also consider invalid puzzle where one of the dimension is greater or equal to 10 (it only expect a single digit for the dimensions) . The code is clear and well structured.
s2405239-s225089	16	Your program fails to correctly detect many invalid input files, or gives an unclear explanation (e.g., "Erreur fichier invalide: null" for an invalid header). It works as expected on valid input files. The code is clear and has good modularity.
s2405429	16	Your archive does not match the required package structure. The program works as expected. The code is clear and well structured.
s2405458-s2401624	17	Your archive does not match the required package structure. The program works as expected. It would have been best to simply print a clean error message for invalid inputs, and not print the entire stack trace. The code is clear, well structured, and very well documented.
s2406337-s2400700	19	The program works as expected. The code is clear, very modular, and very well documented.
s2500726-s224693	15	Your program fails to detect some invalid input files, and also sometimes fail on valid puzzles (from what I have seen in the code, it may come from the part of the solver that deals with the interior of the puzzle). The code is otherwise well structured, although a bit complex (which makes it harder to find the bug mentioned beforehand).
s2503328	14	The program works as expected. The code is clear, although some commenting would have been welcomed. It is not very modular, with the class Puzzle doing almost all the work. The parser and the solver should not be the responsibility of the same class.
s2505875	15	Your program works as expected, except that there is no space between the piece index and the rotation in the displayed solutions. The code is clear and modular. I would only argue that the public method Board.placePiece() should call Board.canPlace() so as to ensure the integrity of the instances of Board.
s2506343	15	Your archive does not match the required package structure. Your program fails to properly handle some invalid input files, or gives very little information on the reason. It otherwise works as expected on valid input. The code is clear, modular, and well commented.

s2506553	19	The program works as expected. The code is clear, well structured, and very well documented.
s2507139	17	Your program works as expected, except for puzzles of the form Nx1 or 1xN which are wrongfully considered invalid. The code is modular, although there are maybe a bit too many classes for the size of this project.
s2507155	15	Your program solves invalid puzzles (with a piece containing 3 FLAT edges, when all the dimensions are greater than 2). The solutions are not displayed as required in the statement. It otherwise works well. The code is clear and modular, although the integrity of the class Board could be improved by requiring that the method Board.place() first check that Board.canPlace() returns true.
s2507556-s181386	17	Your program solves invalid puzzles (with a piece containing 3 FLAT edges, when all the dimensions are greater than 2). It works as expected on most invalid puzzles and valid puzzles. The code is clear and modular.
s2509728	16	Your archive does not match the required package structure. The program works as expected. The code is modular, but some commenting would have been welcomed.
s2509779	17	Your program works as expected. The code is clear and modular. The integrity of the class Board could be improved by enforcing Board.putPiece() to first ensure Board.isFit() returns true, so as to not allow a Board to contain adjacent clashing pieces.