# Assessment Checklist

You are required to implement the program in an object oriented manner. The solution must demonstrate each of the following, shown in the tables below. To help you to ensure that you have covered all the points required in the implementation of the assessment please complete the tables below for each point. Include examples, observations, and justification where appropriate.

## Design requirements:

|  |
| --- |
| **Abstraction, encapsulation and information hiding used where appropriate** |
| Examples: Tile is an abstract class which contains private and protected variables. The X and Y axis variables in the Tile class can only be accessed through their respective get/set methods |

|  |
| --- |
| **Inheritance used if appropriate to the solution** |
| Examples: Ground, Wall, Diamond, and ElementMoveable inherit from Tile. Crate and Warehousekeeper inherit from ElementMoveable. |

|  |
| --- |
| **Polymorphism used if appropriate to the solution** |
| Examples: Polymorphism used to store the various subclasses of tile in the Tile Arraylist “MapRows” |

|  |
| --- |
| **All class-wide variables are private to prevent content coupling class-wide variables are kept to a minimum to ensure a minimum of common coupling** |
| Examples: The tileShape variable in the Tile class is private, and can only be accessed through the getTileShape method so the tile shape is not accessed directly. |

|  |
| --- |
| **Data coupling is used (parameter passing) in preference to content or common coupling** |
| Examples:  The main constructor for Tile passes the tempType String variable to the setNodeType method, rather than having the method access it directly. |

|  |
| --- |
| **Program does not contain a lot of unnecessary data coupling** |
| Examples: |

|  |
| --- |
| **Classes are highly cohesive** |
| Examples: Each class serves an individual purpose, and the methods within all lead towards that purpose. The Map Class only deals with reading a map from file, and converting that map into a playable form. This shows high cohesion as all the functional methods lead to the one goal of the class. |

## Implementation requirements:

|  |
| --- |
| **A working solution which meets the requirements of the given brief** |
| Comments: |

|  |
| --- |
| **Variables are correctly declared and initialised** |
| Examples:The level, Filerows, MapRows, and warehouseKeeper are declared and initialised at the start of the Map Class |

|  |
| --- |
| **Arithmetic and/or logical operators are used correctly** |
| Examples:Arithmetic operators used to modify the x/y coordinates in the ElementMoveable class |

|  |
| --- |
| **A range of control structures are implemented correctly** |
| Examples: A For loop is used in the Map Class to create the array of Tile objects, and a While loop is used to add each line from the level file to the FileRows arraylist |

|  |
| --- |
| **At least two data structures are implemented correctly** |
| Examples: The ElementMoveable class contains an ArrayList of int Arrays to store the coordinates of diamonds covered by crates |

|  |
| --- |
| **The program contains a minimum of four classes, which contain attributes, methods and a constructor method** |
| Examples: 12 classes total, not all yet implemented. Crate, Diamond, Ground, Wall,WarehouseKeeper,ElementMoveable, and Tile all contain constructor methods. |

|  |
| --- |
| **A minimum of three objects are created from the classes, with appropriate initial attribute values set through the constructor methods** |
| Examples:An indeterminate number of Tile objects (or more specifically objects which are created from subclasses of Tile) are created in the Map class, and are initialised with coordinates, a type, and an image. |

|  |
| --- |
| **The program contains at least one overloaded method (this may be a constructor method)** |
| Examples: The Tile class contains an example of constructor overloading. The constructor without parameters shown would create a default floor tile. |

|  |
| --- |
| **Classes are linked appropriately through association, aggregation or inheritance relationships** |
| Examples: Crate inherits from ElementMoveable, as it is a moveable element, and ElementMoveable in turn inherits from Tile as it is a tile with extra functionality |

|  |
| --- |
| **Parameters are passed correctly both within and between objects** |
| Examples: Each of the subclasses of Tile passes the appropriate parameters to the superclass, and the constructor in the Tile class passes a String parameter on to the setNodeType method. |

|  |
| --- |
| **Appropriate access types are defined for methods, attributes and classes** |
| Examples: Most Methods within classes are private other than get/set classes, and those which must be accessed by another class (get/set methods and constructors) |

|  |
| --- |
| **Use of pre-defined classes and/or methods from the standard object library** |
| Examples: BufferedReader Class used from the Java IO library to read the map files |

|  |
| --- |
| **The program appropriately handles errors with exceptions or pre-validation** |
| Examples: A Try-Catch is used in the Map class to catch errors in loading or reading the map files |

|  |
| --- |
| The program code is commented appropriately throughout |
| Examples: Commented where appropriate |