1. UML 18 marks
   1. Good coding practices: 1
   2. 4 classes (Animal, Mammal, Dog, PetOwner): 4\*0.5 = 2
   3. Making Animal and feedMe() abstract: 2\*1 = 2
   4. Having all the methods and variables from the UML diagram: 2
   5. feedMe() is implemented either by Mammal or Dog: 1
   6. Both the multiplicities are appropriately captured in the code: 2\*2 = 4
   7. The 1..\* multiplicity is appropriately validated in the code: 2
   8. The two-way association is implemented in Dog and PetOwner by creating an instance variable in those two classes: 2
   9. Demonstrate that the two way association is implemented by having the method calls on the class instance variable: 2

1. Singleton 15 marks
   1. Good coding practices: 1
   2. A comment which mentions singleton design pattern should be used and the reason for it. 2\*0.75 = 1.5
   3. Three fields in the FileManager class (fileID, fileName, fileManager - this should be static). 3\*0.5 = 1.5
   4. Private constructor 1
   5. fileID and fileName should either be initialized to default values or should not be changed after the first value assignment. 1
   6. Correct implementation of getInstance() so that it returns the same instance everytime. 2
   7. Using the override annotation on the equals method. 1
   8. Correct implementation of the equals method.
      * Check whether or not the argument of equals is null 1
      * Check whether or not the types are the same using getClass() 1
      * Typecast the argument of the equals method to FileManager 1
      * Compare the object states using the fileID and fileName values 1
   9. Create at least two objects of FileManager and show that they are equal 1+1 = 2
2. Complex numbers 20 marks
   1. Good coding practices 1
   2. Create the ComplexNumber class with two fields - real and imaginary 1
   3. ComplexNumber should implement the Serializable interface 1
   4. ComplexNumber class should have the serialVersionID field initialized to any long value 0.5
   5. ComplexNumber class should have the add, subtract, multiply, divide and conjugate methods. 0.5\*5 = 2.5
   6. All these 5 methods should return a ComplexNumber type and have one argument of type ComplexNumber 0.5+0.5 = 1
   7. Correct implementation of the add functionality 0.5
   8. Correct implementation of the subtract functionality 0.5
   9. Correct implementation of the multiply functionality 0.5
   10. Correct implementation of the conjugate functionality 0.5
   11. Correct implementation of the divide functionality 1.5
   12. Multiply and conjugate methods should be called in the divide method 0.5
   13. Create two lists/sets with 10 complex numbers each (using random number generator) 0.5+0.5 = 1
   14. Create four lists to hold the result of each of the four operations (add, subtract, multiply, divide) 4\*0.5=2
   15. ObjectOutputStream needs to be used to save the objects to file. For example, the values in the object should not be saved in a text file. 2
   16. Write the output of the 4 lists for each of the 4 operations   
       to 4 different files. 4\*0.5 = 2
   17. Appropriate exceptions should be handled or thrown (FileNotFoundException and IOException) 0.5+0.5 = 1
   18. All the files should be closed in the finally block 0.5
   19. Before closing the file, check whether or not it is null 0.5

4) SignUp Exception: 17 Marks

* Good Coding Practices 1
* A comment which mentions using flyweight design pattern and the reason. 2\*1 = 2
* New Exception: extending exception class; constructor ; and toString method. 3\*1= 3
* Flyweight class implementation: data members (1), constructor (1), get instance function (2), throwing exception (1), toString (1) 6
* Driver class implementation: adding unique instances beforehand (1), taking username and password as input (1), trying to create a new instance (1), while loop (1) for repeated entry until unique credentials are provided, declaring main function throws SignUp exception (1) 5

5) Outer Product: 20 Marks

* Good Coding Practices 2
* Correct Execution (if outputs are just printed on console, give 7.5 marks. If wrong output, but still matrices, give 5 marks) 15
* Exception Handling either through “throws” or “try-catch” block 3

NOTE: Don’t give zero for compilation error due to file path. Correct the file path and then check the code.

6) Generic Concatenation 15

* Good Coding Practices 2
* Used Generics (implemented some other way, give 2 marks) 3
* Right Generic class (members (1), constructor (2), add function (2)) 5
* Driver class demonstrates both the cases 2x2.5=5