**Instituto Superior de Formación Técnica N.º 151   
Carrera: Analista de Sistemas  
  
Algoritmos y Estructuras de Datos II.  
1 Año. FINAL**Una Empresa nos convoca para armar una APP que controle el Clima de un Datacenter. La misma deberá tener registro (Actual e Histórico) de: Temperatura, Humedad, permitiendo controlar el “Clima” a travez de una API de Terceros (MS-Forecast) que se invocan desde la APP. La APP será x consola a efectos del Final.

La APP MS-Forecast posee las siguientes Interfaces:

* Up\_Temp( int X)
* Down\_Temp( int X)
* Up\_Humity( int X)
* Down\_Humity( int X)
* Read\_Temp
* Read\_Humity

EL componente MS-Forecast se invoca Directamente (se sugiere Construir un MOCK para simularlo)

Desarrollo del Final:

**1.Marco Teórico:**Sobre el Problema Presentado Relacionar:

* Relacionar UML, S.O.L.I.D. GRASP, en el Marco del Diseño de Software.
* Explicar el Polimorfismo basado en Interfaces

**2. Marco Practico Modelado:**Sobre el Problema Presentado Realizar:

* Realizar2 casos de uso.
* Realizar el Diagrama de Clases.
* Realizar 1 Diagramas de Secuencia.

**3.Marco Practico Software**Sobre el Problema Presentado Realizar:

* Desarrollar un APP que contenga las Clases Anteriores que además permita guardar Históricos de Temperatura y Humedad. Además enviar alertas x mail.  
   *Nota: Se recomienda realizar una estrategia de “buenos tiempos de resolución” para poder completar lo solicitado, para aprobar deberán tener los Marcos 1,2 y 3 Aprobados, el Marco 3 “Deberá” funcionar y Compilar “Exitosamente” para ser considerado Aprobado, favor “responder” Estrictamente a lo Solicitado.. El Alumno que apruebe el Marco 1 y 2, pasará a otro “Classroom 2” para defender el “Marco 3”, debiendo esperar para la Devolución Correspondiente.*  
  La Cátedra.  
    
  **Respuestas:**1 – Las herramientas de UML, SOLID y GRASP nos ayudan a modelar y comprender el software que armamos, en los esquemas de UML podemos utilizar los varios diagramas para tener una visión e idea de cómo se va a estructurar nuestro programa, permitiéndonos así adelantarnos a implementar las prácticas que proveen SOLID y GRASP para que nuestro código sea más limpio, utilizando los patrones y la filosofía del bajo acoplamiento y alta cohesión, separando mejor las responsabilidades y capacidades de las clases y métodos que vamos a utilizar.  
    
  2 – El polimorfismo, las interfaces y los métodos abstractos permiten crear un programa que sea fácilmente reutilizable, ya que se puede implementar de diferentes maneras el mismo código sin necesidad de hacer todo desde cero; esto además permite la fácil integración de una o más clases en diferentes aplicaciones.