1. Ud. en este curso aprendió una serie de técnicas orientadas a enfrentar los principales desafíos de la ingeniería de software. Para cada una de las siguientes técnicas indique en no más de 4 líneas cuál es la problemática principal que ella aborda.
   a) Relatos de usuario
      Aborda el problema de lograr construir el software que el usuario realmente quería o necesitaba

   ---

   b) Scrum
      Al igual que la mayoría de los procesos ágiles permite aproximarse a la solución iterativamente con la participación del usuario evitando "sorpresas" y manteniendo al proyecto dentro del plan inicial

   ---

   c) Sprint review
      El Sprint review permite evaluar lo logrado con los interesados para dar la oportunidad de redireccionar o cambiar el plan inicial a partir del siguiente sprint

   ---

   d) Diagramas UML
      Los diagramas UML sirven como standard de comunicación visual y como lenguaje gráfico de alto nivel para razonar sobre el diseño del software

   ---

   e) Patrones de diseño
      Los patrones de diseño permiten reutilizar soluciones que han demostrado ser efectivas en mejorar la extensibilidad y mantenibilidad del software sin tener que redescubrirlas una y otra vez

   ---

   f) Tests unitarios
      Los test unitarios permiten por una parte verificar la correctitud de las piezas menores del producto y por otro sirven también como una manera de entender o aclarar en forma más precisa los requisitos de ese elemento

   ---

g) Estimaciones de esfuerzo

La estimación de esfuerzo es necesaria para, junto con la estimación de tiempo, poder dimensionar el número de personas que deben constituir el equipo de desarrollo

---

h) Arquitectura de Microservicios

Permite dividir la tarea de desarrollo en pequeñas unidades que pueden ser desarrolladas y probadas en forma completamente independientes. También permite reemplazar estas unidades con facilidad en el futuro

---

i) Devops

Permite acelerar dramáticamente el tiempo en que un nuevo feature o producto se pone en producción. Esto le da a una empresa una gran ventaja frente a sus competidores

---

j) Métodos estadísticos

Los métodos estadísticos permiten introducir análisis cuantitativo y tomar decisiones en base a probabilidades. El uso de distribuciones como la normal, la binomial o la de Poisson permiten razonar cuantitativamente sobre variables de desarrollo.

---

2. En cada caso indique cuál es el patrón de diseño que mejor aplica al problema descrito. No necesita justificar.

a) Se desea simplificar un sistema complejo dando una interfaz simple para los programadores que no requieren saber todos los detalles

Fachada (facade)

---

b) Hay un procedimiento que tiene siempre 10 pasos pero la forma de implementarlos puede ser distinta dependiendo del caso

Método plantilla (template method)

---

c) Hay una serie de objetos que deben estar atentos y reaccionar a los cambios que se producen en otro objeto

Observador (observer)

---

d) Quiero poder ser capaz de recordar acciones para ejecutarlas posteriormente en la misma secuencia o en secuencia inversa

Comando (command)

---

e) Dispongo de una clase Barchart que implementa gráficos de barras pero quiero poder configurarlo en runtime para que agregue leyenda, ejes, fondo, etc.

Decorador (decorator)

---

3. Para cada una de las afirmaciones indique si es verdadera o falsa. En el caso que sea falsa justifique brevemente la razón.
   a) Dada una estimación de esfuerzo E medida en meses hombre, y un equipo de 3 personas, el tiempo de desarrollo será aproximadamente E/3
   Falso. Recordar "El mítico hombre mes". El tiempo de desarrollo no decrece linealmente al aumentar el número de personas

   ---

   b) Sobreestimar y subestimar el tiempo de desarrollo son igualmente problemáticos
   Falso. Es mucho peor subestimar el tiempo de desarrollo que sobreestimarlo

   ---

   c) Dada una meta de tiempo de desarrollo X y una estimación 1.1X es razonable establecer el compromiso
   Verdadero

   ---

   d) Los puntos de función corresponden a una métrica de tamaño
   Verdadero

   ---

   e) Al hacer la planificación de un sprint es razonable usar un modelo lineal entre tamaño y esfuerzo
   Verdadero

   ---

4. a) (5 pts) En clases el profesor comentó que los patrones factory method y template method eran "similares" ¿Cuales son los fundamentos de esta afirmación ?

En el patrón template method una clase abstracta define un procedimiento con una serie de pasos que pueden especializarse en las clases concretas. En el patrón factory method este procedimiento abstracto corresponde a la creación de objetos y al igual que en el otro patrón las subclases se encargan de especializar la creación de objetos

---

b) (5 pts) Qué es lo que debe proporcionar una implementación adecuada del patrón singleton en cualquier lenguaje de programación que se

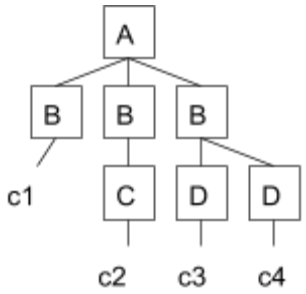
Asegurar que en todo momento existe una sola instancia de la clase singleton

Proporcionar un acceso global a esa instancia única desde cualquier parte del código

---

c) (10 pts) Un documento xml tiene estructura de árbol. Un elemento xml (etiqueta) puede contener otros elementos xml o bien solo contenido. Por ejemplo, el árbol de la figura a la derecha corresponde al documento xml de la izquierda

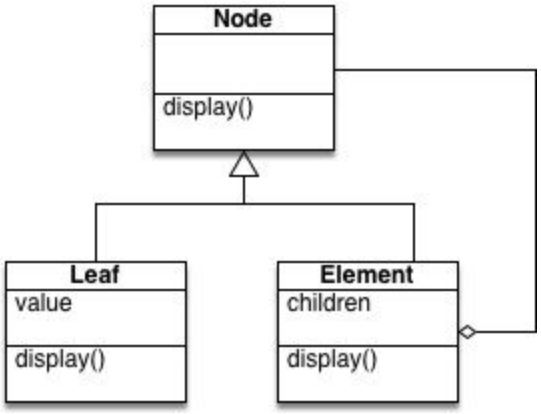
```
<A>
  <B> c1 </B>
  <B>
    <C> c2 </C>
  </B>
  <B>
    <D> c3 </D>
    <D> c4 </D>
  </B>
</A>
```



Utilize el patrón Composite para implementar una clase XmlTree con un método display que genere la versión en texto del árbol. Haga un diagrama de clases con todo lo necesario (métodos y atributos) y escriba (Ruby) el método display.
No se preocupe por identación o espacios, puede mostrar el ábol anterior como:

<A> <B> c1 </B> <B> <C> c2 </C> </B> <B> <D> c3 </D> <D> c4 </D> </B> </A>

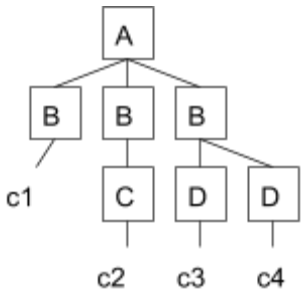
```
class Node
  def initialize()
  end
  def display()
  end
end

class Leaf < Node
  def initialize(value)
    @value = value
  end
  def display
    print " #{@value} "
  end
end
```




```
class Element < Node
  def initialize(label, children)     #children es un array de Node
    @label = label
    @children = children
  end
  def display
    print "< #{label} >"
    @children.each {|node| node.display}
    print "</ #{label} >"
  end
```

Podemos construir el árbol de abajo hacia arriba

```
n1 = Element.new('C', [Leaf.new('c2')])
n2 = Element.new('D', [Leaf.new('c3')])
n3 = Element.new('D', [Leaf.new('c4')])
n4 = Element.new('B', [Leaf.new('c1')])
n5 = Element.new('B', [n1])
n6 = Element.new('B', [n2, n3])
n7 = Element.new('A', [n4, n5, n6])
```



n7.display produce el documento xml asociado

< A > < B > c1 </B> <B> <C> c2 </C> </B> <B> <D> c3 </D> <D> c4 </D> </B> </A>

5. a) Un producto de software comprende 25 módulos. Históricamente un 5% de los módulos contiene algún error. ¿Cual es la probabilidad de que si se seleccionan 5 módulos al azar encontremos algún error en 2 de ellos? (Ayuda: use distribución binomial)

   Distribución binomial de 2 sobre 5 en que la probabilidad de esos 2 es de 0.05

   $$P(X = 2) = C_2^5 0.05^2 (0.95)^3 = 0.02$$

   b) Si lo que se sabe es que en promedio se están encontrando 0.2 defectos por módulo indique cual es la probabilidad de encontrar un defecto en un módulo dado y cuál es el máximo número de defectos que deberíamos esperar por módulo (Ayuda: use distribución de Poisson)

   Distribución de Poisson con lambda (media) de 0.2 y sigma de 0.45 (raíz de lambda)

   $$P(1) = \frac{0.2^1 \quad e^{-0.2}}{1!} = 0.24$$

   Máximo número de defectos = lambda + 3* sigma = 1.55