1. (20 pts) Dos de los patrones básicos muy utilizados que estudiamos en clases son Fábrica (Factory) y Fábrica Abstracta (Abstract Factory). Respecto a ellos conteste las siguientes preguntas en la forma mas corta y precisa posible:

   a) Por qué se necesita "Factory" cuando podríamos simplemente usar constructores

   b) Muestre un ejemplo concreto de "Factory" (diagrama de clases y breve explicación) distinto al que se vio en clases

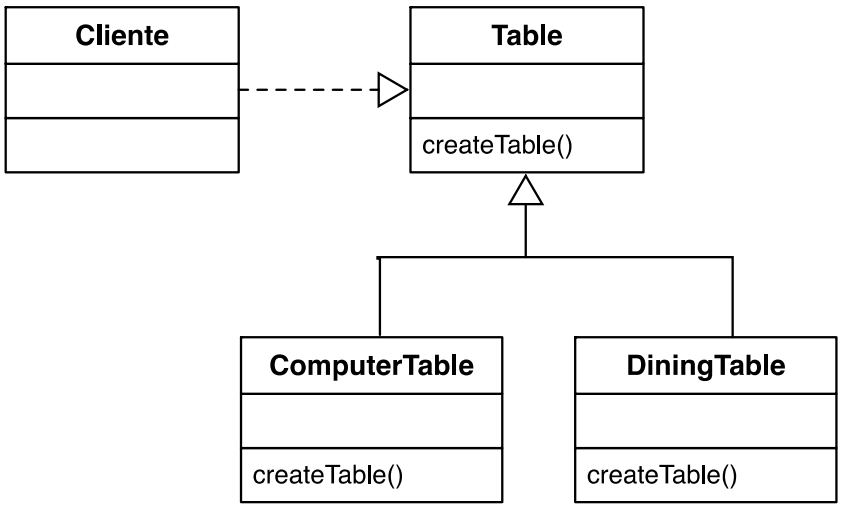
   c) Por qué se necesita "Abstract Factory" y cual es la diferencia con "Factory"

   d) Muestre un ejemplo concreto de "Abstract Factory" (diagrama de clases y breve explicación) distinto al que se vio en clases

---

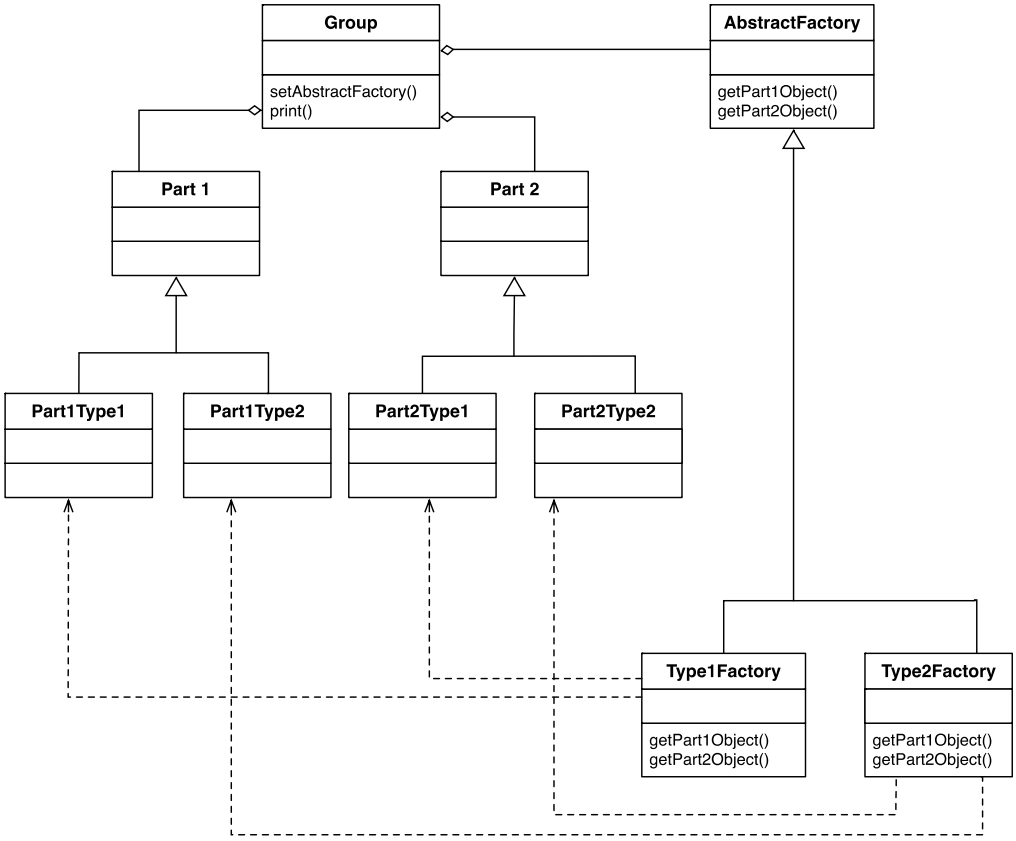
   a) Los constructores son inadecuados cuando se requiere crear objetos de subclases que son determinados en runtime.

   b) Supongamos que hay una clase Table con subclases ComputerTable y DiningTable. El método createTable produce una nueva Table de acuerdo a lo que se necesite.



c) El propósito de "Abstract Factory" es proveer una interfaz para crear familias de objetos relacionados en runtime sin especificar los objetos concretos. Esto se logra creando una clase para esta Abstract Factory que contiene una operación para cada clase en la familia

d) En la figura el cliente invoca al Grupo (objetos Part1 y Part2) para imprimir objetos Part1Type1. Para ello usa la clase AbstractFactory para invocar la operación virtual getPart1 que en realidad va a corresponder a getPart1 de Type1Factory.



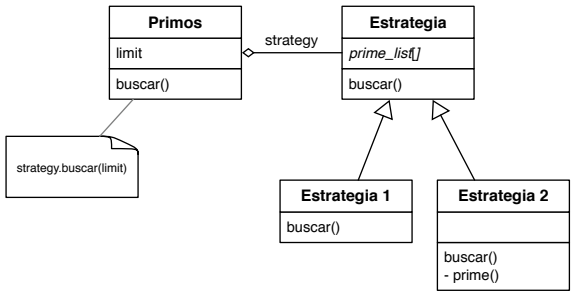
2. (30 pts) Se quiere escribir una pieza de código que sirva para calcular números primos sacando partido del conocido patrón de diseño Estrategia (Strategy). La idea es que podamos tener un algoritmo de cálculo separable del resto, de modo de poder calcular los númerus primos menores a 30 con a lo menos los siguientes dos algoritmos:
   - algoritmo hard_coded – los primeros números primos están guardados en forma estática en una lista
   - algoritmo standard – se verifica si el número es divisible por cada uno de los números comenzando con 2.

   La idea es poder hacer algo como lo siguiente:

```
primos = Primos.new(Estrategia1.new)
primos.buscar
primos = Primos.new(Estrategia2.new)
primos.buscar
```

   a) (10 pts) Describa usando UML la solución de este problema usando el patrón Strategy.
   b) (20 pts) Escriba el código completo de la implementación en Ruby para que funcione como en el ejemplo

a)



b)

```
class Primos
  attr_reader :limit
  attr_accessor :strategy

  def initialize (strategy)
    @limit = 30
    @strategy = strategy
  end

  def buscar
    @strategy.buscar(@limit)
  end
end
```

```
class Estrategia
  @@prime_list = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43,
47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107,109, 113]
end

class Estrategia1 < Estrategia

  def buscar (limit)
    i = 0
    while i < limit
      puts @@prime_list[i]
      i += 1
    end
  end

end

class Estrategia2 < Estrategia

  def buscar (limit)
    i = 0
    n = 2
    while i < limit
      if prime(n)
        puts n
        i += 1
      end
      n += 1
    end
  end

  private

  def prime (n)
    success = true
    from = 2
    to = (Math.sqrt n).floor
    divisor = from
    while divisor <= to
      if n/divisor*divisor == n
        success = false
        break
      end
      divisor += 1
    end
    return success
  end

end
```

3. (30 pts) Esta pregunta tiene que ver con conceptos de arquitectura

a) Indique 4 atributos de calidad que tienen que dependen de la arquitectura del software

performance
escalabilidad
disponibilidad
seguridad
continuidad operacional

b) Explique la diferencia entre "tier" y "capa" y dé ejemplos de cada uno

Las capas son niveles de abstracción cada vez mayores, una capa superior asume y depende de que exista la capa inmediatamente bajo ella (por ejemplo HTTP es una capa sobre TCP/IP) . Los tiers son unidades arquitectónicas separadas de acuerdo a algún criterio determinado (por ejemplo lógica de negocios, presentación y almacenamiento) que no necesariamente es el de nivel de abstracción.

c) Cual es la innovación importante que introdujo la arquitectura SOA y cuáles son las principales ventajas respecto a lo que se tenía antes de su aparición

Hay dos innovaciones importantes: acoplamiento débil entre las componentes a través de uso de mensajes y el uso de estándares de la Web (HTTP, XML)

d) Explique las principales diferencias entre la arquitectura SOA y la arquitectura de Microservicios

Menor granularidad, no hay necesidad de una capa de integración (middleware), bases de datos o almacenamientos propios

e) ¿Cual es el rol del API layer o API Gateway en la arquitectura de microservicios? ¿Es absolutamente necesaria? Justifique

La API layer hace el papel de "fachada" simplificando y exponiendo a los clientes una versión simplificada o reducida a la suma de operaciones que expone cada servicio por si solo. Puede manejar cambios de protocolos de modo de uniformizar el acceso de un cliente a todos los servicios.
No es absolutamente necesaria y puede hacerse que los clientes se comuniquen directamente con las apis.

f) A la derecha aparece un diagrama que ilustra la arquitectura de Microsoft .Net ¿Qué tipo de arquitectura representa este diagrama? Justifique su respuesta

Clásica arquitectura de capas. Puede verse como la capa del sistema operativo es la mas baja y sobre ella la del CLI. Mas arriba ASP.NET y Windows Forms , etc.

4. (20 pts) Esta pregunta tiene que ver con conceptos de DevOps

a) ¿Qué relación hay entre los métodos ágiles y el movimiento "DevOps"?

La necesidad de DevOps surge de ver que la agilización de solamente el desarrollo producía un cuello de botella en las etapas siguientes (producción) hasta que el producto pueda quedar disponible. En el fondo se trata de expandir la agilidad de modo que incorpore todo el ciclo desde que surge la necesidad hasta que se pone en las manos de quienes lo van a utilizar.

b) ¿Por qué se dice que existe una tensión entre los desarrolladores (Devs) y las personas encargadas de poner en producción y operar el producto?

Porque los desarrolladores presionan para que las aplicaciones que ellos construyen queden disponibles lo mas rápidamente posible mientras que la gente de producción tiene la responsabilidad de tener un sistema estable en todo momento.

c) ¿Cuales son las dos prácticas esenciales que debe adoptar una organización para implementar DevOps? Explique en qué consisten y por qué son esenciales

Integración Contínua: en lugar de tener puntos poco frecuentes de mucha integración, los nuevos desarrollos se van incorporando casi de inmediato (todos los días)

Entrega Contínua: no solo se integran los nuevos desarrollos día a día sino que se va dejando en las manos de los usuarios muy frecuentemente

d) ¿Por qué se dice que adoptar DevOps es más que una integración de Dev y Ops?

Porque se trata de inculcar en toda la gente la perspectiva del negocio. El que cada persona se sienta responsable de que al negocio le vaya bien. Esto hace que todos participen en el mejoramiento del proceso y elimina el nefasto "eso a mi no me incumbe"