**ESCUELA UNIVERSITARIA DE INGENIERÍA**

**ASIGNATURA : Ingeniería de Software 2**

**EJERCICIOS**

1. **Indique qué patrón se vislumbra en el siguiente código:**

|  |
| --- |
| class EstadoPedido:      def \_\_init\_\_(self, pedido):          self.\_pedido = pedido      def comprar(self):          pass      def facturar(self):          pass      def imprimir(self):          print("Estado Base")  class EstadoInicial(EstadoPedido):      def comprar(self):          self.\_pedido.\_estado = EstadoComprado(self.\_pedido)      def imprimir(self):          print("Estado Inicial")  class EstadoComprado(EstadoPedido):      def facturar(self):          self.\_pedido.\_estado = EstadoFacturado(self.\_pedido)      def imprimir(self):          print("Estado Comprado") |

1. **Indique qué patrón se vislumbra en el siguiente código:**

|  |
| --- |
| class ConectorJSON:      def \_\_init\_\_(self, filepath):          self.data = dict()          with open(filepath, mode='r', encoding='utf-8') as f:              self.data = json.load(f)      def parse\_data(self):          return self.data  class ConectorXML:      def \_\_init\_\_(self, filepath):          self.tree = etree.parse(filepath)      def parse\_data(self):          return self.tree |

1. **Indique qué patrón se vislumbra en el siguiente código:**

|  |
| --- |
| class MapaManager:  instance = None  @classmethod  def get\_instance(cls):  if cls.instance == None:  cls.instance = MapaManager()  return cls.instance |

1. **Indique qué patrón se vislumbra en el siguiente código:**

|  |
| --- |
| import copy  class Desaparecido:  def \_\_clone\_\_(self):  return copy.deepcopy(self) |

1. **Indique qué patrón se aplica en el siguiente código:**

|  |
| --- |
| class ConectorManager:  def obtener\_conector(self, filepath):  if filepath.endswith('json'):  connector = ConectorJSON  elif filepath.endswith('xml'):  connector = ConectorXML  else:  raise ValueError('No se puede conectar {}'.format(filepath))  return connector(filepath) |

1. **Indique el patrón o patrones que se aplican en el siguiente código:**

|  |
| --- |
| import copy  class LugarDeInteres:  instance = None  latitud = None  longitud = None  def \_\_clone\_\_(self):  return copy.deepcopy(self) |

1. **Indique el patrón o patrones que se aplican en el siguiente código:**

|  |
| --- |
| class Conector:  conector = None  @classmethod  def get\_conector(self):  if self.conector == None:  self.conector = Conector()  return self.conector |

1. **Indique el patrón o patrones que se aplican en el siguiente código:**

|  |
| --- |
| class PedidosManager:      def realizar\_pedido(self, pedido):          cliente = Cliente()          mozo = Mozo()          cliente.realizar\_pedido(mozo, pedido)  class Cliente:      def realizar\_pedido(self, mozo, pedido):          mozo.escribir\_pedido(pedido)          print("El cliente realizó el pedido: %s\n" % pedido)  class Mozo:      def escribir\_pedido(self, pedido):          self.pedido = pedido          print("El mozo escribió el pedido %s en sus notas\n" % pedido)  def main():      manager = PedidosManager()      manager.realizar\_pedido("Plato de fondo") |

1. **Indique el patrón o patrones que se aplican en el siguiente código:**

|  |
| --- |
| class BaseDatosMySQL:  def \_\_init\_\_(self, connection):  pass  def connect(self):  pass  class BaseDatosPostgreSQL:  def \_\_init\_\_(self, connection):  pass  def connect(self):  pass  class BaseDatosGestor:  def obtener\_base\_datos(self, connection):  if "MySQL" in connection:  connector = BaseDatosMySQL  elif "PosgreSQL" in connection:  connector = BaseDatosPostgreSQL  return connector(filepath) |

1. **Indique el patrón o patrones que se aplican en el siguiente código:**

|  |
| --- |
| import json  def convertir\_respuesta(func):  def func\_wrapper(datos, status):  cadena = "{header : {status: %s}, body: %s}" % (status, func(datos))  return cadena  return func\_wrapper  @convertir\_respuesta  def envolver\_json(diccionario):  cad = json.dumps(diccionario)  return cad |

1. **Indique el principio SOLID que se aplica en el siguiente código corregido:**

|  |
| --- |
| Código inicial |
| class Mozo():      def \_\_init\_\_(self, sueldo):          self.sueldo = sueldo      def recibir\_sueldo(self):          print("El mozo recibió: %d" % self.sueldo)  class Empleado(Mozo):      pass  def main():      empleado\_1 = Empleado(3000)      empleado\_1.recibir\_sueldo() |
| Código corregido |
| class Empleado:      def \_\_init\_\_(self, sueldo):          self.sueldo = sueldo  class Mozo(Empleado):      def recibir\_sueldo(self):          print("El mozo recibió: %d" % self.sueldo)  def main():      empleado\_1 = Mozo(3000)      empleado\_1.recibir\_sueldo() |

1. **Indique el principio o los principios SOLID que se aplican en el siguiente código corregido:**

|  |
| --- |
| Código inicial |
| class Restaurant:      def realizar\_pedido(self, pedido):          print("Se realizó el pedido: %s\n" % pedido)      def escribir\_pedido(self, pedido):          self.pedido = pedido          print("Se escribió el pedido %s en las notas\n" % pedido)  def main():      restaurant = Restaurant()      restaurant.escribir\_pedido("Arroz con pollo")      restaurant.realizar\_pedido("Arroz con pollo") |
| Código corregido |
| class Cliente:      def realizar\_pedido(self, mozo, pedido):          mozo.escribir\_pedido(pedido)          print("El cliente realizó el pedido: %s\n" % pedido)  class Mozo:      def escribir\_pedido(self, pedido):          self.pedido = pedido          print("El mozo escribió el pedido %s en sus notas\n" % pedido)  def main():      cliente = Cliente()      mozo = Mozo()      cliente.realizar\_pedido(mozo, "Arroz con pollo") |

1. **Indique el principio o los principios SOLID que se aplican en el siguiente código corregido:**

|  |
| --- |
| Código inicial |
| class Pedido:      def \_\_init\_\_(self, pedido):          self.pedido = pedido      def servir\_pedido(self):          raise NotImplementedError()      def servir\_bebida(self, bebida):          raise NotImplementedError()        def servir\_carne(self, termino):          raise NotImplementedError()  class PedidoCombo(Pedido):      def servir\_pedido(self):          print("Servir pedido %s" % self.pedido)      def servir\_bebida(self, bebida):          print("Servir con bebida %s" % bebida)        def servir\_carne(self, termino):          pass #ignorar |
| Código corregido |
| class Pedido:      def \_\_init\_\_(self, pedido):          self.pedido = pedido      def servir\_pedido(self):          print("Servir pedido %s" % self.pedido)  class PedidoConBebida():      def servir\_bebida(self, bebida):          print("Servir con bebida %s" % bebida)    class PedidoConCarne():      def servir\_carne(self, termino):          print("Servir carne a termino %s" % (termino))  class PedidoCombo(Pedido, PedidoConBebida):      def servir(self, bebida):          self.servir\_pedido()          self.servir\_bebida(bebida)  class PedidoParrillaSola(Pedido, PedidoConCarne):      def servir(self, termino):          self.servir\_pedido()          self.servir\_carne(termino) |

1. **Indique el principio SOLID que se aplica en el siguiente código corregido:**

|  |
| --- |
| Código inicial |
| class Cliente:      def establer\_nombres\_completos(self, nombres\_completos):          self.nombres\_completos = nombres\_completos        def establer\_dni(self, dni):          self.dni = dni        def mostrar\_datos(self):          print("Datos del cliente: %s %s" % (self.nombres\_completos, self.dni)) |
| Código corregido |
| class Cliente:      def mostrar\_datos(self):          raise NotImplementedError()  class ClienteEmpresa(Cliente):      def establer\_razon\_social(self, razon\_social):          self.razon\_social = razon\_social        def establer\_ruc(self, ruc):          self.ruc = ruc      def mostrar\_datos(self):          print("Datos del cliente: %s %s" % (self.razon\_social, self.ruc))  class ClientePersona(Cliente):      def establer\_nombres\_completos(self, nombres\_completos):          self.nombres\_completos = nombres\_completos        def establer\_dni(self, dni):          self.dni = dni      def mostrar\_datos(self):          print("Datos del cliente: %s %s" % (self.nombres\_completos, self.dni)) |

1. **Indique el principio SOLID que se aplica en el siguiente código corregido:**

|  |
| --- |
| Código inicial |
| class Mozo():      def definir\_sueldo(self, sueldo, propinas):          self.sueldo = sueldo + propinas      def recibir\_sueldo(self):          print("El mozo recibió: %d" % self.sueldo)  class Cajero():      def definir\_sueldo(self, sueldo):          self.sueldo = sueldo      def recibir\_sueldo(self):          print("El cajero recibió: %d" % self.sueldo)  class Llamador():      def definir\_sueldo(self, sueldo, comision):          self.sueldo = sueldo + comision      def recibir\_sueldo(self):          print("El llamador recibió: %d" % self.sueldo)  def main():      empleado\_1 = Mozo()      empleado\_2 = Cajero()      empleado\_3 = Llamador()        empleado\_1.definir\_sueldo(3000, 200)      empleado\_2.definir\_sueldo(3500)      empleado\_3.definir\_sueldo(1000, 50)      empleado\_1.recibir\_sueldo()      empleado\_2.recibir\_sueldo()      empleado\_3.recibir\_sueldo() |

|  |
| --- |
| Código corregido |
| class Empleado:      def definir\_sueldo(self, sueldo\_base, ingreso\_extra):          raise NotImplementedError()        def recibir\_sueldo(self):          raise NotImplementedError()  class Mozo(Empleado):      def definir\_sueldo(self, sueldo\_base, ingreso\_extra):          self.sueldo = sueldo\_base + ingreso\_extra      def recibir\_sueldo(self):          print("El mozo recibió: %d" % self.sueldo)  class Cajero(Empleado):      def definir\_sueldo(self, sueldo\_base, ingreso\_extra):          self.sueldo = sueldo\_base + ingreso\_extra      def recibir\_sueldo(self):          print("El cajero recibió: %d" % self.sueldo)  class Llamador(Empleado):      def recibir\_sueldo(self):          print("El llamador recibió: %d" % self.sueldo)  def main():      empleado\_1 = Mozo()      empleado\_2 = Cajero()        empleado\_1.definir\_sueldo(3000, 200)      empleado\_2.definir\_sueldo(3500, 0)      empleado\_1.recibir\_sueldo()      empleado\_2.recibir\_sueldo()      #ejemplo: no cumple con implementar métodos      # muestra error no implementado      empleado\_3 = Llamador()      empleado\_3.definir\_sueldo(1000, 50)      empleado\_3.recibir\_sueldo() |

**EJERCICIOS - RESPUESTAS**

1. **State**
2. **Adapter**
3. **Singleton**
4. **Prototype**
5. **Factory**
6. **Prototype**
7. **Singleton**
8. **Fachada**
9. **Factory, Adapter**
10. **Decorator**
11. **Dependency Inversion**
12. **Single Responsability, Open Close**
13. **Interface Segregation**
14. **Liskov Substitution**
15. **Open Close, Single Responsability**