Imagen que contiene Texto

Descripción generada automáticamente

*Análisis y diseño de sistemas de cómputo*

*Práctica semana 8*

*Trimestre 21-O*

*Prof. Orlando Muñoz Texzocotetla*

**Realizado por Equipo X**

**Integrado por:**

* **Ramírez Ramos Diana**

* **Anaya Castillo Brayan Angel**

* **Frances Nava Jesús**

* **Torres Néquiz Juan Manuel**

1. Una compañía de mantenimiento a un grupo de autos de carreras. Estos autos utilizan algunos de los nuevos motores de 8 cilindros y las nuevas transmisiones. Una vez que los motores son ensamblados, los pistones, el carburador y los tapones no pueden cambiarse con otros motores debido a los  cambios que causan a altas temperaturas. Se desea mantener el registro del  rendimiento alcanzado por cada motor en cada auto, y de cada transmisión en  combinación con cada motor. Los conductores dan su evaluación después de  manejar su correspondiente auto. Para ello se necesita de un sistema que lleve el  seguimiento a las configuraciones de cada auto (y de cada motor), así como de las  evaluaciones dadas por los conductores. Crear un diagrama de clases UML que  representa la información dada en el párrafo anterior. El diagrama debe tener al  menos 5 clases, con sus correspondientes atributos y métodos, así como sus correspondientes relaciones entre ellas: asociación, todo-parte y herencia.

Captura de pantalla de un periódico

Descripción generada automáticamente con confianza media

2. Indique si el siguiente código en Java representa una relación de agregación o de  composición, y explique por qué.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

El código representa una relación de agregación a la clase A de la clase B

por lo tanto, la clase B sería parte de A, la cual esta puede ser ocupada por los métodos de la clase A, y si tuviéramos una clase de prueba podríamos imprimir a b, que es el objeto de B, si este se elimina la clase A podría seguir imprimiendo objetos de otra clase, y si A se elimina podemos llamar a b desde la clase B.

3. Basados en el siguiente diagrama de clases, indique cuáles son las relaciones que  se indican con un signo de interrogación, y escriba un programa en Java que  implemente dichas relaciones.

Diagrama

Descripción generada automáticamente

Las relaciones que se indican con un signo de interrogación son del tipo de agregación.

Diagrama

Descripción generada automáticamente

Codigo en Java:

|  |
| --- |
| public class Cabecera {     private String titulo;       public String getTitulo() {  return titulo;  }   public void setTitulo(String titulo) {  this.titulo = titulo;  }  }  public class Cuerpo {     private String cuerpo;   public String getCuerpo() {  return cuerpo;  }   public void setCuerpo(String cuerpo) {  this.cuerpo = cuerpo;  }     }  public class ArchAdjunto {          private String nombreArchivo;       public String getNombreArchivo() {  return nombreArchivo;     }     public void setNombreArchivo(String nombreArchivo) {  this.nombreArchivo = nombreArchivo;     }          public ArchAdjunto(String nombreArchivo) {  this.nombreArchivo = nombreArchivo;     }  }  public class MensajeCorreo {          public Cabecera cabecera;     public Cuerpo cuerpo;     public ArchAdjunto []archadj;        public MensajeCorreo() {}       public MensajeCorreo(Cabecera cabecera,Cuerpo cuerpo,ArchAdjunto []archadj){         this.cabecera = cabecera;  this.cuerpo   = cuerpo;         this.archadj  = archadj;     }       @Override     public String toString() {  return "Mensaje Correo: \n" +  "[ Cabecera: " + cabecera.getTitulo() +   "\n Cuerpo: " + cuerpo.getCuerpo() +                  "\n Archivo Adjunto: " + Arrays.toString(archadj) +" ]";          } }  public class PracSem8\_Ejer3 {      /\*\*      \* @param args the command line arguments      \*/      public static void main(String[] args) {         // TODO code application logic here         Cabecera head = new Cabecera();  Cuerpo body = new Cuerpo();         ArchAdjunto []ad = new ArchAdjunto[3];          head.setTitulo("Titulo 1");  body.setCuerpo("Aquí va el cuerpo del correo 1");  ad[0] = new ArchAdjunto("Archivo no 1");  ad[1] = new ArchAdjunto("Archivo no 2");  ad[2] = new ArchAdjunto("Archivo no 3");           MensajeCorreo mensaje = new MensajeCorreo(head, body, ad);  System.out.println(mensaje);       }   } |

4. El código de las siguientes clases contiene errores, indique cuáles son e indique  cómo corregirlos.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Solución al problema del llamado a los métodos, que no son heredadas por las clases A, ya que en la parte de la impresión, se hace un llamado al objeto oA.getB(); En la cual getB(); es un metodo que esta contenido en la clase de B, y este hereda únicamente de A, lo que implica que la clase B, puede llamar los métodos de la clase de A, pero la clase A no hereda de la clase de B,  para poder llamar al método getB(); se tiene que realizar una instancia de la clase de B y este se podrá imprimir.

La segunda solución de la parte 2 donde se realiza una instancia de A pero del tipo B, esta forma no se puede, se tiene que realizar una instancia de la clase B del tipo B, y también otra forma de hacerlo una instancia del tipo B, se puede realizar un método del tipo estático.

Ahora en conclusión en la parte dos se tiene que realizar una instancia de B del tipo B.

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

5. Se desea crear una clase llamada vector3D, y se decide que esta clase derive de otra clase llamada vector2D. La explicación de esa decisión es que un vector 3D  es un vector 2D pero con una dimensión más (añadida). Expliqué por qué esa  decisión (y explicación) es un error conceptual de herencia en programación  orientada a objetos.

Mediante la Herencia es posible crear una clase a partir de otra clase padre, la cual heredará tanto sus propiedades o atributos como los métodos de la misma.

Considerando al vector2D como el padre y al vector3D como el hijo, el vector 3d heredaría tanto los atributos como los métodos del vector 2d, el problema es que los métodos del vector 2d pueden no ser útiles para el vector 3d.

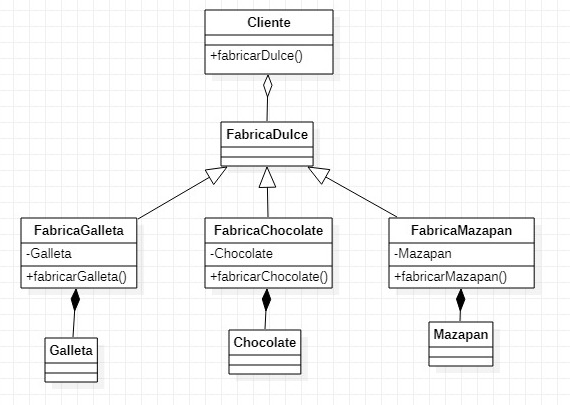
6. El siguiente diagrama de clases contiene tres clases que fabrican un tipo particular  de dulce: galleta, chocolate y mazapán. Se desea implementar UN SOLO método  llamado fabricarDulce(¿Tipo? x). Este método recibirá un objeto x (ya sea una  fábrica de galleta, de chocolate o de mazapán), el cual, creará el dulce  correspondiente.

a. Complete el diagrama UML para que por medio de la herencia y la  asociación sea posible que el método fabricarDulce reciba un solo objeto  x, el cual, pueda fabricar cualquiera de los tres tipos de dulce dependiendo  del objeto x recibido.

b. Implemente el método fabricarDulce(…).

**Diagrama

Descripción generada automáticamente**



Código:

public void fabricarDulce(FabricaDulce dulce) {  
          
 if(dulce.getClass().getSimpleName().compareTo("FabricaChocolate") == 0) {  
            Chocolate c = new FabricaChocolate();  
                c.fabricarChocolate();  
 }else {  
              if(dulce.getClass().getSimpleName().compareTo("FabricaMazapan") == 0){  
                 Mazapan m = new FabricaMazapan();  
     m.fabricarMazapan();

}else {  
       if(dulce.getClass().getSimpleName().compareTo("FabricaGalletas") == 0){  
           Galleta g = new FabricaGalletas();  
                   g.fabricarGalleta();  
 }

   }

}

}