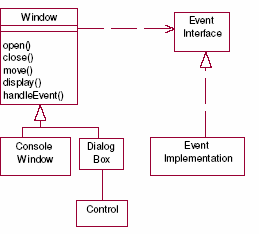
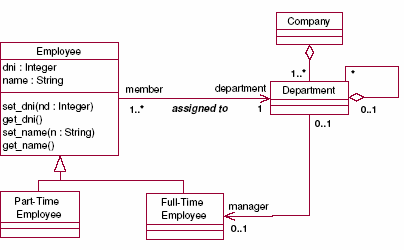
Ejercicios Diagramas de clases

# Ejercicio 1.

Considerando los siguientes diagramas de clases:



1. Indicar cada uno de los elementos de notación que están presentes en dichos diagramas.

--------------------------------------------------------------------------------------------------------------------

***Employee***

--------------------------------------------------------------------------------------------------------------------

* La clase **Employee** tiene una relación de asociación unidireccional con la clase Departamento. La cardinalidad de esta multiplicidad es: - un empleado trabaja en un solo departamento y en el departamento trabajan de 1 a N -. Esto quiere decir cuando creas una instancia de departamento tiene que tener obligatoriamente uno empleado al menos.
* La clase **Employee** tiene dos clases hijas**: Part-Time** y **Full-Time**.

-------------------------------------------------------------------------------------------------------------------

***Departamento***

-------------------------------------------------------------------------------------------------------------------

Dudas: la relación involutiva que quiere decir: lo dirige un empleado o el ***departamento tiene subdepartamentos.***

El departamento con Company: tiene una relación de agregación: La cardinalidad de esta multiplicidad es: - el departamento pertenece a una única empresa y la empresa tiene de 1 a N departamentos -

1. Describir brevemente qué interpretación proporcionan dichos diagramas.

* Una compañía que tiene muchos departamentos. mínimo un departamento. Donde en ese departamento tiene una lista de empleados que trabajan en un solo departamento. Existe empleados a jornada parcial y a jornada completa. Hay departamento que tienen subdepartamentos. Según este diagrama el manager siempre trabajara a jornada completa y puede ser que ese departamento tenga un jefe directo o no.

Ejercicios DC

# Ejercicio 2.

Teniendo en cuenta la tabla que se muestra a continuación, conectar los términos de la columna 1 con los términos de la columna 2 y completar la columna 3 con el tipo de flecha que corresponda. La primera columna enuncia los nombres de tipos de relaciones que se pueden dar entre clases. La segunda columna indica los términos utilizados cuando se leen dichas relaciones entre clases. Por último, la tercera columna es la que muestra las representaciones de dichas relaciones en los diagramas de clases.

|  |  |
| --- | --- |
| **Columna 2** | **Columna 3** |
|  | Tiene un rombo blanco, el rombo blanco se pone en la parte del todo |
|  | Es una línea discontinua con una flecha con la cabeza normal ( -🡪 ) apuntado a la entidad donde tiene la relación de asociacion |
|  | Es cuan una clase usa una segunda clase para crear una instancia de ella, pero no tiene porque estar necesariamente vinculada a la segunda clase. |
|  | La clase hija tiene una flecha de punta blanca(vacía), apuntando a su padre |
|  | Tiene un rombo negro, el rombo negro se pone la la parte del todo. |

|  |
| --- |
| **Columna 1** |
| Agregación |
| Asociación |
| Dependencia |
| Herencia |
| Composición |

# 1.Agregacion

# Relación agregación y composición en Java - Arkan

# 2. Asociacion

# Clases UML a Código Java

# 3.Dependencia

# Patrón de Inyección de dependencias - Adictos al trabajo

# 4.Herencia

# Herencia en Java: definición y ejemplos

# 5.Composicion

# Relación agregación y composición en Java - Arkan

# Ejercicio 3.

¿Qué tipo de relaciones pueden existir entre una clase y una interfaz?



**Clase**

**Interfaz**

1. **Implementación (implements):** Una clase puede implementar una o varias interfaces. Esto significa que la clase proporciona implementaciones concretas para todos los métodos declarados en esas interfaces. La palabra clave **implements** se utiliza para establecer esta relación.
2. **Herencia (extends) e Implementación**: Una clase puede extender otra clase y al mismo tiempo implementar una o varias interfaces. En este caso, la clase hereda los miembros de la clase base y proporciona implementaciones para los métodos de las interfaces. La palabra clave extends se utiliza para la herencia y implements para la implementación de interfaces.
3. **Herencia de interfaz (extends):** Una interfaz puede extender una o varias interfaces. En este caso, la interfaz que se extiende hereda los métodos declarados en las interfaces de las que extiende

-----------------------------------------------------------------------------------------------------------------------

Se pide:

1. Explicar cada una de estas relaciones mediante ejemplos ilustrativos en UML, incluyendo los atributos y métodos mínimos que sean necesarios.
2. Realizar la implementación en Java de los ejemplos anteriores.
3. ***Implementación de Interfaz***

public interface MiInterfaz {

void metodoInterfaz();

}

public class MiClase implements MiInterfaz {

*@Override*

public void metodoInterfaz() {

}

}

1. ***Herencia de Interfaz***

public interface InterfazA {

void metodoA();

}

public interface InterfazB extends InterfazA {

void metodoB();

}

public class MiClase implements InterfazB {

*@Override*

public void metodoA() {

// metodo de InterfazA

}

*@Override*

public void metodoB() {

// metodo de InterfazB

}

}

1. ***Múltiples Implementaciones de Interfaz***

public interface InterfazA {

void metodoA();

}

public interface InterfazB {

void metodoB();

}

public class MiClase implements InterfazA, InterfazB {

*@Override*

public void metodoA() {

// metodo de InterfazA

}

*@Override*

public void metodoB() {

// metodo de InterfazB

}

}

1. ***Relación de Dependencia:***

public interface MiInterfaz {

void metodoInterfaz();

}

public class OtraClase {

public void usarInterfaz(MiInterfaz interfaz) {

// Operaciones utilizando la interfaz

}

}

# Ejercicio 4.

Los requisitos para la gestión de nuestro videojuego son los siguientes:

* + En el videojuego aparecerán: personajes, enemigos, objetos muebles (o móviles) y objetos inmuebles (o inmóviles). Algunos ejemplos de objetos inmuebles son: muros, torres, árboles, etc. Ejemplos de objetos muebles son: alimentos, proyectiles, armas, etc.
  + Todos los elementos anteriores dispondrán de unas coordenadas “x” e “y” para ubicarlos.

Los personajes y los enemigos disponen de: nivel de energía, número de vidas y capacidad ofensiva (todos ellos son números enteros).

* + Los personajes (sólo los personajes) pueden recoger hasta tres objetos muebles en una mochila.
  + Las armas son objetos muebles y constan de un nivel de energía y de un factor potenciador de la capacidad ofensiva de los personajes.

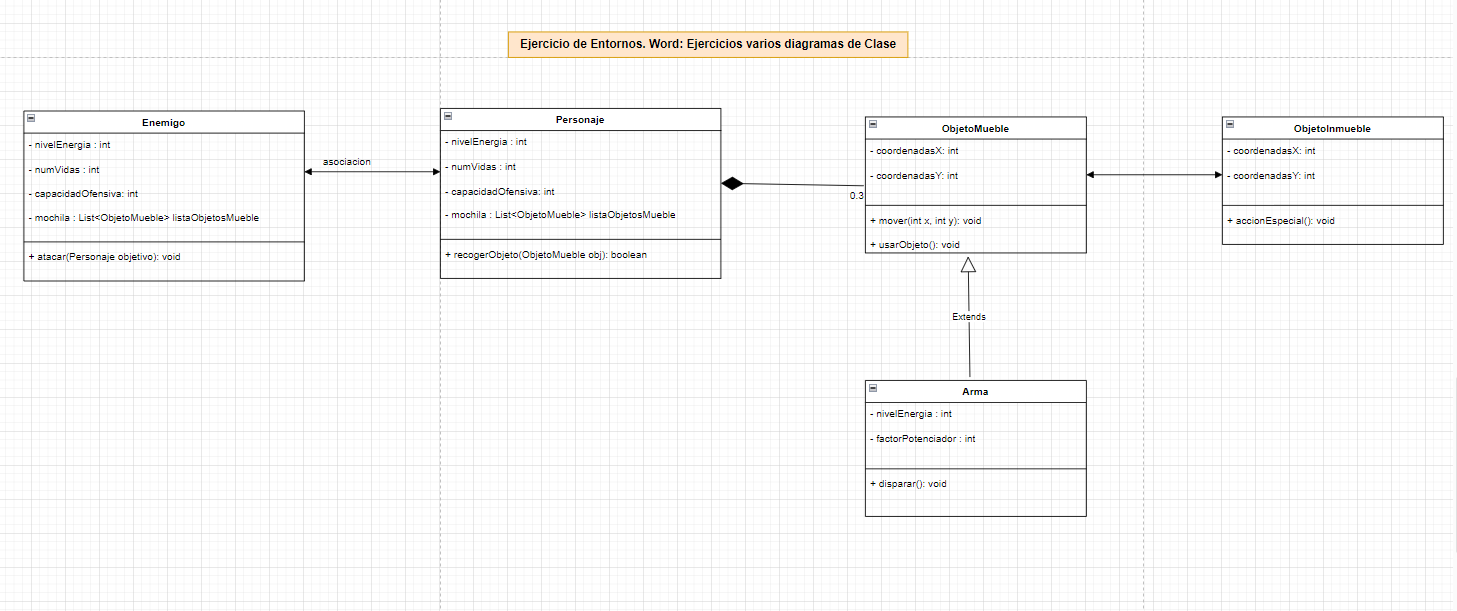
Se pide:

1. Identificar las clases y sus relaciones, y para cada clase indicar todos los atributos.

Ejercicios DC

**Relaciones entre Clases:**

1. **Personaje y ObjetoMueble (Composición):**
   * La clase **Personaje** tiene una relación de composición con la clase **ObjetoMueble** para representar la capacidad de llevar objetos muebles en la mochila. La relación es de composición, ya que la existencia de los objetos muebles en la mochila depende de la existencia del personaje. La multiplicidad indicada (máximo 3) sugiere que un personaje puede contener hasta tres objetos muebles.
2. **Arma y ObjetoMueble (Herencia):**
   * La clase **Arma** es una subclase de la clase **ObjetoMueble**, lo que indica una relación de herencia. Esto significa que **Arma** hereda los atributos de **ObjetoMueble** y también tiene atributos adicionales específicos de las armas.
3. **Personaje y Enemigo (Asociación):**
   * La relación entre **Personaje** y **Enemigo** es una asociación. Ambas clases están relacionadas entre sí, pero cada una puede existir de forma independiente de la otra.
4. **ObjetoMueble y ObjetoInmueble (Asociación):**
   * La relación entre **ObjetoMueble** y **ObjetoInmueble** es una asociación. Ambas clases están relacionadas entre sí, pero cada una puede existir de forma independiente de la otra.
5. Especificar el diagrama de clases.

***Ejercicio 5.***

Los requisitos para la gestión de nuestro sitio web son los siguientes:

* + Un sitio web consta de un nombre y un dominio. Por ejemplo, nuestro cole tiene como dominio [www.colegiojosefinos.es.](http://www.colegiojosefinos.es.)
  + Un sitio web contiene múltiples documentos definidos por su ruta. Por ejemplo, el acceso a servicios dentro del centro, estudios…
  + Los documentos pueden ser imágenes o páginas web.
  + Las imágenes tendrán dimensiones ‘x’ e ‘y’ y las páginas web podrán contener enlaces a otros documentos.

Se pide:

**a. Identificación de Clases y Atributos:**

1. **Clase SitioWeb:**
   * Atributos:
     + nombre: String
     + dominio: String
     + documentos: List<Documento>
2. **Clase Documento:**
   * Atributos:
     + ruta: String
     + tipo: Enum (imagen, paginaWeb)
3. **Clase Imagen (subclase de Documento):**
   * Atributos adicionales:
     + dimensionX: int
     + dimensionY: int
4. **Clase PaginaWeb (subclase de Documento):**
   * Atributos adicionales:
     + enlaces: List<Documento>

**b. Diagrama de Clases:**

