**Trabajo práctico Nro. 2**

|  |  |  |
| --- | --- | --- |
|  | **Asignatura: Programación II** |  |
|  |  |
| **Cursado:** Segundo semestre | **Horas** **semanales**: |
|  | **Horas semestrales:**  *Cantidad estimada de horas semestrales/anuales.* |
| **Carrera**: *Tecnicatura Universitaria en*  *Programación* | **Nivel (Año):**    3°  1  °  2  ° |
| **Ciclo Lectivo: 2024** |

**Integrantes de la Cátedra:**

- **DOCENTES:**

|  |  |  |
| --- | --- | --- |
| **Nombre del Profesor** | **Periodo** | **Cantidad horas materia** |
| **Cinthia Rigoni** |  | 6 horas |

1. ¿Qué es la herencia en java? ¿Cuáles son sus beneficios?

Es la relación de clases más importantes donde una clase brinda la estructura del comportamiento definida en una o más clases. La herencia implica una relación de especialización en la que una subclase especializa el comportamiento o la estructura más general de una superclases

1. ¿Cómo se representa la herencia en un diagrama UML?

La herencia se representa con una flecha que va desde la clase que hereda hasta la superclase y apunta en ella.

Ej Diagrama

Descripción generada automáticamente

1. ¿Cuándo es recomendable usar la herencia?

Es recomedable usar la herencia cuando creamos una clase general donde colocamos métodos y atributos que tienen todas las subclases en común y luego la heredamos a unas subclases mas especifica . Por ej: creamos una clase persona con atributos nombre,edad,dni y luego la heredamos a profesor y estudiante . Después dentro de profesor y estudiante especificamos atributos de cada uno

1. ¿Qué es una interfaz en java?

Una interfaz en java es un conjunto de métodos que deben ser implementados por cualquier clase que implemente esta interfaz donde es un acuerdo formal que garantiza que esa clase tendrá esos comportamientos . Al implementarlo no define el como se realizaran los atributos si no el que hay que hacer

1. ¿Cómo se representa una interfaz en un diagrama UML?

Se representa con una caja donde arriba tenes la palabra <<interface>> + el nombre de la interfaz y abajo sus métodos

Diagrama, Texto, Aplicación

Descripción generada automáticamente con confianza media

1. ¿Cuándo es recomendable usar una interfaz?

Es recomendable usar una interfaz cuando tenemos varias clases y necesitamos que todas tengan métodos en común entonces creamos una interfaz donde creamos esos métodos y luego en cada clase se adaptaran para las necesidades de ella .

Ej: creamos una interfaz Animal con métodos dormir, caminar, comer, reproducirse y se lo pasamos a clase Perro , Gato , Tigre y lo adaptamos a lo que realiza cada una

1. ¿Cuál es la diferencia entre una interfaz y una clase abstracta?

La diferencia es que en una clase abstracta podemos crear métodos , constructores y las subclases que la hereden se limitaran a heredar solo esa clase , en cambio las clases pueden tener varias interfaces , pero estas interfaces no podrán tener atributos y constructores solo métodos

1. ¿Qué tipos de relaciones pueden existir entre clases y objetos?

Pueden existir relaciones de :

* Asociación
  + Unidireccional
  + bidireccional
* Agregación
* Composición

1. ¿Cómo se representa cada relación en un diagrama UML?

Asociación unidimensional : Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Asociación bidimensional: Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Agregación: Imagen que contiene Diagrama

Descripción generada automáticamente

ComposicionImagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

1. ¿Cuándo es recomendable usar cada tipo de relación?

* Asociación:

Se utiliza cuando dos clases están relacionadas de alguna manera, pero no hay una dependencia fuerte entre ellas.

Puede ser bidireccional o unidireccional.

Las instancias de una clase pueden existir independientemente de la otra clase.

Ejemplo: Una clase Estudiante y una clase Curso están asociadas, pero los estudiantes pueden existir sin estar matriculados en un curso específico, y los cursos pueden existir incluso si no hay estudiantes matriculados.

* Composición:

Se utiliza cuando una clase contiene objetos de otras clases como parte de su estado interno.

Hay una relación de "parte-todo", donde las partes no pueden existir independientemente del todo.

La destrucción del objeto principal implica la destrucción de los objetos contenidos.

Ejemplo: Una clase Carro contiene objetos de las clases Motor, Rueda, etc. Los componentes del carro dependen de la existencia del carro en sí.

* Agregación:

Similar a la composición, pero con una relación más flexible.

Las partes pueden existir independientemente del todo.

La destrucción del objeto principal no implica necesariamente la destrucción de los objetos contenidos.

Puede haber múltiples objetos que hagan referencia al mismo objeto contenido.

Ejemplo: Una clase Equipo tiene una agregación de objetos de la clase Jugador. Los jugadores pueden estar en varios equipos y existir independientemente de cualquier equipo en particular.

**A CODIFICAR!**

1. Desarrollar un sistema para gestionar una tienda de mascotas que vende diferentes tipos de animales: perros, gatos, aves.

a. Clases:

1

# Programación II – Prof. Rigoni Cinthia

1. Animal: nombre (String), edad (int), tipoAnimal (String), precio (double). Método Alimentar(String nombre). Sobreescribir el método toString() para que devuelva una pequeña presentación del animal.
2. Perro: raza (String), vacunado (boolean). Hereda de Animal.
3. iii. Gato: raza (String), esterilizado (boolean). Hereda de Animal. iv. Ave: especie (String), habla(boolean). Hereda de Animal.

v. Crear también una clase Pez con los atributos que creas necesarios. Hereda de Animal.

Desarrollá las clases especificadas anteriormente y crea una nueva clase TiendaMascotas para gestionar todas las demás. Esta debe tener los métodos:

o addAnimal(Animal animal): Agrega un animal a la tienda. o venderAnimal(String nombre): Vende un animal por nombre. o alimentarAnimal(String nombre): Alimenta a un animal por nombre. o listarAnimales(): Lista todos los animales de la tienda.

En el main crea una tienda, un animal de cada clase y utilizá todos los métodos definidos.

2. Desarrollar un sistema de gestión para una biblioteca.

1. Definir una clase Libro con los atributos título (String), autor (String), añoDePublicacion (int), prestado (boolean). Podés crear más atributos si lo considerás necesario.
2. Crea una interfaz llamada Prestable con los métodos prestar() y devolver().
3. Libro debe implementar esta interfaz y se deben definir los métodos para cambiar el estado del libro.
4. Crear las clases Novela, Universitario, Infantil que hereden de Libro. Agregá a cada clase un atributo propio (o más, si te parece necesario).
5. Definir la clase Biblioteca, que contenga una lista de libros y métodos para agregar un libro, listar los libros disponibles, prestar un libro, devolver un libro.

En el main crea una biblioteca, un libro de cada clase y utilizá todos los métodos definidos.

2

# Programación II – Prof. Rigoni Cinthia