1. ¿Qué es la herencia en java? ¿Cuáles son sus beneficios?
La herencia en Java es un mecanismo que permite que una clase herede los atributos y métodos de otra clase. Los beneficios de la herencia incluyen la reutilización de código, la organización jerárquica de clases, la posibilidad de extender y modificar el comportamiento de las clases existentes, y la capacidad de crear clases más especializadas basadas en clases más generales.

2. ¿Cómo se representa la herencia en un diagrama UML?
En un diagrama UML, la herencia se representa con una flecha sólida que apunta desde la subclase hacia la superclase, con una línea sólida y una flecha triangular en el extremo de la flecha.

3. ¿Cuándo es recomendable usar la herencia?
Cuando una clase es una versión más específica de otra clase más general. También es útil cuando se desea reutilizar código común entre clases relacionadas.

4. ¿Qué es una interfaz en java?
Una interfaz en Java es una colección de métodos abstractos que define un contrato de comportamiento.

5. ¿Cómo se representa una interfaz en un diagrama UML?
En un diagrama UML, una interfaz se representa como una caja con el nombre de la interfaz en la parte superior, y los métodos definidos en la interfaz listados dentro de la caja.

6. ¿Cuándo es recomendable usar una interfaz?
Es recomendable usar una interfaz cuando se desea definir un contrato de comportamiento que varias clases pueden implementar de manera diferente.

7. ¿Cuál es la diferencia entre una interfaz y una clase abstracta?
La diferencia principal entre una interfaz y una clase abstracta es que una interfaz solo puede contener métodos abstractos y constantes, mientras que una clase abstracta puede contener métodos abstractos y métodos con implementación.

8. ¿Qué tipos de relaciones pueden existir entre clases y objetos?
Los tipos de relaciones que pueden existir entre clases y objetos incluyen la asociación, la composición, la agregación, la herencia y la dependencia.

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

- La asociación se representa con una línea sólida que conecta las clases involucradas.
- La composición se representa con una línea sólida con un rombo relleno en el extremo de la línea que apunta hacia la clase que contiene la otra clase.
- La agregación se representa de manera similar a la composición, pero con un rombo vacío en el extremo de la línea que apunta hacia la clase que contiene la otra clase.
- La herencia se representa con una flecha sólida que apunta desde la subclase hacia la superclase.
- La dependencia se representa con una línea punteada que conecta las clases, con una flecha sólida que apunta desde la clase dependiente hacia la clase de la que depende.

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

- La asociación se utiliza cuando dos clases están relacionadas de alguna manera, pero ninguna es parte de la otra.
- La composición se utiliza cuando una clase es parte de otra clase y no tiene sentido existir independientemente.
- La agregación se utiliza cuando una clase es parte de otra clase, pero puede existir independientemente.
- La herencia se utiliza cuando una clase es una versión más específica de otra clase más general.
- La dependencia se utiliza cuando una clase utiliza los servicios de otra clase, pero no forma parte de su estado interno.

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

a. Clases:

i. 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.

ii. Perro: raza (String), vacunado (boolean). Hereda de Animal.

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

```
import java.util.ArrayList;

import java.util.List;

public class Main {

    public static void main(String[] args) {
```

```
        // Crear una tienda
        TiendaMascotas tienda = new TiendaMascotas();
        // Agregar animales a la tienda
        tienda.addAnimal(new Perro("Rex", 4, "Canino", 200, "Labrador", true));
        tienda.addAnimal(new Gato("Whiskers", 3, "Felino", 150, "Siamese", false));
        tienda.addAnimal(new Ave("Coco", 2, "Ave", 50, "Parrot", true));
        tienda.addAnimal(new Pez("Goldie", 1, "Pez", 10, "Freshwater"));
        // Listar los animales en la tienda
        System.out.println("Animales en la tienda:");
        tienda.listarAnimales();
        System.out.println();
        // Vender un animal
        tienda.venderAnimal("Rex");
        System.out.println("Después de vender a Rex:");
        tienda.listarAnimales();
        System.out.println();
        // Alimentar a un animal
        tienda.alimentarAnimal("Whiskers");
        System.out.println();
        // Listar los animales restantes en la tienda
        System.out.println("Animales restantes en la tienda:");
        tienda.listarAnimales();
    }
}
public class TiendaMascotas {
    private List<Animal> animales;
    public TiendaMascotas() {
        this.animales = new ArrayList<>();
    }
    public void addAnimal(Animal animal) {
        animales.add(animal);
    }
```

```
    public void venderAnimal(String nombre) {
        for (Animal animal : animales) {
            if (animal.toString().contains(nombre)) {
                animales.remove(animal);
                System.out.println(nombre + " fue vendido.");
                return;
            }
        }
        System.out.println("No hay animal con el nombre " + nombre);
    }
    public void alimentarAnimal(String nombre) {
        for (Animal animal : animales) {
            if (animal.toString().contains(nombre)) {
                animal.alimentar();
                return;
            }
        }
        System.out.println("No se encontró ningún animal con el nombre " + nombre);
    }
    public void listarAnimales() {
        System.out.println("Animales en la tienda:");
        for (Animal animal : animales) {
            System.out.println(animal);
        }
    }
}
public class Pez extends Animal {
    private String especie;
    public Pez(String nombre, int edad, String tipoAnimal, double precio, String especie) {
        super(nombre, edad, tipoAnimal, precio);
        this.especie = especie;
    }
```

```
    public String getEspecie() {
        return especie;
    }
    public void setEspecie(String especie) {
        this.especie = especie;
    }
    @Override
    public void alimentar() {
    }
}
public class Perro extends Animal {
    private String raza;
    private boolean vacunado;
    public Perro(String nombre, int edad, String tipoAnimal, double precio, String raza, boolean
vacunado) {
        super(nombre, edad, tipoAnimal, precio);
        this.raza = raza;
        this.vacunado = vacunado;
    }
    public String getRaza() {
        return raza;
    }
    public void setRaza(String raza) {
        this.raza = raza;
    }
    public boolean isVacunado() {
        return vacunado;
    }
    public void setVacunado(boolean vacunado) {
        this.vacunado = vacunado;
    }
    @Override
```

```
    public void alimentar() {
    }
}
public class Gato extends Animal {
    private String raza;
    private boolean esterilizado;
    public Gato(String nombre, int edad, String tipoAnimal, double precio, String raza, boolean esterilizado) {
        super(nombre, edad, tipoAnimal, precio);
        this.raza = raza;
        this.esterilizado = esterilizado;
    }
    public String getRaza() {
        return raza;
    }
    public void setRaza(String raza) {
        this.raza = raza;
    }
    public boolean isEsterilizado() {
        return esterilizado;
    }
    public void setEsterilizado(boolean esterilizado) {
        this.esterilizado = esterilizado;
    }
    @Override
    public void alimentar() {
    }
}
public class Ave extends Animal {
    private String especie;
    private boolean habla;
```

```
    public Ave(String nombre, int edad, String tipoAnimal, double precio, String especie, boolean
habla) {
        super(nombre, edad, tipoAnimal, precio);
        this.especie = especie;
        this.habla = habla;
    }
    public String getEspecie() {
        return especie;
    }
    public void setEspecie(String especie) {
        this.especie = especie;
    }
    public boolean isHabla() {
        return habla;
    }
    public void setHabla(boolean habla) {
        this.habla = habla;
    }
    @Override
    public void alimentar() {
    }
}
public abstract class Animal {
    protected String nombre;
    protected int edad;
    protected String tipoAnimal;
    protected double precio;
    public Animal(String nombre, int edad, String tipoAnimal, double precio) {
        this.nombre = nombre;
        this.edad = edad;
        this.tipoAnimal = tipoAnimal;
        this.precio = precio;
```

```
    }
    public abstract void alimentar();
    @Override
    public String toString() {
        return "Animal" +
                "nombre='" + nombre + '\'' +
                ", edad=" + edad +
                ", tipoAnimal='" + tipoAnimal + '\'' +
                ", precio=" + precio;
    }
}
```

12. Desarrollar un sistema de gestión para una biblioteca.

a. Definir una clase Libro con los atributos título (String), autor (String), añoDePublicacion (int), prestado (boolean). Podés crear más atributos si lo considerás necesario.

b. Crea una interfaz llamada Prestable con los métodos prestar() y devolver().

c. Libro debe implementar esta interfaz y se deben definir los métodos para cambiar el estado del libro.

d. Crear las clases Novela, Universitario, Infantil que hereden de Libro. Agregá a cada clase un atributo propio (o más, si te parece necesario).

e. 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.

```
public class Main {
    public static void main(String[] args) {


        Biblioteca biblioteca = new Biblioteca();


        Libro libroNovela = new Novela("Cien años de soledad", "Gabriel García Márquez", 1967, false, "Romance");
        Libro libroUniversitario = new Universitario("Introducción a la programación", "John Doe", 2020, true, "Ingeniería");
        Libro libroInfantil = new Infantil("El principito", "Antoine de Saint-Exupéry", 1943, true, 5);


        biblioteca.agregarLibro(libroNovela);
```

```
        biblioteca.agregarLibro(libroUniversitario);
        biblioteca.agregarLibro(libroInfantil);

        biblioteca.listarLibrosDisponibles();

        biblioteca.prestarLibro("El principito");
        biblioteca.listarLibrosDisponibles();

        biblioteca.prestarLibro("El principito");

        biblioteca.devolverLibro("El principito");
        biblioteca.listarLibrosDisponibles();

        biblioteca.devolverLibro("Don Quijote");
    }
}
import java.util.ArrayList;
import java.util.List;

public class Biblioteca {
    private List<Libro> libros;

    public Biblioteca() {
        this.libros = new ArrayList<>();
    }

    public void agregarLibro(Libro libro) {
        libros.add(libro);
    }

    public void listarLibrosDisponibles() {
        System.out.println("Lista de libros disponibles:");
```

```
    for (Libro libro : libros) {
        if (!libro.prestado) {
            System.out.println(libro.titulo);
        }
    }
}

public void listaDeLibrosDisponibles() {
    for (Libro libro : libros) {
        if (!libro.prestado) {
            System.out.println(libro.titulo);
        }
    }
}

public void prestarLibro(String titulo) {
    for (Libro libro : libros) {
        if (libro.titulo.equals(titulo)) {
            System.out.println("El libro " + titulo + " ha sido prestado");
            libro.prestar();
            return;
        }
    }
    System.out.println("El libro " + titulo + " no está disponible para prestar en este momento");
}

public void devolverLibro(String titulo) {
    for (Libro libro : libros) {
        if (libro.titulo.equals(titulo)) {
            System.out.println("El libro " + titulo + " ha sido devuelto");
            libro.devolver();
            return;
```

```
            }
        }
        System.out.println("El libro " + titulo + " sigue en inventario, por lo tanto no es de esta biblioteca");
    }
}
public class Infantil extends Libro {
    private int edadRecomendada;

    public Infantil(String titulo, String autor, int añoDePublicacion, boolean prestado, int edadRecomendada) {
        super(titulo, autor, añoDePublicacion, prestado);
        this.edadRecomendada = edadRecomendada;
    }

    public int getEdadRecomendada() {
        return edadRecomendada;
    }

    public void setEdadRecomendada(int edadRecomendada) {
        this.edadRecomendada = edadRecomendada;
    }
}
public abstract class Libro implements Prestable {
    protected String titulo;
    protected String autor;
    protected int añoDePublicacion;
    protected boolean prestado;

    public Libro(String titulo, String autor, int añoDePublicacion, boolean prestado) {
        this.titulo = titulo;
        this.autor = autor;
```

```
        this.añoDePublicacion = añoDePublicacion;
        this.prestado = prestado;
    }

    @Override
    public void prestar() {
        if (!prestado) {
            System.out.println("El libro " + titulo + " ha sido prestado");
            prestado = true;
        } else {
            System.out.println("El libro " + titulo + " no está disponible para prestar en este momento");
        }
    }

    @Override
    public void devolver() {
        if (prestado) {
            System.out.println("El libro " + titulo + " ha sido devuelto");
            prestado = false;
        } else {
            System.out.println("El libro " + titulo + " sigue en inventario, por lo tanto no es de esta
biblioteca");
        }
    }
}
public class Novela extends Libro {
    private String genero;

    public Novela(String titulo, String autor, int añoDePublicacion, boolean prestado, String genero)
{
        super(titulo, autor, añoDePublicacion, prestado);
        this.genero = genero;
```

```
    }

    public String getGenero() {
        return genero;
    }

    public void setGenero(String genero) {
        this.genero = genero;
    }
}
public interface Prestable {
    public void prestar();

    public void devolver();
}
public class Universitario extends Libro {
    private String carrera;

    public Universitario(String titulo, String autor, int añoDePublicacion, boolean prestado, String
carrera) {
        super(titulo, autor, añoDePublicacion, prestado);
        this.carrera = carrera;
    }

    public String getCarrera() {
        return carrera;
    }

    public void setCarrera(String carrera) {
        this.carrera = carrera;
    }
}
```