Informe de Teoria 03

Tema: Herencia - Ejercicio 03

Nota

Estudiante	Escuela	Asignatura
Roni Companocca Checco rcompanocca@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Semestre: II Código: 20210558

Teoria	Tema	Duración
03	Herencia - Ejercicio 03	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - B	Del 22 Diciembre 2023	Al 23 Diciembre 2023

1. EQUIPOS, MATERIALES Y TEMAS UTILIZADOS

- Sistema Operativo Windows
- OpenJDK 64-Bits 17.0.7.
- Git 2.39.2.
- Cuenta en GitHub con el correo institucional.

2. URL DE REPOSITORIO GITHUB

- URL para el Repositorio GitHub.
- `https://github.com/RONI-COMPANOCCA-CHECCO`
- URL para el laboratorio 22 en el Repositorio GitHub.
- `https://github.com/RONI-COMPANOCCA-CHECCO/FP2-TRABAJO03/tree/master/trabajo03/Ejercicio3.java`

3. Ejercicio 03:

- Caso de estudio especial: herencia múltiple. Es un tipo de herencia en la que una clase hereda el estado (estructura) y el comportamiento de más de una clase base. (hay herencia múltiple cuando una clase hereda de más de una clase). Java no permite la herencia múltiple, pero se puede conseguir la implementación de la herencia múltiple usando interfaces. Implemente el siguiente diagrama de clases UML y consiga pruebas válidas.
- Codigo
- Para la herencia múltiple en Java utilizando interfaces, Vamos a crear tres interfaces: Vmarino, Vaereo y Vehiculo. Luego, implementaremos las clases Barco, Avion y Hidroavion para demostrar la herencia múltiple:
- Interfaz Vmarino.java

```java
// este es una interfaz
public interface Vmarino {
    // creamos dos metodos vacios en esta interfaz
    void navegar();
    void detenerse();
}
```

- Interfaz Vaereo.java

```java
// este es una interfaz
public interface Vaereo {
    // creamos dos metodos en esta intefaz
    void volar();
    void aterrizar();
}
```

- Interfaz Vehiculo.java

```java
// este es un interfaz con un solo metodo
public interface Vehiculo extends Vmarino, Vaereo {
    void aterrizarEnAgua();
}
```

- clase Barco.java

```java
// RONI COMPANOCCA CHECCO
// FP2 - TRABAJO 03
// CUI 20210558
public class Barco implements Vmarino {
    @Override
    public void navegar() {
        System.out.println("El barco est navegando.");
    }

    @Override
    public void detenerse() {
        System.out.println("El barco se ha detenido.");
    }
```

```
    public void cargarMercancia() {
        System.out.println("Cargando mercanca en el barco.");
    }
}
```

- clase Avion.java

```
// RONI COMPANOCCA CHECCO
// FP2 - TRABAJO 03
// CUI 20210558
// creamos esta clase con tres metodos
public class Avion implements Vaereo {
    @Override
    public void volar() {
        System.out.println("El avin est volando.");
    }

    @Override
    public void aterrizar() {
        System.out.println("El avin ha aterrizado.");
    }

    public void despegar() {
        System.out.println("El avin est despegando.");
    }
}
```

- clase Hidroavion.java

```
public class Hidroavion implements Vehiculo {
    @Override
    public void navegar() {
        System.out.println("El hidroavin est navegando.");
    }

    @Override
    public void volar() {
        System.out.println("El hidroavin est volando.");
    }

    @Override
    public void aterrizarEnAgua() {
        System.out.println("El hidroavin est aterrizando en agua.");
    }

    @Override
    public void detenerse() {
        System.out.println("El hidroavin se ha detenido.");
    }

    public void despegarDelAgua() {
        System.out.println("El hidroavin est despegando del agua.");
    }
```

```
    public void despegar() {
        System.out.println("El hidroavin est despegando.");
    }

    @Override
    public void aterrizar() {
        System.out.println("El hidroavin est aterrizando.");
    }
}
```

- clase Main.java

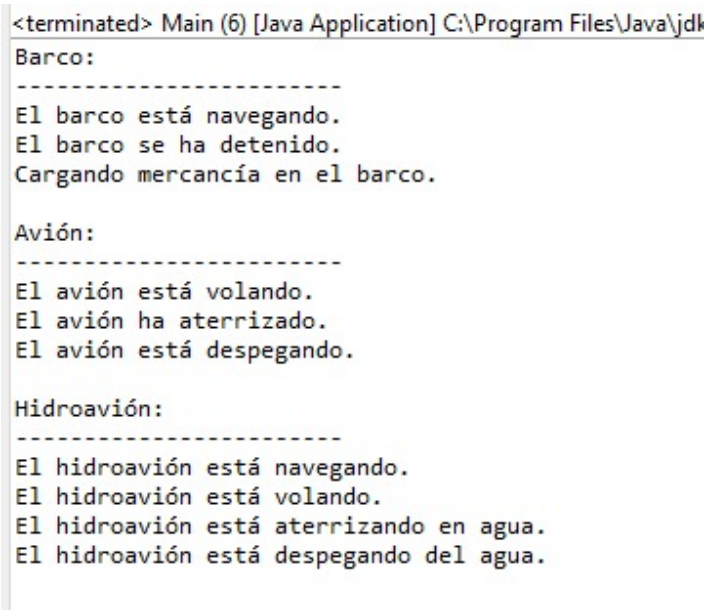
```
// RONI COMPANOCCA CHECCO
// FP2 - TRABAJO 03
// CUI 20210558
public class Main {
    public static void main(String[] args) {
        Barco barco = new Barco();
        Avion avion = new Avion();
        Hidroavion hidroavion = new Hidroavion();

        // Pruebas individuales
        System.out.println("Barco:");
        System.out.println("------------------------");
        barco.navegar();
        barco.detenerse();
        barco.cargarMercancia();

        System.out.println("\nAvin:");
        System.out.println("------------------------");
        avion.volar();
        avion.aterrizar();
        avion.despegar();

        System.out.println("\nHidroavin:");
        System.out.println("------------------------");
        hidroavion.navegar();
        hidroavion.volar();
        hidroavion.aterrizarEnAgua();
        hidroavion.despegarDelAgua();
    }
}
```

- Esta salida indica que cada tipo de vehículo implementa correctamente sus respectivas interfaces (Vmarino, Vaereo y Vehiculo), y también demuestra cómo puedes llamar a métodos específicos de cada clase.



- .

4. REFERENCIAS

- M. Aedo, "Fundamentos de Programación 2 - Tópicos de Programación Orientada a Objetos", Primera Edición, 2021, Editorial UNSA.
- `https://github.com/rescobedoq/programacion.git`
- J. Dean, Ïntroduction to programming with Java: A Problem Solving Approach", Third Edition, 2021, McGraw-Hill.
- C. T. Wu, .ªn Introduction to Object-Oriented Programming with Java", Fifth Edition, 2010, McGraw-Hill.
- P. Deitel, "Java How to Program", Eleventh Edition, 2017, Prentice Hall.