Informe de Teoría 03
Tema: Herencia en java

Nota

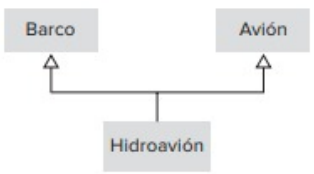
Estudiante	Escuela	Asignatura
Hernan Andy Choquehuanca Zapana hchoquehuancaz@unsa.edu.pe	Escuela Profesional de Ingenieria de Sistemas	Fundamentos de la Programacion II Semestre: II Código: 1701213

Trabajo	Tema	Duración
03	Herencia en java	02 horas

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

1. Tarea

- Ejercicio 01: Crear una clase base denominada Punto que conste de las coordenadas x e y. A partir de esta clase, definir una clase denominada Circulo que tenga las coordenadas del centro y un atributo denominado radio. Entre las funciones miembro de la primera clase, deberá existir una función distancia() que devuelva la distancia entre dos puntos, donde:
  Distancia = ((x2 – x1)2 + (y2 – y1)2)1/2 Entregable: fase02/trabajo03/Ejercicio1.java
- Ejercicio 02: Utilizando la clase construida en el ejercicio 01, obtener una clase derivada Cilindro derivada de Circulo. La clase Cilindro deberá tener una función miembro que calcule la superficie de dicho cilindro. La fórmula que calcula la superficie del cilindro es S = 2r(l + r) donde r es el radio del cilindro y l es la longitud. Entregable: fase02/trabajo03/Ejercicio2.java
- 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.



2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows 11 Pro 22H2 64 bits.
- Vim 9.0.
- Git 2.42.0.
- Cuenta en GitHub con el correo institucional.
- Editor LaTeX en línea Overleaf.
- Atributos de clase.
- Métodos de clase.
- Herencia.
- Herencia múltiple.
- Clases.
- Interfaces.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/hernanchoquehuanca/fp2-23b.git`
- URL para el trabajo 03 en el Repositorio GitHub.
- `https://github.com/hernanchoquehuanca/fp2-23b/tree/main/fase02/trabajo03`

4. Trabajo 03

4.1. Ejercicio 01

- La clase Punto contiene 2 atributos protegidos, los cuales son las coordenadas del punto creado.
- Contiene los setters y getters, además del método distancia el cual se halla con la siguiente fórmula:

$$\text{Distancia} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- Usando la clase Math se implementó el contenido del método distancia().

```
17 class Punto {
18   protected double x;
19   protected double y;
20
21   public void setX(double x) {
22     this.x = x;
23   }
24
25   public double getX(){
26     return x;
27   }
28
29   public void setY(double y) {
30     this.y = y;
31   }
32
33   public double getY(){
34     return y;
35   }
36
37   public double distancia(Punto p) {
38     return Math.sqrt(Math.pow((p.getX() - x), 2) + Math.pow((p.getY() - y), 2));
39   }
40 }
```

- La clase Circulo contiene 2 atributos protegidos que son su radio y el punto de su centro.
- Contiene los setters y getters de cada atributo.

```
42 class Circulo {
43   protected Punto centro;
44   protected double radio;
45
46   public void setCentro(Punto centro) {
47     this.centro = centro;
48   }
49
50   public Punto getCentro() {
51     return centro;
52   }
53
54   public void setRadio(double radio) {
```

```
    this.radio = radio;
  }

  public double getRadio() {
    return radio;
  }
}
```

- La clase Ejercicio1 contiene el método principal donde se realiza una prueba de las clases anteriormente creadas.
- Crea 2 puntos y usando las clases creadas hallamos sus distancias.

```
class Ejercicio1 {
  public static void main(String[] args){
    Punto p1 = new Punto();
    Punto p2 = new Punto();

    p1.setX(1.0);
    p1.setY(4.5);

    p2.setX(2.0);
    p2.setY(6.0);

    double distancia = p1.distancia(p2);
    System.out.println("La distancia entre el punto 1 y el punto 2 es: " + distancia);
  }
}
```

- PRUEBA DE EJECUCIÓN DEL CÓDIGO (Ejercicio1.java):

```
PS E:\fp2-23b\fase02\trabajo03> java Ejercicio1
La distancia entre el punto 1 y el punto 2 es: 1.8027756377319946

PS E:\fp2-23b\fase02\trabajo03>
```

- Como se vio en el código de la clase Ejercicio1, se crearon 2 puntos (1.0,4.5) y (2.0,6.0).
- Entonces al hacer un llamado al método distancia usando estos puntos nos da como resultado lo mostrado.

4.2. Ejercicio 02

- La clase Cilindro hereda de la clase Circulo para de esta forma contener sus atributos y métodos, aparte de ello se implementó el atributo protegido longitud de tipo double.
- Contiene sus getters y setters, además también un método que devuelve un double que contiene el valor de la superficie del cilindro usando la siguiente fórmula:

$$S = 2r(l + r)$$

```java
class Cilindro extends Circulo{
  protected double longitud;

  public void setLongitud(double longitud) {
    this.longitud = longitud;
  }

  public double getLongitud() {
    return longitud;
  }

  public double superficie() {
    return 2 * radio * (longitud + radio);
  }
}
```

- La clase Ejercicio2 contiene el método principal donde se realiza una prueba de la clase Cilindro.
- Se crea un Punto llamado centro y luego el Cilindro para hallar su superficie.

```java
public class Ejercicio2 {
  public static void main(String[] args) {
    Cilindro cilindro = new Cilindro();
    Punto centro = new Punto();
    centro.setX(3.0);
    centro.setY(0.0);

    cilindro.setCentro(centro);
    cilindro.setRadio(2.5);
    cilindro.setLongitud(4.0);

    double superficieCilindro = cilindro.superficie();
    System.out.println("La superficie del cilindro es: " + superficieCilindro);
  }
}
```

- PRUEBA DE EJECUCIÓN DEL CÓDIGO (Ejercicio2.java):

```
PS E:\fp2-23b\fase02\trabajo03> java Ejercicio2
La superficie del cilindro es: 32.5
PS E:\fp2-23b\fase02\trabajo03>
```

- Como se vio en el código de la clase Ejercicio2, se creó el Punto centro (3.0,0.0) y (0.0,0.0).
- Luego el Cilindro cilindro con su radio y longitud.
- Finalmente se hace un llamado para hallar la superficie del cilindro creado y se muestra.

4.3. Ejercicio 03

- La interfaz Barco contiene métodos propios de una un vehículo que anda en el mar como los que se observa.

```
25 interface Barco {
26   void navegar();
27   void detenerse();
28   void sonarSirena();
29   void gestionarTripulacion();
30   void tirarAncla();
31   void comunicarTorreControlMaritima();
32 }
```

- La interfaz Barco contiene métodos propios de una un vehículo que anda en el aire como los que se observa.

```
34 interface Avion {
35   void volar();
36   void aterrizar();
37   void hacerAnuncio();
38   void gestionarPasajeros();
39   void desplegarTrenAterrizaje();
40   void comunicarTorreControlAerea();
41 }
```

- La clase Hidroavion implementa las interfaces Barco y Avion.
- Además contiene atributos propios con sus getters y setters.
- Sobre-escribe los métodos de ambas interfaces las cuales en su mayoría imprimen en consola y algunos utilizan ciertos atributos.

```
43 class Hidroavion implements Barco, Avion {
44   private String modelo;
45   private int velocidad;
46   private int pasajeros;
47   private String ubicacion;
48   private boolean estado;
49
50   public void setModelo(String modelo) {
51     this.modelo = modelo;
52   }
53
54   public String getModelo() {
55     return modelo;
56   }
```

```
57
58  public void setVelocidad(int velocidad) {
59    this.velocidad = velocidad;
60  }
61
62  public int getVelocidad() {
63    return velocidad;
64  }
65
66  public void setPasajeros(int pasajeros) {
67    this.pasajeros = pasajeros;
68  }
69
70  public int getPasajeros() {
71    return pasajeros;
72  }
73
74  public void setUbicacion(String ubicacion) {
75    this.ubicacion = ubicacion;
76  }
77
78  public String getUbicacion() {
79    return ubicacion;
80  }
81
82  @Override
83  public void navegar() {
84    System.out.println("El hidroavin est navegando en " + ubicacion);
85  }
86
87  @Override
88  public void detenerse() {
89    System.out.println("El hidroavin est " + (estado ? "en movimiento":"detenido"));
90  }
91
92  @Override
93  public void sonarSirena() {
94    System.out.println("El hidroavin est sonando la sirena.");
95  }
96
97  @Override
98  public void volar() {
99    System.out.println("El hidroavin est volando en el aire.");
100 }
101
102 @Override
103 public void aterrizar() {
104   System.out.println("El hidroavin est aterrizando.");
105 }
106
107 @Override
108 public void hacerAnuncio() {
109   System.out.println("Anuncio desde el hidroavin a los "+ pasajeros +" pasajeros:
          Bienvenidos a bordo.");
110 }
111
```

```
112   public void mostrarInformacion() {
113     System.out.println("Modelo: " + getModelo() + ", Velocidad: " + getVelocidad());
114   }
115
116   @Override
117   public void gestionarTripulacion() {
118     System.out.println("Gestionando la tripulacin del hidroavin.");
119   }
120
121   @Override
122   public void tirarAncla() {
123     System.out.println("Tirando el ancla del hidroavin.");
124   }
125
126   @Override
127   public void comunicarTorreControlMaritima() {
128     System.out.println("Comunicndose con la torre de control martima.");
129   }
130
131   @Override
132   public void gestionarPasajeros() {
133     System.out.println("Gestionando a los pasajeros del hidroavin.");
134   }
135
136   @Override
137   public void desplegarTrenAterrizaje() {
138     System.out.println("Desplegando el tren de aterrizaje del hidroavin.");
139   }
140
141   @Override
142   public void comunicarTorreControlAerea() {
143     System.out.println("Comunicndose con la torre de control area.");
144   }
145 }
```

- La clase Ejercicio3 contiene el método principal donde se realiza una prueba de la clase Hidroavion.
- En este se establecen algunos atributos tales como el modelo y la velocidad.
- Finalmente se hace llamado a todos los métodos que contiene la clase.

```
1  public class Ejercicio3 {
2    public static void main(String[] args) {
3      Hidroavion hidroavion = new Hidroavion();
4
5      hidroavion.setModelo("Hidro-3000");
6      hidroavion.setVelocidad(3000);
7
8      hidroavion.navegar();
9      hidroavion.detenerse();
10     hidroavion.sonarSirena();
11     hidroavion.volar();
12     hidroavion.aterrizar();
13     hidroavion.hacerAnuncio();
```

```
14      hidroavion.mostrarInformacion();
15
16      hidroavion.gestionarTripulacion();
17      hidroavion.tirarAncla();
18      hidroavion.comunicarTorreControlMaritima();
19      hidroavion.gestionarPasajeros();
20      hidroavion.desplegarTrenAterrizaje();
21      hidroavion.comunicarTorreControlAerea();
22    }
23  }
```

- PRUEBA DE EJECUCIÓN DEL CÓDIGO (Ejercicio2.java):

```
El hidroavión está navegando en null
El hidroavión está detenido
El hidroavión está sonando la sirena.
El hidroavión está volando en el aire.
El hidroavión está aterrizando.
Anuncio desde el hidroavión a los 0 pasajeros: Bienvenidos a bordo.
Modelo: Hidro-3000, Velocidad: 3000
Gestionando la tripulación del hidroavión.
Tirando el ancla del hidroavión.
Comunicándose con la torre de control marítima.
Gestionando a los pasajeros del hidroavión.
Desplegando el tren de aterrizaje del hidroavión.
Comunicándose con la torre de control aérea.
```

- Como se vio en el código de la clase Ejercicio3, se creó el Hidroavion con algunos atributos, mas no todos ya se observa en la primera linea que el hidroavion está navegando en null, ya que no fue definido si está en aire o mar.
- En cuanto a los demás métodos nos muestran mensajes conforme se implementó en cada método.

4.4. Estructura de trabajo 03

- El contenido que se entrega en este trabajo es el siguiente:

```
trabajo03
|   Ejercicio1.java
|   Ejercicio2.java
|   Ejercicio3.java
|------latex
    |   Informe_Trabajo03.pdf
    |   Informe_Trabajo03.tex
    |------img
    |       Ejercicio1_prueba.jpg
    |       Ejercicio2_prueba.jpg
    |       Ejercicio3_prueba.jpg
    |       Ejercicio3.jpg
    |       logo_episunsa.png
    |       logo_abet.png
    |       logo_unsa.jpg
    |------src
        |   Ejercicio1.java
        |   Ejercicio2.java
        |   Ejercicio3.java
```

5. Rúbricas

5.1. Entregable Informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y fácil de leer.

5.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplió con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo ítems.
- El alumno debe auto calificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	3	
Total		20		19	

6. Referencias

- https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html
- https://stackoverflow.com/questions/1321122/what-is-an-interface-in-java
- https://es.stackoverflow.com/questions/104165/por-qu%C3%A9-la-herencia-m%C3%BAltiple-no-se-admit