CENTRO UNIVERSITARIO UAEM ATLACOMULCO

MATERIA

PARADIGMAS DE LA PROGRAMACIÓN

DOCENTE

JULIO ALBERTO DE LA TEJA LÓPEZ

ALUMNO

ALEXIS VALENCIA MARTÍNEZ

CARRERA

LICENCIATURA EN INGENIERÍA EN SISTEMAS COMPUTACIONALES

ICO-27

FECHA DE ENTREGA

28/AGOSTO/2023

INTRODUCCIÓN

Hay tres clases (`Habitacion`, `HabitacionIndividual` y `HabitacionDoble`), la clase `Habitacion` es la clase base y tiene atributos como el número de habitación y el precio por noche, así como métodos para calcular el costo total de una estadía y mostrar información sobre la habitación. Las clases `HabitacionIndividual` y `HabitacionDoble` son clases derivadas de la clase `Habitacion` y representan tipos específicos de habitaciones. La clase `HabitacionIndividual` tiene un atributo adicional que indica si la habitación tiene vista al mar, mientras que la clase `HabitacionDoble` tiene un atributo adicional que indica el número de camas en la habitación. Ambas clases sobrescriben el método `mostrarInformacion()` para mostrar información adicional sobre sus atributos específicos. En el método `main`, se crea una instancia de la clase `Habitacion` y se utilizan sus métodos para mostrar información sobre la habitación y calcular el costo total de una estadía de 5 noches. Los resultados principales de esta práctica son la implementación exitosa de la herencia y el polimorfismo en la modelización de un sistema de habitaciones de hotel. Los objetivos de la práctica fueron logrados mediante el uso de aspectos metodológicos clave de la programación orientada a objetos, como la definición de clases, atributos y métodos, así como la implementación de relaciones de herencia entre clases

En resumen, esta práctica demuestra cómo se pueden utilizar los conceptos fundamentales de la programación orientada a objetos para modelar sistemas complejos de manera eficiente y modular

METODOLOGÍA

Para el desarrollo de esta práctica, se utilizaron los siguientes materiales: una computadora con acceso a internet, el entorno de desarrollo en línea Replit y el editor de código Visual Studio Code. El procedimiento que se siguió para el diseño de los cálculos teóricos, la realización de mediciones, obtención de curvas y adquisición de datos fue el siguiente:

1. Se utilizó la computadora para acceder a internet y analizar la práctica asignada por el profesor en la plataforma de teams

2. Se empleó el entorno de desarrollo en línea Replit para escribir y ejecutar el código proporcionado por el profesor, que permite realizar los cálculos teóricos.

3. Se analizó el código proporcionado por el profesor para entender su funcionamiento y cómo se relacionan las clases `Habitacion`, `HabitacionIndividual` y `HabitacionDoble`.

4. Se utilizaron las herramientas disponibles en Replit para realizar las mediciones necesarias y obtener las curvas correspondientes.

5. Se empleó el editor de código Visual Studio Code para escribir y depurar el código que permitió la adquisición de datos, pero sin el resultado esperado (no permitia leer el código.

ACTIVIDAD A REALIZAR


```java
// Clase base: Habitacion
class Habitacion {
    private int numero;
    private double precioPorNoche;

    public Habitacion(int numero, double precioPorNoche) {
        this.numero = numero;
        this.precioPorNoche = precioPorNoche;
    }

    public int getNumero() {
        return numero;
    }

    public double getPrecioPorNoche() {
        return precioPorNoche;
    }

    public void mostrarInformacion() {
        System.out.println("Número de habitación: " + numero);
        System.out.println("Precio por noche: $" + precioPorNoche);
    }
}
```

Script 1. Clase Habitacion (encapsulamiento, herencia y polimorfismo)

```java
// Clase derivada: HabitacionIndividual
class HabitacionIndividual extends Habitacion {
    private boolean tieneVistaAlMar;

    public HabitacionIndividual(int numero, double precioPorNoche, boolean tieneVistaAlMar) {
        super(numero, precioPorNoche);
        this.tieneVistaAlMar = tieneVistaAlMar;
    }

    public boolean tieneVistaAlMar() {
        return tieneVistaAlMar;
    }

    @Override
    public void mostrarInformacion() {
        super.mostrarInformacion();
        System.out.println("Tiene vista al mar: " + (tieneVistaAlMar ? "Sí" : "No"));
    }
}
```

Script 2. Clases HabitacionIndividual (encapsulamiento, herencia y polimorfismo)

```java
// Clase base: Habitacion
class Habitacion {
    private int numero;
    private double precioPorNoche;

    public Habitacion(int numero, double precioPorNoche) {
        this.numero = numero;
        this.precioPorNoche = precioPorNoche;
    }

    public int getNumero() {
        return numero;
    }

    public double getPrecioPorNoche() {
        return precioPorNoche;
    }

    public void mostrarInformacion() {
        System.out.println("Número de habitación: " + numero);
        System.out.println("Precio por noche: $" + precioPorNoche);
    }
}
```

Script 1. Clase Habitacion (encapsulamiento, herencia y polimorfismo)

```java
// Clase derivada: HabitacionIndividual
class HabitacionIndividual extends Habitacion {
    private boolean tieneVistaAlMar;

    public HabitacionIndividual(int numero, double precioPorNoche, boolean tieneVistaAlMar) {
        super(numero, precioPorNoche);
        this.tieneVistaAlMar = tieneVistaAlMar;
    }

    public boolean tieneVistaAlMar() {
        return tieneVistaAlMar;
    }

    @Override
    public void mostrarInformacion() {
        super.mostrarInformacion();
        System.out.println("Tiene vista al mar: " + (tieneVistaAlMar ? "Sí" : "No"));
    }
}
```

Script 2. Clases HabitacionIndividual (encapsulamiento, herencia y polimorfismo)

7. Crear un diagrama de UML a partir del código dado.
8. Agregar los diagramas UML de clases, casos de uso y secuencia.
9. Colocar los códigos y diagramas en GitHub (Uml en PlantUML)
10. Desarrollar el reporte de práctica de acuerdo con el formato dado (anexos).

El profesor:

1. Explicar el funcionamiento de las clases y los modificadores de acceso, y dar un panorama a el alumno de cómo se utilizan en la programación.
2. Corroborará que los alumnos investiguen en la bibliografía proporcionada los conceptos relacionados con las secciones del laboratorio.
3. Asesoría a los alumnos durante el desarrollo de la práctica, para el uso del IDE.
4. En debate se analizarán los resultados obtenidos, así como los problemas que se presenten antes y después de la práctica.

Evaluación.

Se evaluará la participación del alumno durante las demostraciones y la discusión de la presente práctica.

Se evaluará la práctica de acuerdo con la rúbrica en el anexo.

Actividades de integración.

- ¿Para qué se utiliza el modificador **protected**?
- ¿Qué fin tiene el realizar el encapsulamiento de los datos?
- ¿A qué datos es posible acceder desde una clase externa?
- ¿Cuál es la ventaja de la herencia?
- ¿Cómo podemos generalizar el código por medio del polimorfismo?

Referencias

Básico.

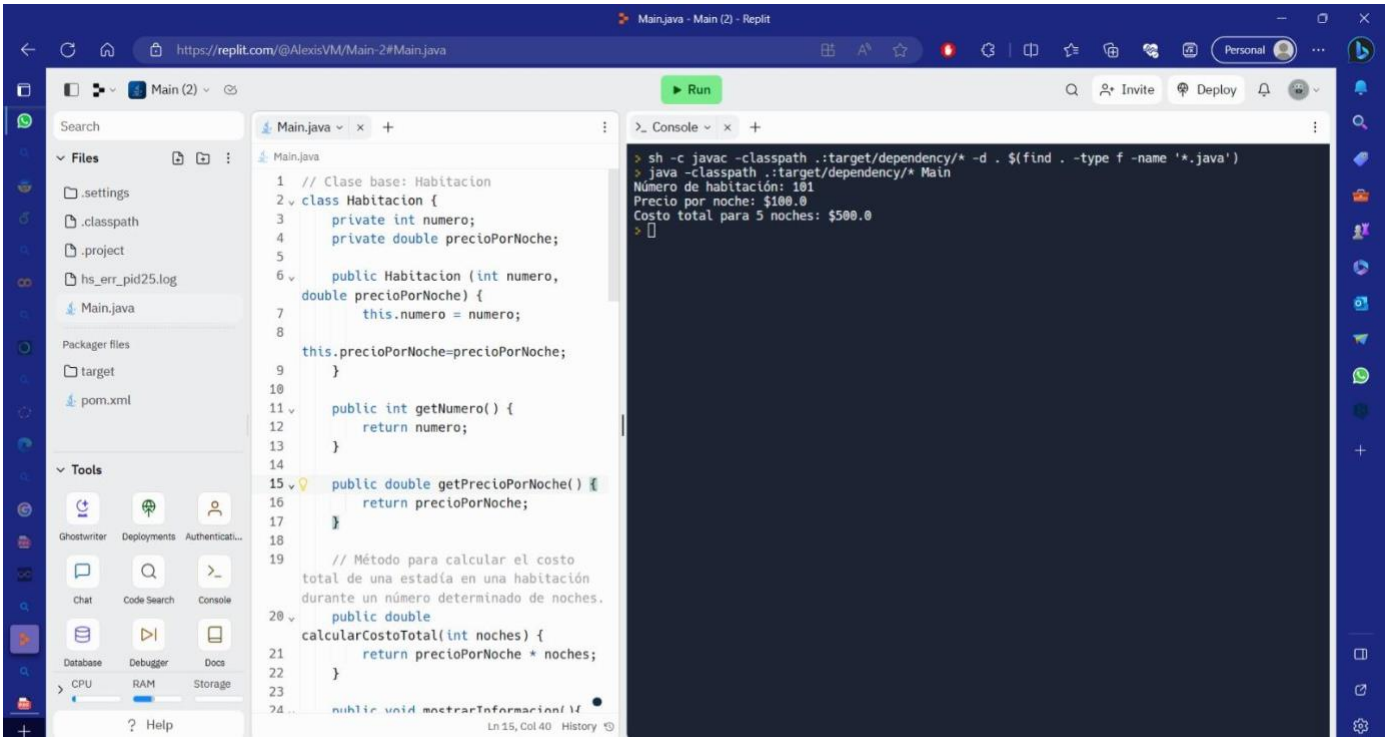
1. Deitel, P. and Deitel, H., 2012. Cómo Programar En Java. México: Pearson Educación.
2. Bell, D. and Parr, M., 2011. Java Para Estudiantes. México: Pearson Educación.

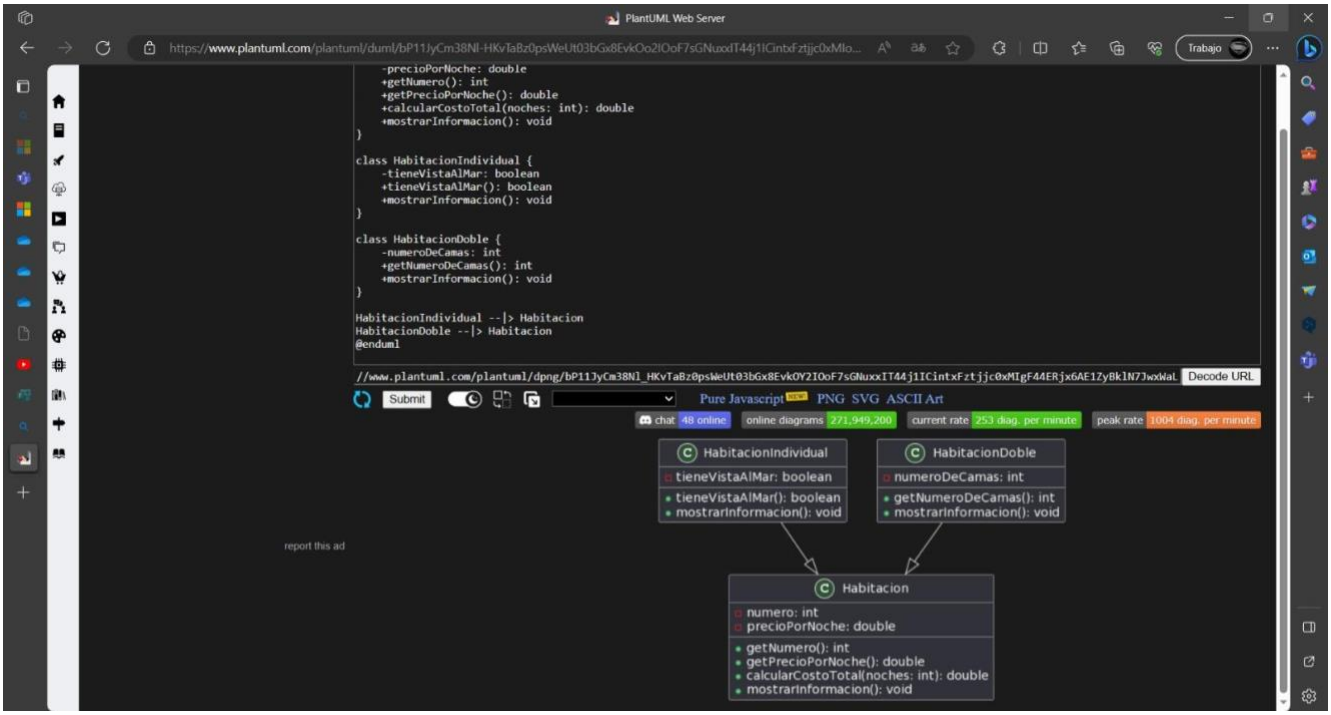
- CODIGO

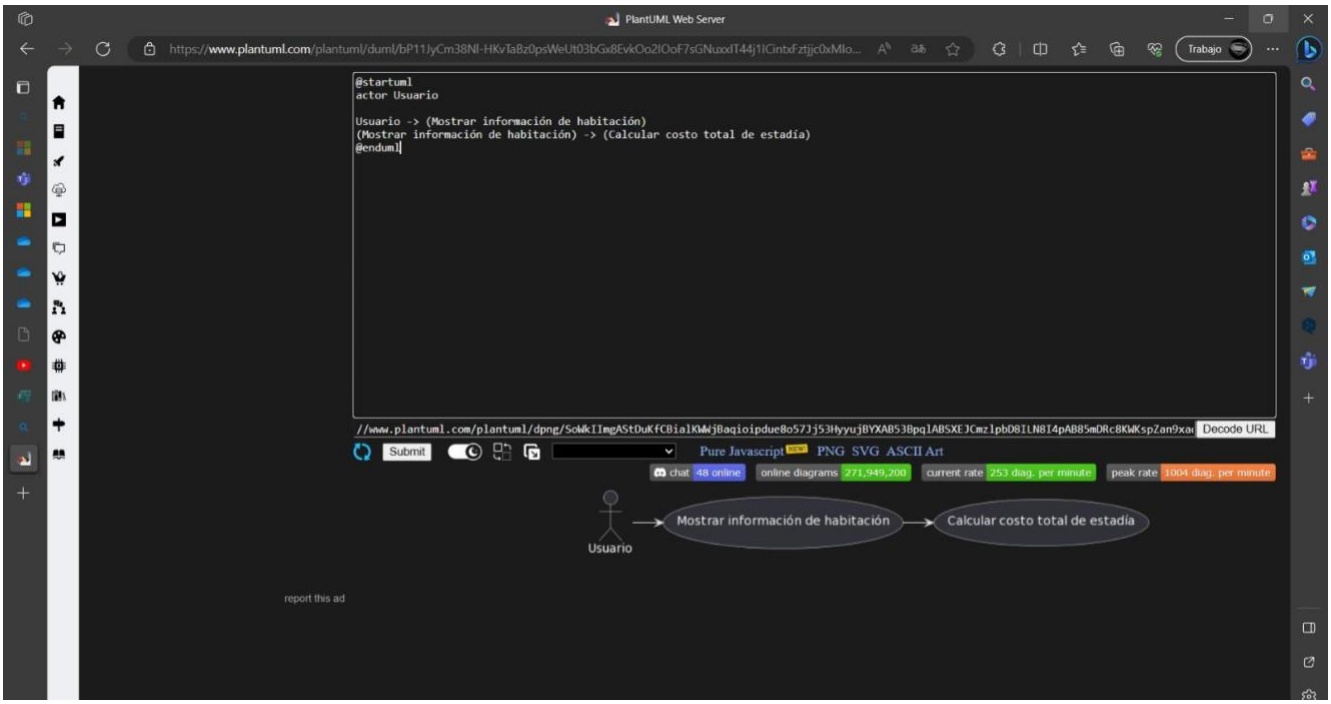
```
// Clase base: Habitacion
class Habitacion {
   private int numero;
   private double precioPorNoche;

   public Habitacion (int numero, double precioPorNoche) {
      this.numero = numero;
      this.precioPorNoche=precioPorNoche;
   }

   public int getNumero() {
      return numero;
   }

   public double getPrecioPorNoche() {
      return precioPorNoche;
   }

   // Método para calcular el costo total de una estadía en una habitación durante un número determinado de
noches.
   public double calcularCostoTotal(int noches) {
      return precioPorNoche * noches;
   }

   public void mostrarInformacion(){
      System.out.println("Número de habitación: "+numero);
      System.out.println("Precio por noche: $"+precioPorNoche);
   }
}

// Clase derivada: HabitacionIndividual
class HabitacionIndividual extends Habitacion {
   private boolean tieneVistaAlMar;

   public HabitacionIndividual(int numero, double precioPorNoche, boolean tieneVistaAlMar){
      super(numero, precioPorNoche);
      this.tieneVistaAlMar=tieneVistaAlMar;
   }

   public boolean tieneVistaAlMar(){
      return tieneVistaAlMar;
   }

   @Override
   public void mostrarInformacion(){
      super.mostrarInformacion();
      System.out.println("Tiene vista al mar: "+(tieneVistaAlMar  ? "Sí":"No"));
```

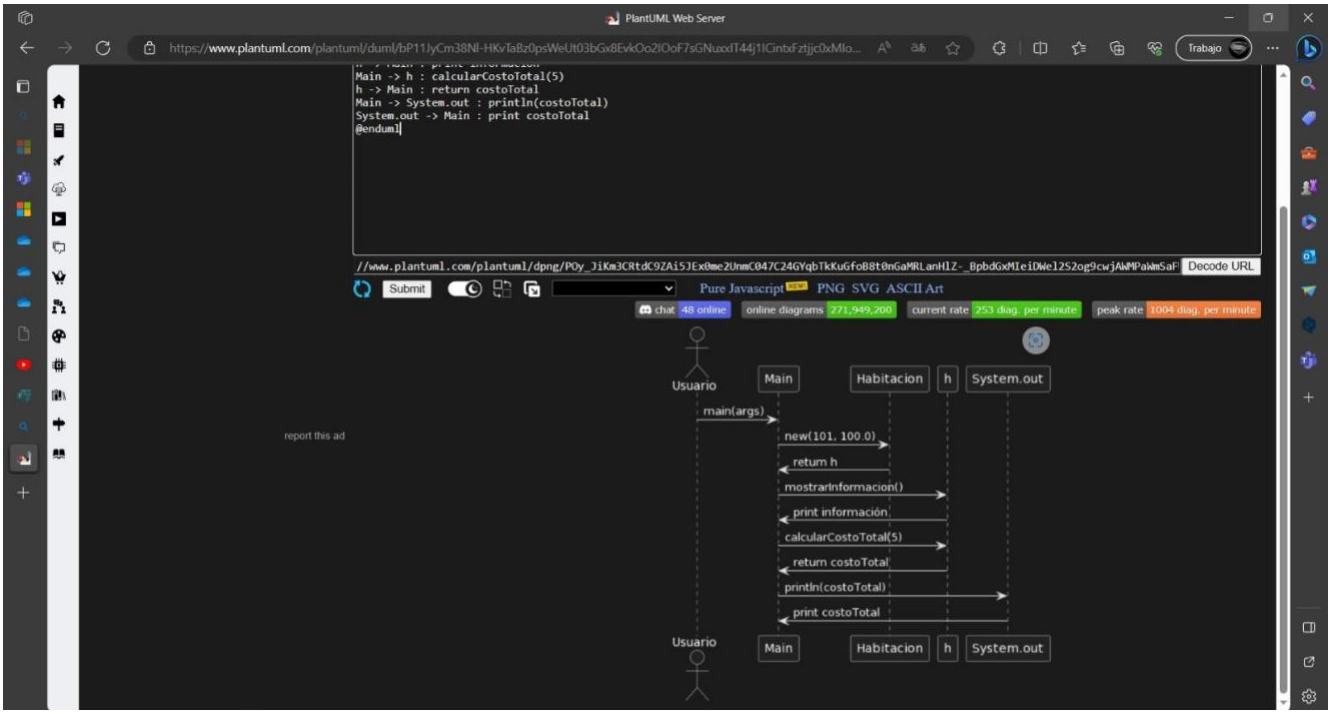
```
    }
}

//Clase derivada: HabitacionDoble
class HabitacionDoble extends Habitacion{
    private int numeroDeCamas;

    public HabitacionDoble(int numero, double precioPorNoche, int numeroDeCamas){
        super(numero, precioPorNoche);
        this.numeroDeCamas=numeroDeCamas;
    }

    public int getNumeroDeCamas(){
        return numeroDeCamas;
    }

    @Override
    public void mostrarInformacion(){
        super.mostrarInformacion();
        System.out.println("Número de camas: "+numeroDeCamas);
    }
}

public class Main {
  public static void main(String[] args) {
      Habitacion h = new Habitacion(101, 100.0);
      h.mostrarInformacion();
      double costoTotal = h.calcularCostoTotal(5);
      System.out.println("Costo total para 5 noches: $" + costoTotal);
  }
}
```

- SALIDA


Main.java - Main (2) - Replit
https://replit.com/@AlexisVM/Main-2#Main.java
Main (2)
Run
Invite
Deploy
Personal
Search
Files
.settings
.classpath
.project
hs_err_pid25.log
Main.java
Packager files
target
pom.xml
Tools
Ghostwriter
Deployments
Authenticati...
Chat
Code Search
Console
Database
Debugger
Docs
CPU
RAM
Storage
Help
Main.java
// Clase base: Habitacion
class Habitacion {
private int numero;
private double precioPorNoche;
public Habitacion (int numero, double precioPorNoche) {
this.numero = numero;
this.precioPorNoche=precioPorNoche;
}
public int getNumero( ) {
return numero;
}
public double getPrecioPorNoche( ) {
return precioPorNoche;
}
// Método para calcular el costo total de una estadía en una habitación durante un número determinado de noches.
public double calcularCostoTotal(int noches) {
return precioPorNoche * noches;
}
Ln 15, Col 40
History
Console
> sh -c javac -classpath .:target/dependency/* -d . $(find . -type f -name '*.java')
> java -classpath .:target/dependency/* Main
Número de habitación: 101
Precio por noche: $100.0
Costo total para 5 noches: $500.0


PlantUML Web Server
https://www.plantuml.com/plantuml/duml/bP11JyCm38Nl-HKvTaBz0psWeUt03bGx8EvkOo2IOoF7sGNuxxIT44j1ICintxFztjjc0xMIo...
Trabajo
-precioPorNoche: double
+getNumero(): int
+getPrecioPorNoche(): double
+calcularCostoTotal(noches: int): double
+mostrarInformacion(): void
}
class HabitacionIndividual {
-tieneVistaAlMar: boolean
+tieneVistaAlMar(): boolean
+mostrarInformacion(): void
}
class HabitacionDoble {
-numeroDeCamas: int
+getNumeroDeCamas(): int
+mostrarInformacion(): void
}
HabitacionIndividual --|> Habitacion
HabitacionDoble --|> Habitacion
@enduml
//www.plantuml.com/plantuml/dpng/bP11JyCm38Nl_HKvTaBz0psWeUt03bGx8EvkOY2IOoF7sGNuxxIT44j1ICintxFztjjc0xMIgF44ERjx6AE1ZyBklN7JwxWaL
Decode URL
Submit
Pure Javascript
PNG SVG ASCII Art
chat 48 online
online diagrams 271,949,200
current rate 253 diag. per minute
peak rate 1004 diag. per minute
HabitacionIndividual
tieneVistaAlMar: boolean
tieneVistaAlMar(): boolean
mostrarInformacion(): void
HabitacionDoble
numeroDeCamas: int
getNumeroDeCamas(): int
mostrarInformacion(): void
report this ad
Habitacion
numero: int
precioPorNoche: double
getNumero(): int
getPrecioPorNoche(): double
calcularCostoTotal(noches: int): double
mostrarInformacion(): void


PlantUML Web Server
@startuml
actor Usuario
Usuario -> (Mostrar información de habitación)
(Mostrar información de habitación) -> (Calcular costo total de estadía)
@enduml
Submit
Pure Javascript
PNG SVG ASCII Art
Usuario
Mostrar información de habitación
Calcular costo total de estadía
report this ad


PlantUML Web Server
Main -> h : calcularCostoTotal(5)
h -> Main : return costoTotal
Main -> System.out : println(costoTotal)
System.out -> Main : print costoTotal
@enduml
Submit
Pure Javascript
PNG SVG ASCII Art
Usuario
Main
Habitacion
h
System.out
main(args)
new(101, 100.0)
return h
mostrarInformacion()
print información
calcularCostoTotal(5)
return costoTotal
println(costoTotal)
print costoTotal
report this ad

¿PARA QUE SE UTILIZA EL MODIFICADOR protected?
El modificador protected se utiliza para controlar el acceso a los miembros de una clase. Un miembro protegido es accesible dentro de su clase y por parte de instancias de clase derivadas. Esto significa que solo las clases que heredan de la clase en la que se declara el miembro protegido pueden acceder a él

¿QUÉ FIN TIENE EL REALIZAR EL ENCAPSULAMIENTO DE LOS DATOS?
El encapsulamiento de datos es un mecanismo para reunir datos y métodos dentro de una estructura ocultando la implementación del objeto, es decir, impidiendo el acceso a los datos por cualquier medio que no sean los servicios propuestos. La encapsulación permite, por tanto, garantizar la integridad de los datos contenidos en el objeto

¿A QUÉ DATOS ES POSIBLE ACCEDER DESDE UNA CLASE EXTERNA?
Desde una clase externa, es posible acceder a los datos o métodos de una clase que se definen con el nivel de acceso public. Los miembros públicos son accesibles desde cualquier lugar del código, incluyendo desde clases externas

¿CUÁL ES LA VENTAJA DE LA HERENCIA?
La herencia tiene varias ventajas en la programación orientada a objetos. Una ventaja importante es que promueve la reutilización del código al permitir que las clases hereden propiedades y métodos de otras clases. Esto puede reducir la redundancia del código y mejorar la confiabilidad y extensibilidad del mismo

¿CÓMO PODEMOS GENERALIZAR EL CÓDIGO POR MEDIO DEL POLIMORFISMO?
El polimorfismo nos permite escribir código más genérico y flexible al permitir que diferentes objetos sean tratados como si fueran del mismo tipo. Esto facilita la creación de código modular y escalable, ya que podemos escribir código que funciona con objetos de diferentes tipos sin tener que preocuparnos por sus detalles específicos. Por ejemplo, podemos tener una función que acepte un objeto `Shape` como parámetro y llame a su método `draw()`, sin tener que preocuparnos por si el objeto es un `Circle`, un `Rectangle` o cualquier otro tipo de forma. Esto nos permite generalizar el código y hacerlo más fácil de leer y mantener

CONCLUSIÓN
En conclusión, la práctica en cuestión demuestra el uso efectivo de la herencia y el encapsulamiento en la programación orientada a objetos. La herencia permite una estructura clara y modular del código, facilitando su comprensión y mantenimiento, mientras que el encapsulamiento ayuda a ocultar los detalles de implementación de un objeto y proporcionar una interfaz pública para interactuar con ese objeto

Rubrica.

Nivel / Criterios (puntaje)	Excelente (90 -100%)	Bueno (75 – 89 %)	Suficiente (60 -74 %)	Inadecuado (0 – 59 %)
Formato (5)	El formato del documento corresponde al mostrado en el anexo 1, con un texto a doble columna, 4-6 páginas de contenido, haciendo uso de un tamaño adecuado de texto y distribución de la información.	El formato del documento corresponde al mostrado en el anexo 1, pero la extensión de documento es muy breve o extensa.	El formato del documento corresponde al mostrado en el anexo 1, y la información se encuentra correctamen-te distribuida, pero la extensión y el tamaño de fuente en texto, tablas y figuras no son adecuados.	El documento no tiene un formato aceptable, o la información se encuentra dispersa y confusa.
Resumen (5)	Presenta los aspectos más relevantes de la práctica y la metodología utilizada, enfatizando en la contribución propia.	El resumen presenta los aspectos más relevante s de la práctica, pero solamente se identifican de manera parcial los aspectos metodológicos o la contribución propia.	El resumen presenta solamente los resultados más sobresalientes sin mencionar cuales aspectos metodológicos hicieron posibles dichos resultados.	No se identifican ni los aspectos más importantes de los resultados ni de la metodología utilizada.
Introducción (10)	Muestra un adecuado análisis de la literatura consultada, incluyendo un marco teórico fundamentado en referencias confiables y recientes.	Muestra un adecuado análisis de la literatura consultada, pero el marco teórico no se encuentra debidamente fundamenta-do en referencias confiables y recientes.	Se presenta un marco teórico, pero no un análisis de la literatura consultada.	No se realizó un análisis de la literatura ni se presenta un marco teórico adecuado.
Metodología (20)	La metodología presentada es clara, presenta claramente el procedimiento utilizado, los materiales empleados para el ensamble de circuitos, modelos, etc., apoyados en cálculos teóricos correctos.	La metodología presentada es poco clara, pero existe un procedimiento documentado, apoyado en cálculos teóricos correctos.	La metodología es clara, pero existen algunos errores menores en los cálculos o procedimiento de diseño, que no comprome-ten gravemente los resultados de la práctica.	La metodología no es clara, o los cálculos empleados para soportar el diseño no son correctos, comprometiendo los resultados obtenidos.
Resultados (20)	Los resultados son claros y demostrables, mediante el apoyo de gráficas, oscilogramas, y fotografías.	Los resultados son claros y demostrables, pero el soporte documental debe ser mejorado para una correcta interpretación.	Los resultados obtenidos no son muy claros, pero existe un soporte documental de los mismos o la interpreta-ción no es completamente adecuada.	Los resultados obtenidos no son soportados o no se documentaron.
Conclusiones (5)	Las conclusiones reflejan el conocimiento adquirido por los estudiantes, y exhiben una discusión de los resultados más trascendentales.	Las conclusiones reflejan el conocimiento adquirido por los estudiantes, pero la discusión de los resultados más trascendentales es ambigua.	Las conclusiones reflejan el conocimiento adquirido por los estudiantes, pero su interpretación no es completamente correcta.	Las conclusiones son vagas y no demuestran una comprensión de los resultados obtenidos.
Referencias (5)	Las referencias son suficientes en número para el tema en cuestión, son confiables y se encuentran correctamente citadas en el formato IEEE.	Las referencias son suficientes en número para el tema en cuestión, son confiables, pero no se encuentran correctamente citadas en el formato IEEE.	Se tienen algunas referencias confiables y otras que no pueden ser corroboradas de manera total.	No se tienen referencias o éstas provienen de fuentes que no son confiables.
Cumplimiento de los objetivos (20)	Los objetivos de la práctica fueron cubiertos en su totalidad, resultando un circuito completamente funcional y congruente con los datos que se presentan en el documento escrito.	Los objetivos de la práctica fueron cubiertos en su totalidad, pero el circuito no es completamente funcional. Los resultados son congruentes con los datos que se presentan en el documento escrito.	Los objetivos de la práctica fueron parcialmente cubiertos, y el circuito presenta fallas o incongruencias notorias con los resultados documentados.	Los objetivos de la práctica no pudieron ser cubiertos o éstos no son congruentes con lo reportado en el documento escrito.
Participación del alumno en la discusión (10)	El alumno (en prácticas individuales) o la mayoría de los integrantes del equipo que conforman (en prácticas grupales) son capaces de desarrollar una discusión del tema, contribuyendo con ideas propias.	El alumno o solamente algunos de los integrantes del equipo que conforman son capaces de desarrollar una discusión del tema, pero existen algunos conceptos que requieren ser fundamentados.	El alumno o muy pocos de los integrantes del equipo que conforman son capaces de desarrollar una discusión del tema, y en esta discusión existen algunos conceptos vagamente fundamentados.	El/los alumno(s) no son capaces de desarrollar una discusión fundamentada sobre el tema tratado en la práctica.
Total		**100 puntos**		

MEDIOS DE APOYO

Replit

https://replit.com/languages/java10

PlantUML

https://www.plantuml.com/plantuml/duml/bP11JyCm38NI-HKvTaBz0psWeUt03bGx8EvkOo2IOoF7sGNuxxIT44j1ICintxFztjjc0xMIo8q4EVjx6AB168Nzk_FJQxaaLEbyP3j_l2bE_2ZQo_YAdGzI1iA5tRoWzOk-sVnCpB_7QcYYrTW8D1O2tKasUHG3QdZMvMN9IfmcaO8U-5aqBLRdTIU9mNswNp4E7E8fXWBq7SWYCZx5R7173w2T7qG8WS_RLxJPLFpQwtiP2EiczxY31Bckfxz4LNV_t_ipPzj-t5vuS6ixQ-wsoA4a-W80