|  |  |  |  |
| --- | --- | --- | --- |
| Materia: | Programacion | | |
| Nivel: | 2 | | |
| Tipo de Examen: | Parcial | | |
| Apellido**(1)**: | Sanchez | Fecha: | 18 oct 2024 |
| Nombre/s**(1)**: | Luciano | Docente a cargo**(2)**: |  |
| División**(1)**: | 122 | Nota**(2)**: |  |
| DNI**(1)**: | 43057393 | Firma**(2)**: |  |

**(1)** Campos a completar solo por el estudiante en caso de imprimir este enunciado en papel.

**(2)** Campos a completar solo por el docente en caso de imprimir este enunciado en papel.

**Enunciado:**

Desarrollar un sistema de gestión de servicios de hospedaje y gastronomía necesita ser desarrollado para permitir la creación y manipulación de diferentes tipos de servicios, cada uno con características particulares. El sistema debe manejar la creación de servicios de tipo Gastronomia y Hospedaje, calcular precios finales con descuentos según promociones, y proporcionar funcionalidades para recuperar servicios basados en su estado de promoción y fecha.



**Capa Modelo**

* **Clase Servicio (Superclase abstracta)**
* **Atributos:**
* String codServicio: Identificador único que debe tener una longitud de 6. Implementar excepción si no cumple esta condición.
* double porcentajeDescuento
* boolean enPromocion
* **Métodos:**
* ~ calcularPrecioFinal(LocalDate dia) : double: método abstracto que se implementará en las subclases.
* **Clase Hospedaje (Subclase de Servicio)**
* **Atributos:**
* String hospedaje
* double precioPorNoche
* **Métodos:**
* + calcularPrecioFinal(LocalDate dia) : double: aplica descuento de lunes a viernes si está en promoción.
* **Clase Gastronomia (Subclase de Servicio)**
* **Atributos:**
* String gastronomia
* double precio
* int diaSemDesc: día de la semana en que se aplica el descuento.
* **Métodos:**
* + calcularPrecioFinal(LocalDate dia) : double: aplica el porcentaje de descuento si está en promoción.
* **Clase Sistema**
* **Atributos:**
* List<Servicio> lstServicio: lista de servicios disponibles.
* **Métodos:**
* + traerServicio (String codServicio) : Servicio
* + traerServicio (boolean enPromocion) : List<Servicio>: retorna lista de servicios según parámetro.
* + traerServicio (boolean enPromocion, LocalDate dia) : List<Servicio>: retorna lista de servicios según la fecha.
* + agregarGastronomia(String codServicio, double porcentajeDescuento, boolean enPromocion, String gastronomia, double precio, int diaSemDesc) : boolean: implementar excepción si el objeto ya existe.
* + agregarHospedaje(String codServicio, double porcentajeDescuento, boolean enPromocion, String hospedaje,double precioPorNoche) : boolean: implementar excepción si el objeto ya existe.

**Capa Test**

***Nota:*** *Al comenzar cada test, indicar el número a resolver, ej: System.out.println("1-3”); y luego la implementación del mismo.*

**Test.java**

* **Pruebas de Creación de Objetos:**
* **1-1)** Intentar crear el objeto Gastronomia:
* codServicio=4892, gastronomia=Hamburguesa criolla, precio=180.0, diaSemDesc=4, porcentajeDescuento=15.0, enPromocion=true
* **1-2)** Crear el objeto Gastronomia:
* codServicio=489235, gastronomia=Hamburguesa criolla, precio=180.0, diaSemDesc=4, porcentajeDescuento=15.0, enPromocion=true
* **1-3)** Intentar crear el objeto Hospedaje:
* codServicio=2872, hospedaje=Cabaña 3 personas, precioPorNoche=1500.0, porcentajeDescuento=10.0, enPromocion=true
* **1-4)** Crear el objeto Hospedaje:
* codServicio=287282, hospedaje=Cabaña 3 personas, precioPorNoche=1500.0, porcentajeDescuento=10.0, enPromocion=true
* **Cálculo de Precios Finales:**
* **2-1)** Calcular e imprimir precio final del servicio:
* Gastronomia [489235, porcentajeDescuento=15.0, enPromocion=true, gastronomia=Hamburguesa criolla, precio=180.0, diaSemDesc=4] para el día: 2020-10-28
* **2-2)** Calcular e imprimir precio final del servicio:
* Hospedaje [287282, porcentajeDescuento=10.0, enPromocion=true, hospedaje=Cabaña 3 personas, precioPorNoche=1500.0] para el día: 2020-10-27
* **Agregar Servicios:**
* **3)** Agregar los siguientes objetos e imprimir la lista de servicios:
* Gastronomia [858927, porcentajeDescuento=15.0, enPromocion=true, gastronomia=Milanesa con pure, precio=350.0, diaSemDesc=3]
* Hospedaje [489259, porcentajeDescuento=10.0, enPromocion=true, hospedaje=Habitación triple, precioPorNoche=2200.0]
* Gastronomia [182835, porcentajeDescuento=20.0, enPromocion=false, gastronomia=Gaseosa, precio=120.0, diaSemDesc=3]
* Hospedaje [758972, porcentajeDescuento=15.0, enPromocion=false, hospedaje=Habitación simple, precioPorNoche=1000.0]
* **Consultas de Servicios:**
* **4-1)** Traer todos los objetos Servicios en enPromocion=true.
* **4-2)** Traer todos los objetos Servicios del día 2020-10-28 y enPromocion=true.

**Objetivos de Aprobación No Directa (Calificación de 4 a 5 puntos):**

* **Implementación de Clases y Métodos**: El alumno debe demostrar una correcta implementación de la superclase abstracta Servicio y sus subclases Hospedaje y Gastronomia, asegurando que se utilicen los métodos abstractos de manera adecuada.
* **Acceso a Atributos y Métodos**: El alumno debe demostrar comprensión del encapsulamiento mediante la correcta declaración y uso de modificadores de acceso en las clases, asegurando que los atributos no se expongan innecesariamente.
* **Uso de Excepciones**: El alumno debe implementar y manejar correctamente excepciones, especialmente en la validación de codServicio y al agregar nuevos servicios al sistema. Esto incluye el lanzamiento de excepciones en los casos donde se intenten agregar objetos duplicados.

**Objetivos de Aprobación Directa (Calificación de 6 a 10 puntos):**

* **Aplicación de Polimorfismo**: El alumno debe implementar correctamente el método calcularPrecioFinal en las subclases, mostrando cómo se puede utilizar el polimorfismo para aplicar diferentes lógicas de cálculo de precios según el tipo de servicio.
* **Pruebas Exhaustivas**: El alumno debe proporcionar pruebas bien estructuradas y documentadas que cubran todos los casos de uso planteados en el parcial.
* **Consistencia en la Lógica del Sistema**: El alumno debe asegurar que la lógica del sistema sea consistente y que las listas de servicios se actualicen y consulten correctamente según las especificaciones del parcial, demostrando una comprensión profunda de la interacción entre las distintas clases y métodos.
* **Documentación y Comentarios**: El alumno debe incluir comentarios claros y concisos en el código que expliquen la funcionalidad de las clases y métodos, así como la lógica detrás de las pruebas, lo que demuestra un entendimiento claro del diseño y la estructura del sistema.