# Ejercicios prácticos

|  |
| --- |
| **Asignatura**: Programación |
| **Nombre**: Actividad práctica 2 |
| **Unidades**: 6 y 7 |

## Guía

* Se plantean diferentes problemas que tratan aspectos de las unidades 6 y 7.
* Para aquellos ejercicios que se pida código fuente se debe poner todo el código de cada clase en formato texto dentro del recuadro.
* Los ejercicios pueden tener más de una solución, buscar la óptima, teniendo en cuenta el número de instrucciones, sentencias y estructuras, etc.
* Es recomendable utilizar de apoyo la documentación oficial de java de Oracle. <https://docs.oracle.com/en/java/>

## Ejemplo

Realiza una clase llamada Persona, que tenga los campos nombre y edad. Debe de tener los métodos set y get.

|  |
| --- |
| **public** **class** Persona {  **private** String nombre;  **private** **int** edad;  **public** String getNombre() {  **return** nombre;  }  **public** **void** setNombre(String nombre) {  **this**.nombre = nombre;  }  **public** **int** getEdad() {  **return** edad;  }  **public** **void** setEdad(**int** edad) {  **this**.edad = edad;  }  @Override  **public** String toString() {  **return** "Persona [nombre=" + nombre + ", edad=" + edad + "]";  }  } |

## Enunciado de los ejercicios

## Ejercicio 1.

Realizar una aplicación Java para la gestión de los empleados de empresas. En la aplicación se deben almacenar los datos genéricos de cualquier persona (DNI, nombre, edad) de las cuales se extiende a otro tipo denominado empleado (sueldo, categoría).

Las empresas disponen de su nombre y CIF para poder identificarlas. La relación que hay entre la empresa y el empleado se formaliza mediante un contrato. El contrato tiene la fecha en el que se crea y el tipo de contrato que es (Temporal o Indefinido), sabemos que los tipos de contrato van a ser siempre los mismos, no van a cambiar, con lo cual, la aplicación tiene que poder reflejar esta casuística.

Los contratos deben disponer de la información de la empresa y el empleado al que pertenece, así mismo también nos interesa conocer a partir del trabajador el contrato que tiene, debido a que, cuando se crea el trabajador, automáticamente se crea el contrato. No ocurre lo mismo con la empresa, no es necesario conocer los contratos que se han formalizado porque ya disponen de una aplicación externa que les muestra todos los contratos que tienen.

Todas las clases deben de tener sus métodos de acceso y modificación de sus atributos, además del método *toString* para poder mostrar los datos que tienen los objetos.

Finalmente, genera una clase con el método Main para crear una empresa y tres trabajadores y muestra su contenido.

|  |
| --- |
| package ejercicio1.gestorempleados;  public class Persona {  private String nombre;  private String dni;  private int edad;  public Persona(String nombre, String dni, int edad) {  this.nombre = nombre;  this.dni = dni;  this.edad = edad;  }  public String getNombre() {  return nombre;  }  public void setNombre(String nombre) {  this.nombre = nombre;  }  public String getDni() {  return dni;  }  public void setDni(String dni) {  this.dni = dni;  }  public int getEdad() {  return edad;  }  public void setEdad(int edad) {  this.edad = edad;  }  @Override  public String toString() {  String str = "Nombre: " + this.nombre + "\nDNI: " + this.dni + "\nEdad: " + this.edad;  return str;  }  }  package ejercicio1.gestorempleados;  import java.util.Date;  public class Empleado extends Persona {  private double sueldo;  private String categoria;  private Contrato contrato;  public Empleado(String nombre, String dni, int edad, double sueldo, String categoria, Empresa empresa) {  super(nombre, dni, edad);  this.sueldo = sueldo;  this.categoria = categoria;  this.contrato = crearContrato(empresa); // Con el fin de crear el contrato automáticamente  }  public double getSueldo() {  return sueldo;  }  public void setSueldo(double sueldo) {  this.sueldo = sueldo;  }  public String getCategoria() {  return categoria;  }  public void setCategoria(String categoria) {  this.categoria = categoria;  }  @Override // from Persona  public String toString() {  return super.toString() + "\nSueldo: " + sueldo + "\nCategoria: " + categoria;  }  public Contrato crearContrato(Empresa empresa) {  Contrato nuevoContrato = new Contrato(new Date(), TipoContrato.INDEFINIDO, this);  nuevoContrato.setEmpresa(empresa); // Asignar empresa al contrato  return nuevoContrato;  }  public Contrato getContrato() {  return contrato;  }  }  package ejercicio1.gestorempleados;  public class Empresa {  private String nombre;  private String cif;  public Empresa(String nombre, String cif) {  this.nombre = nombre;  this.cif = cif;  }  public String getNombre() {  return nombre;  }  public String getCif() {  return cif;  }  public void setNombre(String nombre) {  this.nombre = nombre;  }  public void setCif(String cif) {  this.cif = cif;  }  @Override  public String toString() {  return "Empresa{" + "nombre=" + nombre + ", cif=" + cif + '}';  }  }  package ejercicio1.gestorempleados;  import java.util.Date;  enum TipoContrato {  INDEFINIDO, TEMPORAL  }  public class Contrato {  private Date fechaCreacion;  private TipoContrato tipoContrato;  private Empresa empresa;  private Empleado empleado;  public Contrato(Date fechaCreacion, TipoContrato tipoContrato, Empleado empleado) {  this.fechaCreacion = fechaCreacion;  this.tipoContrato = tipoContrato;  this.empleado = empleado;  }  public Date getFechaCreacion() {  return fechaCreacion;  }  public void setFechaCreacion(Date fechaCreacion) {  this.fechaCreacion = fechaCreacion;  }  public TipoContrato getTipoContrato() {  return tipoContrato;  }  public void setTipoContrato(TipoContrato tipoContrato) {  this.tipoContrato = tipoContrato;  }  public Empresa getEmpresa() {  return empresa;  }  public void setEmpresa(Empresa empresa) {  this.empresa = empresa;  }  public Empleado getEmpleado() {  return empleado;  }  public void setEmpleado(Empleado empleado) {  this.empleado = empleado;  }  @Override  public String toString() {  return "Fecha de creacion: " + this.fechaCreacion + "\nTipo de contrato: " + this.tipoContrato  + "\nEmpresa: " + (empresa != null ? empresa.toString() : "No asignada");  }  }  package ejercicio1.gestorempleados;  public class Main {  public static void main(String[] args) {  Empresa empresa = new Empresa("Empresa A", "12345678A");  Empleado empleado1 = new Empleado("11111111A", "Juan", 25, 30000, "Programador", empresa);  Empleado empleado2 = new Empleado("22222222B", "Maria", 30, 40000, "Analista", empresa);  Empleado empleado3 = new Empleado("33333333C", "Pedro", 28, 35000, "Diseñador", empresa);  System.out.println("Empresa: " + empresa);  System.out.println("Empleado 1: " + empleado1);  System.out.println("Contrato Empleado 1: " + empleado1.getContrato());  System.out.println("Empleado 2: " + empleado2);  System.out.println("Contrato Empleado 2: " + empleado2.getContrato());  System.out.println("Empleado 3: " + empleado3);  System.out.println("Contrato Empleado 3: " + empleado3.getContrato());  }  } |

## Ejercicio 2.

Desarrolla una aplicación Java de un sistema de gestión de productos para una tienda que vende varios tipos de productos: electrónicos, belleza y packs. Todos los productos tienen algunas características en común: un código de identificación, un nombre y un precio. Sin embargo, cada tipo de producto tiene diferentes maneras de calcular su precio final para el cliente.

Para los productos electrónicos, se agrega una tarifa de reciclaje al precio; para los de belleza, se agrega un impuesto sobre las ventas; y los paquetes, se componen de uno o dos productos de cualquier tipo, incluso pueden estar compuestos de otros paquetes. El precio final se compone de la suma de todos los precios de los productos que conforman el pack menos un descuento por pack, el descuento se puede cambiar cuando sea necesario.

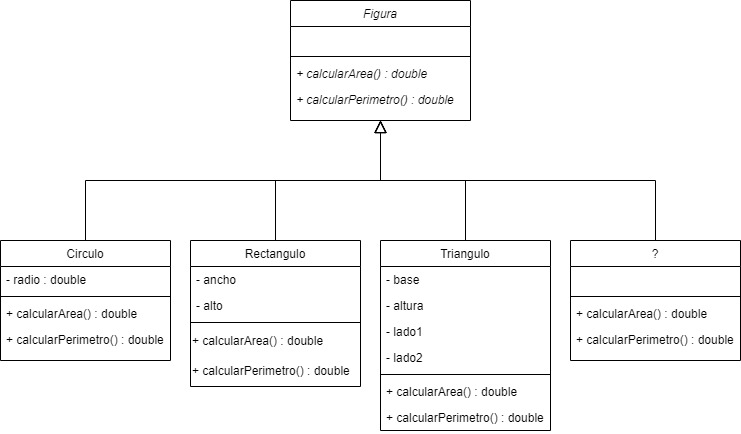
Crea una clase general producto con las características comunes y un método calcularPrecioFinal para que cada producto defina como se calcula su precio.

Finalmente, genera una clase con el método Main para crear una empresa y tres trabajadores y muestra su contenido.

|  |
| --- |
| package ejercicio2.gestorproductos;  public abstract class Producto {  private int codigo;  private String nombre;  private double precio;  public Producto(int codigo, String nombre, double precio) {  this.codigo = codigo;  this.nombre = nombre;  this.precio = precio;  }  public abstract double calcularPrecioFinal();  public int getCodigo() {  return codigo;  }  public void setCodigo(int codigo) {  this.codigo = codigo;  }  public String getNombre() {  return nombre;  }  public void setNombre(String nombre) {  this.nombre = nombre;  }  public double getPrecio() {  return precio;  }  public void setPrecio(double precio) {  this.precio = precio;  }  }  package ejercicio2.gestorproductos;  public class ProductoElectronico extends Producto {  private double tarifaReciclaje;  public ProductoElectronico(int codigo, String nombre, double precio, double tarifaReciclaje) {  super(codigo, nombre, precio);  this.tarifaReciclaje = tarifaReciclaje;  }  @Override  public double calcularPrecioFinal() {  return getPrecio() + tarifaReciclaje;  }  }  package ejercicio2.gestorproductos;  public class ProductoBelleza extends Producto {  private double impuestoVentas;  public ProductoBelleza(int codigo, String nombre, double precio, double impuestoVentas) {  super(codigo, nombre, precio);  this.impuestoVentas = impuestoVentas;  }  @Override  public double calcularPrecioFinal() {  return getPrecio() + (getPrecio() + impuestoVentas);  }  }  package ejercicio2.gestorproductos;  import java.util.ArrayList;  import java.util.List;  public class Pack extends Producto {  private List<Producto> productos;  private double descuentoPorPack;  public Pack(int codigo, String nombre, double descuentoPorPack) {  super(codigo, nombre, 0); // El precio inicial se asigna en 0 ya que el precio final se calcula en base a los productos que contiene.  this.productos = new ArrayList<>();  this.descuentoPorPack = descuentoPorPack;  }  public void agregarProducto(Producto producto) {  productos.add(producto);  }  @Override  public double calcularPrecioFinal() {  double precioTotal = productos.stream().mapToDouble(Producto::calcularPrecioFinal).sum();  return precioTotal - (precioTotal \* descuentoPorPack);  }  }  package ejercicio2.gestorproductos;  public class EmpresaMain {  public static void main(String[] args) {  // Crear productos  ProductoElectronico laptop = new ProductoElectronico(1, "Laptop", 1000, 50);  ProductoBelleza perfume = new ProductoBelleza(2, "Perfume", 50, 0.1);  // Crear paquete  Pack pack = new Pack(3, "Electronics Pack", 0.1);  pack.agregarProducto(laptop);  pack.agregarProducto(perfume);  // Mostrar precios finales  System.out.println("Precio final Laptop: " + laptop.calcularPrecioFinal() + " €.");  System.out.println("Precio final Perfume: " + perfume.calcularPrecioFinal() + " €.");  System.out.println("Precio final Electronics Pack: " + pack.calcularPrecioFinal() + " €.");  }  } |

## Ejercicio 3

A partir del siguiente diagrama de clases realiza la aplicación que permita calcular el área y perímetro de las figuras que se muestran en la imagen. La figura ? es de libre elección.



|  |
| --- |
| package ejercicio3.calculogeometrico;  public interface Figura {  public double calcularArea();  public double calcularPerimetro();  }  package ejercicio3.calculogeometrico;  public class Circulo implements Figura {  private double radio;  public Circulo(double radio) {  this.radio = radio;  }  @Override  public double calcularArea() {  return Math.PI \* Math.pow(this.radio, 2);  }  @Override  public double calcularPerimetro() {  return 2 \* Math.PI \* this.radio;  }  public double getRadio() {  return this.radio;  }  public void setRadio(double radio) {  this.radio = radio;  }  @Override  public String toString() {  return "Circulo [radio=" + radio + "]";  }  }  package ejercicio3.calculogeometrico;  public class Cuadrado implements Figura {  private double lado;  public Cuadrado(double lado) {  this.lado = lado;  }  @Override  public double calcularArea() {  return lado \* lado;  }  @Override  public double calcularPerimetro() {  return 4 \* lado;  }  public double getLado() {  return lado;  }  public void setLado(double lado) {  this.lado = lado;  }  }  package ejercicio3.calculogeometrico;  public class Rectangulo implements Figura {  private double ancho;  private double alto;  public Rectangulo(double ancho, double alto) {  this.ancho = ancho;  this.alto = alto;  }  @Override  public double calcularArea() {  return this.ancho \* this.alto;  }  @Override  public double calcularPerimetro() {  return 2 \* (this.ancho + this.alto);  }  public double getAncho() {  return ancho;  }  public void setAncho(double ancho) {  this.ancho = ancho;  }  public double getAlto() {  return alto;  }  public void setAlto(double alto) {  this.alto = alto;  }  @Override  public String toString() {  return "Rectangulo [ancho=" + ancho + ", alto=" + alto + "]";  }  }  package ejercicio3.calculogeometrico;  public class Main {  public static void main(String[] args) {  // Instanciaciones de las clases  Circulo circulo = new Circulo(5.0);  Cuadrado cuadrado = new Cuadrado(4.0);  Rectangulo rectangulo = new Rectangulo(6.0, 8.0);  // Uso de los métodos  System.out.println("Área del círculo: " + circulo.calcularArea());  System.out.println("Perímetro del círculo: " + circulo.calcularPerimetro());  System.out.println("Área del cuadrado: " + cuadrado.calcularArea());  System.out.println("Perímetro del cuadrado: " + cuadrado.calcularPerimetro());  System.out.println("Área del rectángulo: " + rectangulo.calcularArea());  System.out.println("Perímetro del rectángulo: " + rectangulo.calcularPerimetro());  }  } |

## Ejercicio 4.

Desarrolla una aplicación Java de un sistema de reservas de vuelos y hoteles para que los clientes puedan reservar vuelos y habitaciones en hoteles.

Por cada vuelo se debe conocer el número de vuelo y el número de plazas. Se conoce que el total de plazas se divide entre plazas normales y plazas VIP para clientes exclusivos.

Por otro lado, existen los hoteles que puede ser del tipo de gama alta, de gama media y de gama baja. Todos los **hoteles** comparten atributos comunes como el nombre, el total de habitaciones y el número de plantas pero no puede existir hoteles sin tipo.

En toda la aplicación existen dos procesos comunes, reservar y cancelarReserva, aún que cada sistema define la forma de realizar y cancelar una reserva.

Los clientes pueden reservar vuelos según el número de plazas que están libres, estas pueden ser normales o VIP. Se considera que el vuelo esta completo, si se han ocupado el total de plazas normales y VIPS. Al cancelar una reserva se restablece la plaza siempre y cuando el vuelo no estuviera completo, en este caso, se realiza la cancelación pero no se devuelve la plaza.

Los hoteles de gama alta, a parte de las habitaciones normales, tienen un número de suites, habitaciones dobles y simples, los clientes pueden reservar cualquier tipo según sus gustos siempre y cuando haya disponibles. Los hoteles de gama media, solo tiene habitaciones normales y ofrece la posibilidad de reservar un catering de comida. El catering lo ofrece un restaurante común a todos los hoteles, solo necesitamos conocer el total de clientes que tienen catering. Los hoteles de gama baja solo ofrecen la posibilidad de compartir habitación con otros clientes, las habitaciones están compuestas de 10 camas, de esta forma, para conocer la habitación del cliente se divide el número de reserva de cama entre 10. Al cancelar una reserva se restablece la habitación que había ocupado y cualquier otro servicio.

Finalmente, genera una clase con el método Main con vuelos y hoteles con simulaciones de reservas y cancelaciones para comprobar que están definidos.

|  |
| --- |
| package ejercicio4.sistemareservas;  public class Vuelo {  private int numeroVuelo;  private TipoPlaza tipoPlaza;  private int plazasNormales;  private int plazasVIP;  enum TipoPlaza {  NORMALES, VIP  }  public Vuelo(int numeroVuelo, TipoPlaza tipoPlaza, int plazasNormales, int plazasVIP) {  this.numeroVuelo = numeroVuelo;  this.tipoPlaza = tipoPlaza;  this.plazasNormales = plazasNormales;  this.plazasVIP = plazasVIP;  }  public void reservar(int cantidad) {  // Lógica de reserva según el tipo de plaza  switch (tipoPlaza) {  case NORMALES:  if (plazasNormales >= cantidad) {  System.out  .println("Reserva exitosa de " + cantidad + " plazas normales en el vuelo " + numeroVuelo);  plazasNormales -= cantidad;  } else {  System.out.println("No hay suficientes plazas normales disponibles en el vuelo " + numeroVuelo);  }  break;  case VIP:  if (plazasVIP >= cantidad) {  System.out.println("Reserva exitosa de " + cantidad + " plazas VIP en el vuelo " + numeroVuelo);  plazasVIP -= cantidad;  } else {  System.out.println("No hay suficientes plazas VIP disponibles en el vuelo " + numeroVuelo);  }  break;  default:  System.out.println("Tipo de plaza no válido en el vuelo " + numeroVuelo);  }  }  public void cancelarReserva(int cantidad) {  int totalPlazasDisponibles = plazasNormales + plazasVIP;  // Lógica de cancelación de reserva según el tipo de plaza  switch (tipoPlaza) {  case NORMALES:  if (plazasNormales + cantidad <= totalPlazasDisponibles) {  System.out.println(  "Cancelación exitosa de " + cantidad + " plazas normales en el vuelo " + numeroVuelo);  plazasNormales += cantidad;  } else {  System.out.println("No se puede cancelar la reserva especificada en el vuelo " + numeroVuelo);  }  break;  case VIP:  if (plazasVIP + cantidad <= totalPlazasDisponibles) {  System.out.println("Cancelación exitosa de " + cantidad + " plazas VIP en el vuelo " + numeroVuelo);  plazasVIP += cantidad;  } else {  System.out.println("No se puede cancelar la reserva especificada en el vuelo " + numeroVuelo);  }  break;  default:  System.out.println("Tipo de plaza no válido en el vuelo " + numeroVuelo);  }  }  }  package ejercicio4.sistemareservas;  public class Cliente {  private String dni;  private String nombreCompleto;  public Cliente(String dni, String nombreCompleto) {  this.dni = dni;  this.nombreCompleto = nombreCompleto;  }  public String getDni() {  return dni;  }  public void setDni(String dni) {  this.dni = dni;  }  public String getNombreCompleto() {  return nombreCompleto;  }  public void setNombreCompleto(String nombreCompleto) {  this.nombreCompleto = nombreCompleto;  }  @Override  public String toString() {  return "Cliente [dni=" + dni + ", nombreCompleto=" + nombreCompleto + "]";  }  }  package ejercicio4.sistemareservas;  import ejercicio4.sistemareservas.Hotel.TipoHabitacion;  public class ReservaHoteles {  private int identificadorReserva;  private Cliente cliente;  private TipoHabitacion tipoHabitacion;  public ReservaHoteles(int numero, Cliente cliente, TipoHabitacion tipoHabitacion) {  this.identificadorReserva = numero;  this.cliente = cliente;  this.tipoHabitacion = tipoHabitacion;  }  public int getIdentificadorReserva() {  return identificadorReserva;  }  public Cliente getCliente() {  return cliente;  }  public TipoHabitacion getTipoHabitacion() {  return tipoHabitacion;  }  }  package ejercicio4.sistemareservas;  import java.util.ArrayList;  import java.util.List;  public class HotelAltaGama extends Hotel {  private int suitesDisponibles;  private int habitacionesDoblesDisponibles;  private int habitacionesSimplesDisponibles;  private int suitesReservadas;  private int habitacionesNormalesDisponibles;  private int habitacionesNormalesReservadas;  private int habitacionesDoblesReservadas;  private int habitacionesSimplesReservadas;  private List<ReservaHoteles> reservas;  public HotelAltaGama(String nombre, int totalHabitaciones, int suites, int habitacionesDobles,  int habitacionesSimples, int habitacionesNormalesDisponibles) {  super(nombre, totalHabitaciones);  this.suitesDisponibles = suites;  this.habitacionesDoblesDisponibles = habitacionesDobles;  this.habitacionesSimplesDisponibles = habitacionesSimples;  this.habitacionesNormalesDisponibles = habitacionesNormalesDisponibles;  this.habitacionesNormalesReservadas = 0;  this.suitesReservadas = 0;  this.habitacionesDoblesReservadas = 0;  this.habitacionesSimplesReservadas = 0;  this.reservas = new ArrayList<>();  }  @Override  public void reservar(Cliente cliente, TipoHabitacion tipoHabitacion) {  int numeroReserva = reservas.size() + 1;  ReservaHoteles reserva = new ReservaHoteles(numeroReserva, cliente, tipoHabitacion);  reservas.add(reserva);  switch (tipoHabitacion) {  case SUITE:  suitesDisponibles--;  break;  case HABITACION\_DOBLE:  habitacionesDoblesDisponibles--;  break;  case HABITACION\_SIMPLE:  habitacionesSimplesDisponibles--;  break;  case HABITACION\_NORMAL:  habitacionesNormalesDisponibles--;  break;  default:  System.out.println("Tipo de habitación no válido");  }  System.out.println("Se ha reservado una habitación de tipo " + tipoHabitacion + " para " + cliente  + " en el hotel de alta gama " + getNombre());  }  @Override  public void cancelarReserva(Cliente cliente, int identificadorReserva) {  // Lógica de cancelación de reserva en hotel de alta gama  for (ReservaHoteles reserva : reservas) {  if (reserva.getIdentificadorReserva() == identificadorReserva && reserva.getCliente().equals(cliente)) {  // Encontrar la reserva asociada al cliente y número de reserva  reservas.remove(reserva);  switch (reserva.getTipoHabitacion()) {  case SUITE:  suitesDisponibles++;  break;  case HABITACION\_DOBLE:  habitacionesDoblesDisponibles++;  break;  case HABITACION\_SIMPLE:  habitacionesSimplesDisponibles++;  break;  case HABITACION\_NORMAL:  habitacionesNormalesDisponibles++;  break;  default:  System.out.println("Tipo de habitación no válido");  }  System.out  .println(cliente + " ha cancelado la reserva " + identificadorReserva + " en el hotel "  + getNombre());  return;  }  }  System.out.println(  "No se encontró la reserva asociada al cliente " + cliente + " con el número " + identificadorReserva);  }  public int getSuitesDisponibles() {  return suitesDisponibles;  }  public void setSuitesDisponibles(int suitesDisponibles) {  this.suitesDisponibles = suitesDisponibles;  }  public int getHabitacionesDoblesDisponibles() {  return habitacionesDoblesDisponibles;  }  public void setHabitacionesDoblesDisponibles(int habitacionesDoblesDisponibles) {  this.habitacionesDoblesDisponibles = habitacionesDoblesDisponibles;  }  public int getHabitacionesSimplesDisponibles() {  return habitacionesSimplesDisponibles;  }  public void setHabitacionesSimplesDisponibles(int habitacionesSimplesDisponibles) {  this.habitacionesSimplesDisponibles = habitacionesSimplesDisponibles;  }  public int getSuitesReservadas() {  return suitesReservadas;  }  public void setSuitesReservadas(int suitesReservadas) {  this.suitesReservadas = suitesReservadas;  }  public int getHabitacionesNormalesDisponibles() {  return habitacionesNormalesDisponibles;  }  public void setHabitacionesNormalesDisponibles(int habitacionesNormalesDisponibles) {  this.habitacionesNormalesDisponibles = habitacionesNormalesDisponibles;  }  public int getHabitacionesNormalesReservadas() {  return habitacionesNormalesReservadas;  }  public void setHabitacionesNormalesReservadas(int habitacionesNormalesReservadas) {  this.habitacionesNormalesReservadas = habitacionesNormalesReservadas;  }  public int getHabitacionesDoblesReservadas() {  return habitacionesDoblesReservadas;  }  public void setHabitacionesDoblesReservadas(int habitacionesDoblesReservadas) {  this.habitacionesDoblesReservadas = habitacionesDoblesReservadas;  }  public int getHabitacionesSimplesReservadas() {  return habitacionesSimplesReservadas;  }  public void setHabitacionesSimplesReservadas(int habitacionesSimplesReservadas) {  this.habitacionesSimplesReservadas = habitacionesSimplesReservadas;  }  }  package ejercicio4.sistemareservas;  public class HotelGamaMedia extends Hotel {  private boolean cateringDisponible;  private boolean cateringReservado;  public HotelGamaMedia(String nombre, int totalHabitaciones, boolean cateringDisponible) {  super(nombre, totalHabitaciones);  this.cateringDisponible = cateringDisponible;  this.cateringReservado = false;  }  @Override  public void reservar(Cliente cliente, TipoHabitacion tipoHabitacion) {  // Logica reserva hotel gama media  if (tipoHabitacion == TipoHabitacion.HABITACION\_NORMAL && totalHabitaciones > 0) {  totalHabitaciones--;  System.out.println("Se ha reservado una habitación de tipo " + tipoHabitacion + " para " + cliente  + " en el hotel de gama media " + getNombre());  } else {  System.out.println("No se puede reservar la habitación especificada en el hotel " + getNombre());  }  }  @Override  public void cancelarReserva(Cliente cliente, int identificadorReserva) {  // Logica cancelacion reserva hotel gama media  for (ReservaHoteles reserva : reservas) {  if (reserva.getIdentificadorReserva() == identificadorReserva && reserva.getCliente().equals(cliente)) {  totalHabitaciones++;  System.out.println("Se ha cancelado la reserva de una habitación de tipo "  + reserva.getTipoHabitacion() + " para " + cliente + " en el hotel de gama media "  + getNombre());  } else {  System.out.println("No se puede cancelar la reserva especificada en el hotel " + getNombre());  }  }  }  public void reservarCatering(Cliente cliente) {  if (cateringDisponible && !cateringReservado) {  this.cateringReservado = true;  System.out.println("Se ha reservado el catering para " + cliente + " en el hotel de gama media "  + getNombre());  } else {  System.out.println("No se puede reservar el catering en el hotel " + getNombre());  }  }  }  package ejercicio4.sistemareservas;  import java.util.HashMap;  import java.util.Map;  import java.util.Random;  import java.util.Scanner;  public class HotelGamaBaja extends Hotel {  private int camasDisponibles;  private Map<Integer, Integer> habitacionesOcupadas; // Con esto asignamos una relacion clave (numero de cama) con un  // valor (numeros de reserva)  public HotelGamaBaja(String nombre, int totalHabitaciones, int habitacionesDoblesDisponibles) {  super(nombre, totalHabitaciones);  this.camasDisponibles = totalHabitaciones \* 10;  this.habitacionesOcupadas = new HashMap<>();  }  @Override  public void reservar(Cliente cliente, TipoHabitacion tipoHabitacion) {  if (camasDisponibles >= 10) {  // Reserva de habitación compartida (10 camas)  int numeroReservaCama = generarNumeroReservaCama();  int numeroHabitacion = obtenerNumeroReservaCamaDesdeCliente();  habitacionesOcupadas.put(numeroReservaCama, numeroHabitacion);  camasDisponibles -= 10;  System.out.println("Se ha reservado una habitación en el hotel " + getNombre() + " para " + cliente  + " con número de reserva " + numeroReservaCama);  }  }  @Override  public void cancelarReserva(Cliente cliente, int identificadorReserva) {  int numeroReservaCama = obtenerNumeroReservaCamaDesdeCliente();  if (habitacionesOcupadas.containsKey(numeroReservaCama)) {  // Restablecer habitación y cualquier otro servicio  int numeroHabitacion = habitacionesOcupadas.get(numeroReservaCama);  habitacionesOcupadas.remove(numeroReservaCama);  camasDisponibles += 10;  System.out.println("Se ha cancelado la reserva de la habitación " + numeroHabitacion +  " en el hotel de gama baja " + getNombre());  } else {  System.out.println("No se encontró la reserva asociada al número de reserva de cama " + numeroReservaCama);  }  }  private int generarNumeroReservaCama() {  Random random = new Random();  return random.nextInt(1000); // Identificador aleatorio asignado  }  private int obtenerNumeroReservaCamaDesdeCliente() {  // Lógica para obtener el número de reserva de cama del cliente  Scanner scanner = new Scanner(System.in);  try {  System.out.println("Ingrese el número de reserva de cama: ");  return scanner.nextInt();  } catch (Exception e) {  System.out.print("Entrada no válida. Se asignará un número de reserva de cama aleatorio.");  return generarNumeroReservaCama();  } finally {  scanner.close();  }  }  public int getCamasDisponibles() {  return camasDisponibles;  }  public void setCamasDisponibles(int camasDisponibles) {  this.camasDisponibles = camasDisponibles;  }  public Map<Integer, Integer> getHabitacionesOcupadas() {  return habitacionesOcupadas;  }  public void setHabitacionesOcupadas(Map<Integer, Integer> habitacionesOcupadas) {  this.habitacionesOcupadas = habitacionesOcupadas;  }  }  package ejercicio4.sistemareservas;  public class Main {  public static void main(String[] args) {  // Crear vuelos  Vuelo vuelo1 = new Vuelo(1, Vuelo.TipoPlaza.NORMALES, 50, 10);  Vuelo vuelo2 = new Vuelo(2, Vuelo.TipoPlaza.VIP, 30, 5);  // Crear hoteles  HotelAltaGama hotelAltaGama = new HotelAltaGama("Hotel de Lujo", 100, 20, 30, 50, 100);  HotelGamaMedia hotelMediaGama = new HotelGamaMedia("Hotel Estándar", 80, true);  HotelGamaBaja hotelBajaGama = new HotelGamaBaja("Hostal Sencillo", 50, 20);  // Crear clientes  Cliente cliente1 = new Cliente("12345678A", "Juan Pérez");  Cliente cliente2 = new Cliente("98765432B", "María López");  // Reservas de vuelos  vuelo1.reservar(5);  vuelo2.reservar(3);  // Cancelar reservas de vuelos  vuelo1.cancelarReserva(2);  vuelo2.cancelarReserva(1);  // Reservas de hoteles  hotelAltaGama.reservar(cliente1, Hotel.TipoHabitacion.SUITE);  hotelMediaGama.reservar(cliente1, Hotel.TipoHabitacion.HABITACION\_NORMAL);  hotelBajaGama.reservar(cliente2, Hotel.TipoHabitacion.HABITACION\_COMPARTIDA);  // Cancelar reservas de hoteles  hotelAltaGama.cancelarReserva(cliente1, 1);  hotelMediaGama.cancelarReserva(cliente1, 2);  hotelBajaGama.cancelarReserva(cliente2, 3);  }  } |