Samsung
TECH INSTITUTE

MÓDULO 2. PROGRAMACIÓN ORIENTADA A OBJETOS CON JAVA

Relación de Problemas Nº 3

Proyecto prCoche (herencia)

Cread la clase `Coche` cuyas instancias mantienen el nombre del coche y su precio (antes de impuestos). El precio total de un coche se calcula aplicando un IVA al precio marcado. Este IVA puede variar (por defecto el IVA es del 16%), pero será siempre el mismo para todos los coches. Proporcionad métodos para cambiar el IVA (`public void setPiva(double)`), calcular el precio total (`public double precioTotal()`) y mostrar el coche como cadena de caracteres con el formato:

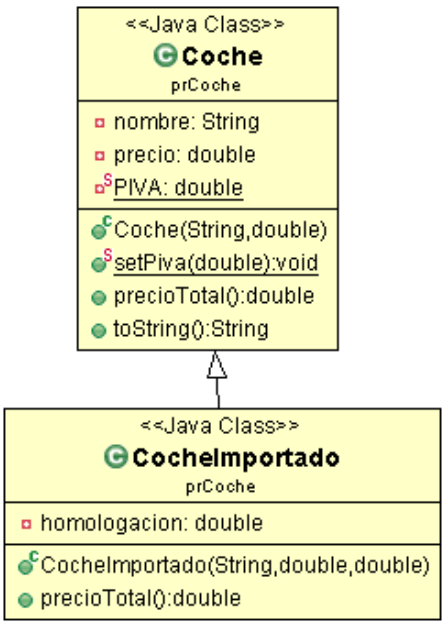
`<nombre> -> <precio_total>`

Un coche importado es un coche para el que además del IVA aplicamos un impuesto de homologación. El impuesto de homologación es específico de cada vehículo, y será dado en el momento de su creación. El precio total de un coche importado se calcula como el de cualquier coche pero ahora hay que sumar este nuevo impuesto. Cread la clase `CocheImportado` para mantener información de coches importados.

Probad las clases con la siguiente clase de prueba:

```
import prCoche.Coche;
import prCoche.Importado;

public class EjCoches {
	static Coche[] cs = {
			new Coche("Seat Panda", 15000),
			new CocheImportado("Ferrari T-R", 65000, 8000),
			new Coche("Seat Toledo", 21000),
			new CocheImportado("Jaguar XK", 41000, 6000),
			new CocheImportado("Porche GT3", 44000, 7000) };

	static public void main(String[] args) {
		for (Coche c : cs) {
			System.out.println(c);
		}

		Coche.setPiva(0.18);
		System.out.println("Con IVA de 18%");

		for (Coche c : cs) {
			System.out.println(c);
		}
	}
}
```

La salida del programa `EjCoches` debe ser:




```
Seat Panda -> 17400.0
Ferrari T-R -> 83400.0
Seat Toledo -> 24360.0
Jaguar XK -> 53560.0
Porche GT3 -> 58040.0
Con IVA de 18%
Seat Panda -> 17700.0
Ferrari T-R -> 84700.0
Seat Toledo -> 24780.0
Jaguar XK -> 54380.0
Porche GT3 -> 58920.0
```

Proyecto prJarrasMezcla (herencia)

Este proyecto usará las clases creadas en el proyecto prJarras. Se trata de construir unas jarras capaces de almacenar agua o vino. Una jarra cuyo contenido es de agua diremos que tiene una pureza de 0 mientras que la que contiene todo vino tiene una pureza de 100. Si una jarra que contiene 3 litros con una pureza de 30 le añadimos 2 litros con una pureza de 80, la jarra finalmente contendrá 5 litros con una pureza de (3*30+2*80)/5 = 50.

Crear la clase `JarraMezcla`, heredera de Jarra que implemente que además de contenido y capacidad, contiene una variable de tipo double que almacena la pureza.

Esta jarra tendrá el siguiente comportamiento:

- Un constructor que crea una jarra mezcla de una capacidad dada. La jarra se crea sin contenido y con pureza = 0.
- Sobreescribe el método `public void llena()` para que su comportamiento sea que la jarra se llena de agua.
- Un método `public void llenaVino()` que llena la jarra de vino.
- Sobreescribe el método `public void llenaDesde(Jarra)` para que se interprete al argumento como una jarra de agua.
- Un método `public void llenaDesde(JarraMezcla)` que vuelque la jarra argumento sobre la receptora hasta que se llene una o se vacíe la otra.
- Sobreescribe el método `public String toString()` para que además del contenido y la capacidad, muestre la pureza del líquido de la jarra.

Probar la jarra con el siguiente programa:

```
import prJarrasMezcla.JarraMezcla;
import prJarras.Jarra;

public class Main {
	public static void main(String [] args) {
		JarraMezcla ja = new JarraMezcla(4);
		JarraMezcla jb = new JarraMezcla(4);
		ja.llena();
		jb.llenaVino();
		Jarra jTemp = new Jarra(2);
		jTemp.llenaDesde(ja);
		jTemp.vacia();
		jTemp.llenaDesde(jb);
		// Las dos jarras estan con 2 litros
		JarraMezcla ji = new JarraMezcla(1);
		for (int i = 0 ; i < 1000; i++) {
			ji.llenaDesde(ja);
			jb.llenaDesde(ji);
			ji.llenaDesde(jb);
			ja.llenaDesde(ji);
			System.out.println(i + " "+ ja + " " + jb);
		}
	}
}
```

Proyecto prHospitalV1 (arrays, enum, herencia)

El objetivo de este ejercicio es crear una serie de clases que representen una visión simplificada de la estructura organizativa de los hospitales. En esta estructura intervendrán elementos diversos como: pacientes, médicos, departamentos y hospitales, representados en sendas clases: `Paciente`, `Medico`, `Departamento` y `Hospital`. Tanto pacientes como médicos incluyen características (atributos y métodos) comunes, que deberán establecerse en una clase `Persona`. Cada departamento está compuesto por un conjunto de médicos, que organizaremos en un array. Y un hospital estará formado por una colección de departamentos, que también organizaremos en un array.

Las clases y sus relaciones se incluyen en el diagrama que se adjunta a continuación. Los atributos y métodos que se deben incluir en cada clase también se especifican en el diagrama, con el significado que se desprende de forma natural de los nombres utilizados, teniendo en cuenta que:

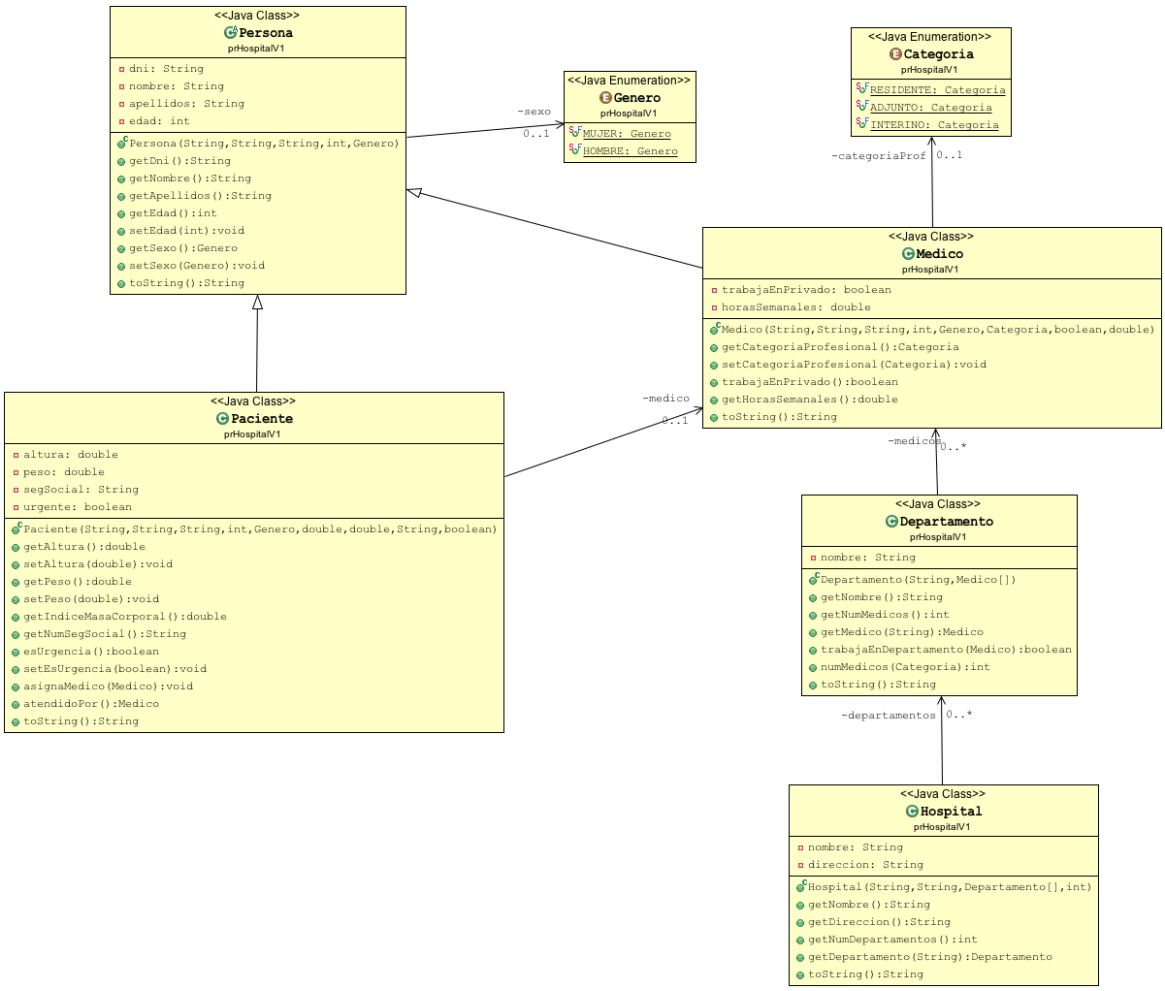
a) El índice de masa corporal en un paciente se calcula dividiendo el peso (en Kg.) por la altura (en metros) al cuadrado.

b) El método `public Medico atendidoPor()` de la clase `Paciente` debe devolver el médico asignado si existe, o `null`, si aún no se ha asignado ningún médico.

c) El método `public int numMedicos(Categoria cat)` de la clase `Departamento` devuelve el número de médicos con la categoría indicada en el argumento.

d) El método `public Medico getMedico(String dni)` de la clase `Departamento` devuelve el objeto Medico con el DNI indicado como argumento.

La práctica consistirá en implementar todas estas clases, cuidando que los identificadores (de clases, variables de instancia y métodos) se correspondan con los que aparecen en el diagrama, y sean acordes a la convención de Java. Obsérvese que la clase `Persona` incluye solo las características propias de una persona física (tal y como se entiende desde un punto de vista jurídico); es decir, lo relevante son los datos jurídicos. Otras características como la altura y el peso son solo relevantes para los pacientes (al menos en el modelo que estamos interesados).


<<Java Class>>
Persona
prHospitalV1
dni: String
nombre: String
apellidos: String
edad: int
Persona(String,String,String,int,Genero)
getDni():String
getNombre():String
getApellidos():String
getEdad():int
setEdad(int):void
getSexo():Genero
setSexo(Genero):void
toString():String
<<Java Enumeration>>
Genero
prHospitalV1
MUJER: Genero
HOMBRE: Genero
-sexo
0..1
<<Java Enumeration>>
Categoria
prHospitalV1
RESIDENTE: Categoria
ADJUNTO: Categoria
INTERINO: Categoria
-categoriaProf
0..1
<<Java Class>>
Medico
prHospitalV1
trabajaEnPrivado: boolean
horasSemanales: double
Medico(String,String,String,int,Genero,Categoria,boolean,double)
getCategoriaProfesional():Categoria
setCategoriaProfesional(Categoria):void
trabajaEnPrivado():boolean
getHorasSemanales():double
toString():String
<<Java Class>>
Paciente
prHospitalV1
altura: double
peso: double
segSocial: String
urgente: boolean
Paciente(String,String,String,int,Genero,double,double,String,boolean)
getAltura():double
setAltura(double):void
getPeso():double
setPeso(double):void
getIndiceMasaCorporal():double
getNumSegSocial():String
esUrgencia():boolean
setEsUrgencia(boolean):void
asignaMedico(Medico):void
atendidoPor():Medico
toString():String
-medico
0..1
-medicos
0..*
<<Java Class>>
Departamento
prHospitalV1
nombre: String
Departamento(String,Medico[])
getNombre():String
getNumMedicos():int
getMedico(String):Medico
trabajaEnDepartamento(Medico):boolean
numMedicos(Categoria):int
toString():String
-departamentos
0..*
<<Java Class>>
Hospital
prHospitalV1
nombre: String
direccion: String
Hospital(String,String,Departamento[],int)
getNombre():String
getDireccion():String
getNumDepartamentos():int
getDepartamento(String):Departamento
toString():String

Para probar las clases, utilícese la clase de prueba, `PruebaHospitalV1`.

```
import prHospitalV1.Categoria;
import prHospitalV1.Departamento;
import prHospitalV1.Genero;
import prHospitalV1.Hospital;
import prHospitalV1.Medico;
import prHospitalV1.Paciente;


public class PruebaHospitalV1 {
	public static void main(String [] args) {
		Medico med1 = new Medico("22626262", "Juan", "Garcia Lopez", 34, Genero.HOMBRE,
				Categoria.ADJUNTO, true, 12);
		Medico med2 = new Medico("73737373", "Pedro", "Marin Perez", 33, Genero.HOMBRE,
				Categoria.INTERINO, true, 12);
		Medico med3 = new Medico("99487474", "Maria", "Velez Lopez", 39, Genero.MUJER,
				Categoria.ADJUNTO, true, 12);
		Medico med4 = new Medico("25252525", "Luisa", "Morilla Antera", 50, Genero.MUJER,
				Categoria.RESIDENTE, false, 12);
		Medico med5 = new Medico("88477444", "Ramon", "Gullon Mande", 46, Genero.HOMBRE,
				Categoria.RESIDENTE, false, 12);
		Medico med6 = new Medico("53536733", "Andres", "Robles Marea", 47, Genero.HOMBRE,
				Categoria.RESIDENTE, true, 12);
		Medico med7 = new Medico("77464564", "Ana", "Peralta Monte", 45, Genero.MUJER,
				Categoria.ADJUNTO, false, 12);
		Medico med8 = new Medico("61261523", "Ramon", "Linde Masa", 34, Genero.HOMBRE,
				Categoria.INTERINO, true, 12);
		Medico med9 = new Medico("99474643", "Joaquin", "Perez Valdes", 59, Genero.HOMBRE,
				Categoria.RESIDENTE, false, 12);

		Departamento dep1 = new Departamento("Oncologia", new Medico[] {med1, med2, med3});
		Departamento dep2 = new Departamento("Radioterapia", new Medico[] {med4, med5});
		Departamento dep3 = new Departamento("Pediatria", new Medico[] {med6, med7, med8, med9});

		Hospital hospital = new Hospital("La Gloria", "Camino del Cielo s/n",
				new Departamento[] {dep1, dep2, dep3}, 5);

		Paciente pac1 = new Paciente("65353535" , "Pedro", "Mal Tratado", 25, Genero.HOMBRE,
				1.80, 76, "1231532514345132513", false);
		Paciente pac2 = new Paciente("88376363" , "Juan", "Mas Malito", 57, Genero.HOMBRE,
				1.83, 86, "38746387467834738467", true);
		Paciente pac3 = new Paciente("44262623" , "Lourdes", "Lisa Manera", 29, Genero.MUJER,
				1.60, 68, "883736363653333345434", false);
		Paciente pac4 = new Paciente("77262562" , "Angela", "Aquime Quedo", 76, Genero.MUJER,
				1.63, 82, "23847283482364723823", false);
		Paciente pac5 = new Paciente("65624524" , "Juan", "Tengo Algo", 20, Genero.HOMBRE,
				1.90, 58, "289378723672836455 28", true);
		Paciente pac6 = new Paciente("11525252" , "Jonh", "Vengod Elejos", 82, Genero.HOMBRE,
				1.64, 73, "34374638746834687865", false);
		pac1.asignaMedico(med1);
		pac2.asignaMedico(med2);
		pac3.asignaMedico(med5);
		pac4.asignaMedico(med8);
		pac5.asignaMedico(med8);
		pac6.asignaMedico(med2);

		// A partir de aquí podemos probar vuestros metodos
		System.out.println(dep1);
		System.out.println("Medicos adjuntos " +
				dep1.numMedicos(Categoria.ADJUNTO));
		Medico me = dep3.getMedico("99474643");
		if (me == null) {
			System.out.println("No existe el medico en ese departamento");
		} else {
			System.out.println(me);
		}
	}
}
```

Proyecto prHospitalV2 (recorridos, herencia, excepciones)

El objetivo de este ejercicio es completar las clases que representen una visión simplificada de la estructura organizativa de los hospitales y que fue objeto de la práctica prHospitalV1. En esta estructura se incorporan nuevos elementos como `Planta`, `Habitacion` y `Cama` y se modifican `Paciente` y `Hospital`. Así, ahora, un paciente siempre tiene asignada una cama hasta que se dé de alta y un hospital incluye además una colección de plantas. Cada planta incluye una colección de habitaciones y cada habitación incluye una colección de camas. Por su parte, una cama, puede contener a un paciente. Todas las colecciones las organizaremos en arrays.

Las clases y sus relaciones se incluyen en el diagrama que se adjunta a continuación. Los atributos y métodos que se deben incluir en cada clase también se especifican en el diagrama, con el significado que se desprende de forma natural de los nombres utilizados, teniendo en cuenta que:

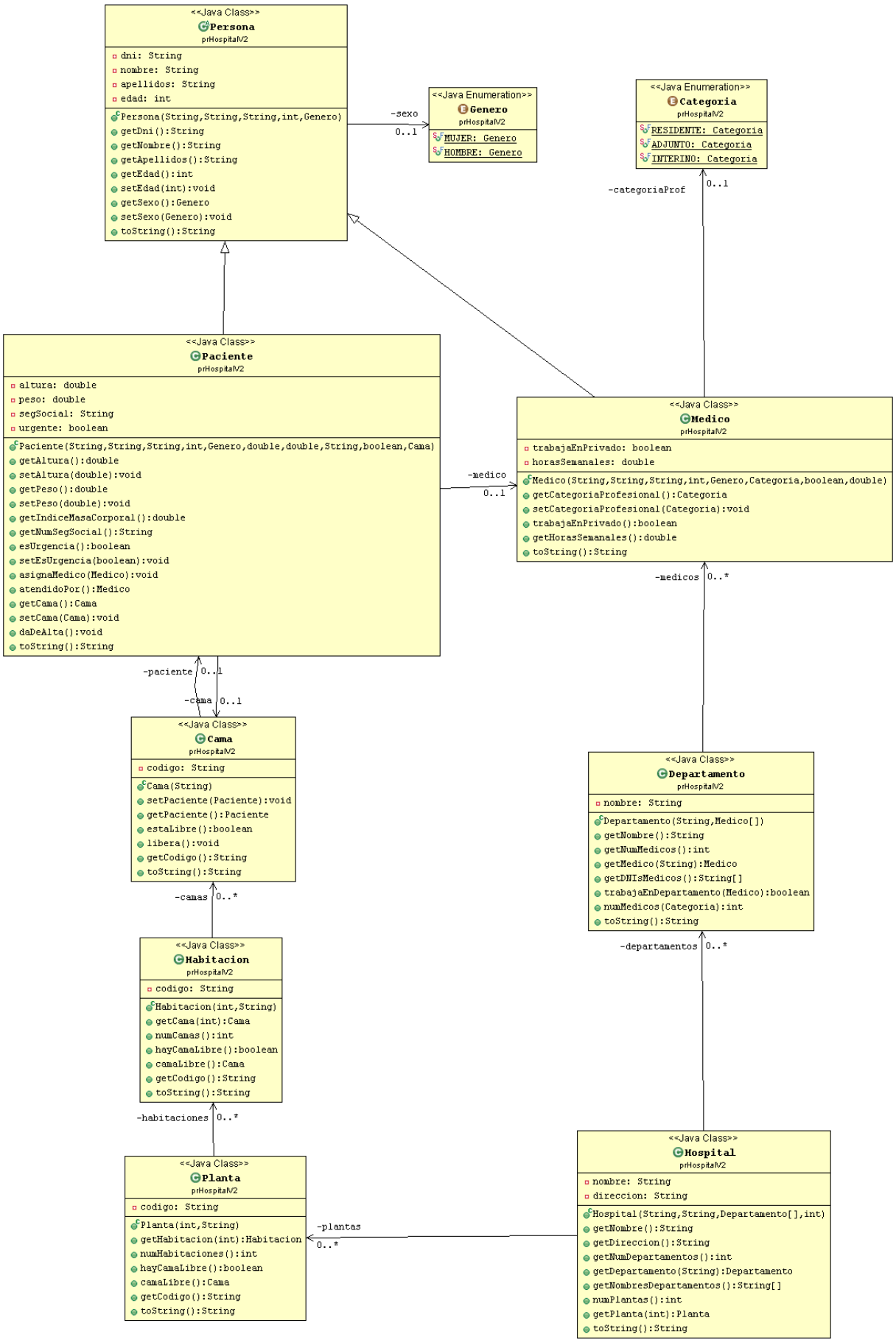
- El constructor de Paciente incluye un parámetro más para proporcionar la cama en la que se instalará el paciente.
- Cada cama puede instalar a lo sumo a un paciente.
- El constructor de hospital incluye un nuevo parámetro que indica el número de plantas que tendrá el hospital.
- La construcción de un hospital incluye la construcción de sus plantas. Un parámetro indica el número de plantas. La clase de prueba que se proporciona incluye la creación de un hospital con 5 plantas numeradas del 0 al 4 y con códigos "P0", "P1", … "P4".
- La construcción de una planta incluirá la construcción de las habitaciones que contiene. Supondremos que cada planta contiene 8 habitaciones numeradas de 0 a 7 y con códigos dependientes de la planta. Por ejemplo, las habitaciones de la planta 3 tendrán por código "P3H0", "P3H1",…,"P3H7".
- La construcción de una habitación incluye la creación de sus camas. Supondremos 4 camas por habitación numeradas de 0 a 3 y sus códigos dependen de la planta y habitación en la que se encuentren. Por ejemplo, las camas de la habitación 4 de la planta 3 serán la "P3H4C0"; "P3H4C1", "P3H4C2" y "P3H4C3".
- Los métodos `public Planta getPlanta(int i)` de la clase `Hospital` y `public Habitacion getHabitacion(int i)` de la clase `Planta` lanzarán una `RuntimeException` si el argumento no está dentro de los rangos válidos.
- El método `public Cama camaLibre()` definido en las clases `Planta` y `Habitacion` proporcionará una cama cualquiera que esté libre. Lanzará una `RuntimeException` si no hay camas libres en donde se solicita.
- El método `setPaciente(Paciente)` de la clase `Cama` debe asegurarse de que la cama está libre (si no lanzar una excepción) y debe mantener consistencia con la clase `Paciente` (asignar al paciente la cama).
- El método `daAlta()` en la clase `Paciente` debe mantener la consistencia con la clase `Cama`.

La práctica consistirá en modificar las clases `Paciente` y `Hospital` e incluir las nuevas clases según el diagrama adjunto.

Para probar las clases, utilícese la clase de prueba, `PruebaHospitalv2`.

Además, complétese con nuevos métodos de clase que resuelvan los siguientes problemas:

- Mostrar todos los pacientes que están tratados en una planta dada.
- Mostrar los médicos que tienen asignado paciente.
- Mostrar todas las camas vacías del hospital.
- Mostrar todas las habitaciones vacías del hospital
- Mostrar todas las habitaciones en las que hay alguna cama libre.
- ¿Cuántos pacientes se están tratando en una planta dada?
- ¿Cuántas camas hay libres en una planta dada?
- ¿Están todas las habitaciones del hospital ocupadas por al menos un paciente?
- ¿Tiene un médico dado asignado algún paciente en una planta dada?
- Mostrar todos los departamentos cuyos médicos tienen pacientes en una planta dada.


<<Java Class>>
Persona
prHospitalV2
dni: String
nombre: String
apellidos: String
edad: int
Persona(String,String,String,int,Genero)
getDni():String
getNombre():String
getApellidos():String
getEdad():int
setEdad(int):void
getSexo():Genero
setSexo(Genero):void
toString():String
-sexo
0..1
<<Java Enumeration>>
Genero
prHospitalV2
MUJER: Genero
HOMBRE: Genero
<<Java Enumeration>>
Categoria
prHospitalV2
RESIDENTE: Categoria
ADJUNTO: Categoria
INTERINO: Categoria
-categoriaProf
0..1
<<Java Class>>
Paciente
prHospitalV2
altura: double
peso: double
segSocial: String
urgente: boolean
Paciente(String,String,String,int,Genero,double,double,String,boolean,Cama)
getAltura():double
setAltura(double):void
getPeso():double
setPeso(double):void
getIndiceMasaCorporal():double
getNumSegSocial():String
esUrgencia():boolean
setEsUrgencia(boolean):void
asignaMedico(Medico):void
atendidoPor():Medico
getCama():Cama
setCama(Cama):void
daDeAlta():void
toString():String
-medico
0..1
<<Java Class>>
Medico
prHospitalV2
trabajaEnPrivado: boolean
horasSemanales: double
Medico(String,String,String,int,Genero,Categoria,boolean,double)
getCategoriaProfesional():Categoria
setCategoriaProfesional(Categoria):void
trabajaEnPrivado():boolean
getHorasSemanales():double
toString():String
-medicos
0..*
-paciente
0..1
-cama
0..1
<<Java Class>>
Cama
prHospitalV2
codigo: String
Cama(String)
setPaciente(Paciente):void
getPaciente():Paciente
estaLibre():boolean
libera():void
getCodigo():String
toString():String
-camas
0..*
<<Java Class>>
Habitacion
prHospitalV2
codigo: String
Habitacion(int,String)
getCama(int):Cama
numCamas():int
hayCamaLibre():boolean
camaLibre():Cama
getCodigo():String
toString():String
-habitaciones
0..*
<<Java Class>>
Planta
prHospitalV2
codigo: String
Planta(int,String)
getHabitacion(int):Habitacion
numHabitaciones():int
hayCamaLibre():boolean
camaLibre():Cama
getCodigo():String
toString():String
<<Java Class>>
Departamento
prHospitalV2
nombre: String
Departamento(String,Medico[])
getNombre():String
getNumMedicos():int
getMedico(String):Medico
getDNIsMedicos():String[]
trabajaEnDepartamento(Medico):boolean
numMedicos(Categoria):int
toString():String
-departamentos
0..*
-plantas
0..*
<<Java Class>>
Hospital
prHospitalV2
nombre: String
direccion: String
Hospital(String,String,Departamento[],int)
getNombre():String
getDireccion():String
getNumDepartamentos():int
getDepartamento(String):Departamento
getNombresDepartamentos():String[]
numPlantas():int
getPlanta(int):Planta
toString():String

```java
import prHospitalV2.Cama;
import prHospitalV2.Categoria;
import prHospitalV2.Departamento;
import prHospitalV2.Genero;
import prHospitalV2.Habitacion;
import prHospitalV2.Hospital;
import prHospitalV2.Medico;
import prHospitalV2.Paciente;
import prHospitalV2.Planta;


public class PruebaHospitalV2 {
	public static void main(String [] args) {
		Medico med1 = new Medico("22626262", "Juan", "Garcia Lopez", 34, Genero.HOMBRE,
				Categoria.ADJUNTO, true, 12);
		Medico med2 = new Medico("73737373", "Pedro", "Marin Perez", 33, Genero.HOMBRE,
				Categoria.INTERINO, true, 12);
		Medico med3 = new Medico("99487474", "Maria", "Velez Lopez", 39, Genero.MUJER,
				Categoria.ADJUNTO, true, 12);
		Medico med4 = new Medico("25252525", "Luisa", "Morilla Antera", 50, Genero.MUJER,
				Categoria.RESIDENTE, false, 12);
		Medico med5 = new Medico("88477444", "Ramon", "Gullon Mande", 46, Genero.HOMBRE,
				Categoria.RESIDENTE, false, 12);
		Medico med6 = new Medico("53536733", "Andres", "Robles Marea", 47, Genero.HOMBRE,
				Categoria.RESIDENTE, true, 12);
		Medico med7 = new Medico("77464564", "Ana", "Peralta Monte", 45, Genero.MUJER,
				Categoria.ADJUNTO, false, 12);
		Medico med8 = new Medico("61261523", "Ramon", "Linde Masa", 34, Genero.HOMBRE,
				Categoria.INTERINO, true, 12);
		Medico med9 = new Medico("99474643", "Joaquin", "Perez Valdes", 59, Genero.HOMBRE,
				Categoria.RESIDENTE, false, 12);

		Departamento dep1 = new Departamento("Oncologia", new Medico[] {med1, med2, med3});
		Departamento dep2 = new Departamento("Radioterapia", new Medico[] {med4, med5});
		Departamento dep3 = new Departamento("Pediatria", new Medico[] {med6, med7, med8, med9});

		Hospital hospital = new Hospital("La Gloria", "Camino del Cielo s/n",
				new Departamento[] {dep1, dep2, dep3}, 5);

		Paciente pac1 = new Paciente("65353535" , "Pedro", "Mal Tratado", 25, Genero.HOMBRE,
				1.80, 76, "1231532514345132513", false, hospital.getPlanta(0).camaLibre());
		Paciente pac2 = new Paciente("88376363" , "Juan", "Mas Malito", 57, Genero.HOMBRE,
				1.83, 86, "38746387467834738467", true,
		hospital.getPlanta(1).getHabitacion(2).camaLibre());
		Paciente pac3 = new Paciente("44262623" , "Lourdes", "Lisa Manera", 29, Genero.MUJER,
				1.60, 68, "88373636365333345434", false, hospital.getPlanta(0).camaLibre());
		Paciente pac4 = new Paciente("77262562" , "Angela", "Aquime Quedo", 76, Genero.MUJER,
				1.63, 82, "23847283482364723823", false,
		hospital.getPlanta(3).getHabitacion(4).camaLibre());
		Paciente pac5 = new Paciente("65624524" , "Juan", "Tengo Algo", 20, Genero.HOMBRE,
				1.90, 58, "28937872367283645528", true, hospital.getPlanta(1).camaLibre());
		Paciente pac6 = new Paciente("11525252" , "Jonh", "Vengod Elejos", 82, Genero.HOMBRE,
				1.64, 73, "34374638746834687865", false, hospital.getPlanta(3).camaLibre());
		pac1.asignaMedico(med1);
		pac2.asignaMedico(med2);
		pac3.asignaMedico(med5);
		pac4.asignaMedico(med8);
		pac5.asignaMedico(med8);
		pac6.asignaMedico(med2);

		// A partir de aquí podeís probar vuestros metodos
		System.out.println(hospital);
		System.out.println(pac1);
		muestraMedicosEnPlanta(hospital, 3);

	}

	/**
	 * Muestra los medicos que tratan pacientes en la planta dada del hospital
	 * @param hop                EL hospital
	 * @param numPlanta El numero de la planta
	 */
	public static void muestraMedicosEnPlanta(Hospital hop, int numPlanta) {
		Planta planta = hop.getPlanta(numPlanta);
		for (int i = 0; i < planta.numHabitaciones(); i++ ) {
			Habitacion habitacion = planta.getHabitacion(i);
			for(int j = 0; j < habitacion.numCamas(); j++) {
				Cama cama = habitacion.getCama(j);
				if (!cama.estaLibre()) {
					Paciente paciente = cama.getPaciente();
					Medico medico = paciente.atendidoPor();
					if (medico != null) {
						System.out.println(medico);
					}
				}
			}
		}
	}
}
```