Trabajo Práctico

Diagrama de Clases

Lista de Conceptos Tratados:

Clase; Atributo; Método; Operación; Relaciones entre Clases: Generalización/Especialización, Asociación, Agregación, Composición, Realización, Instanciación; Roles y Multiplicidad en una relación

Ejercicio 1

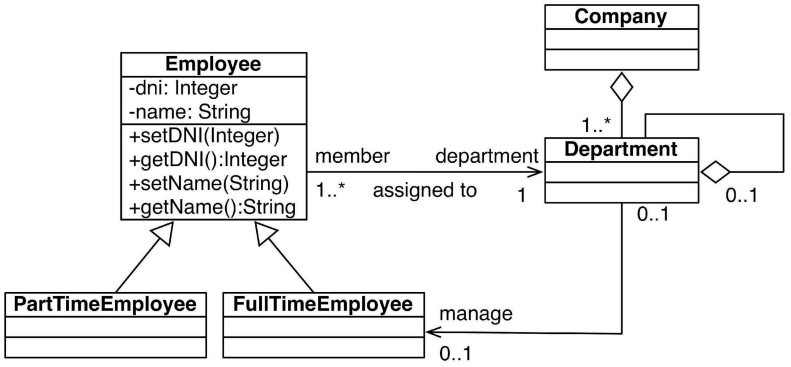
Para cada una de las siguientes afirmaciones discuta e indique si esta es Verdadera o Falsa.

	Verdadera	Falsa
Una clase es un descriptor de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.		
Una clase representa un concepto físico del domino de aplicación de un sistema que se esta modelando.		
Un diagrama de clases enfoca una perspectiva comportamental al modelar clases, sus atributos, sus operaciones, y relaciones entre clases.		

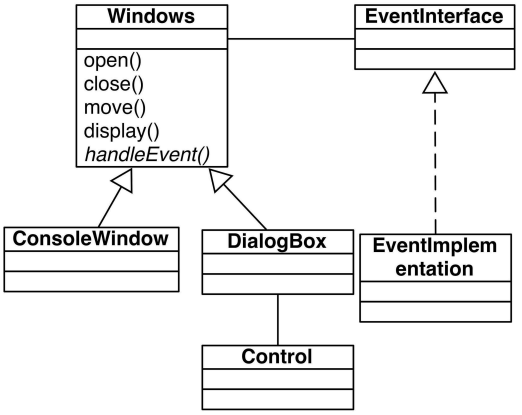
Ejercicio 2

- **Nombre cada uno de los elementos de notación o sintaxis que están presentes en los siguientes diagramas de clases.**
- **Describa brevemente qué interpreta en cada uno de dichos diagramas.**

a)



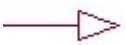
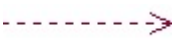
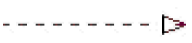
b)



Ejercicio 3

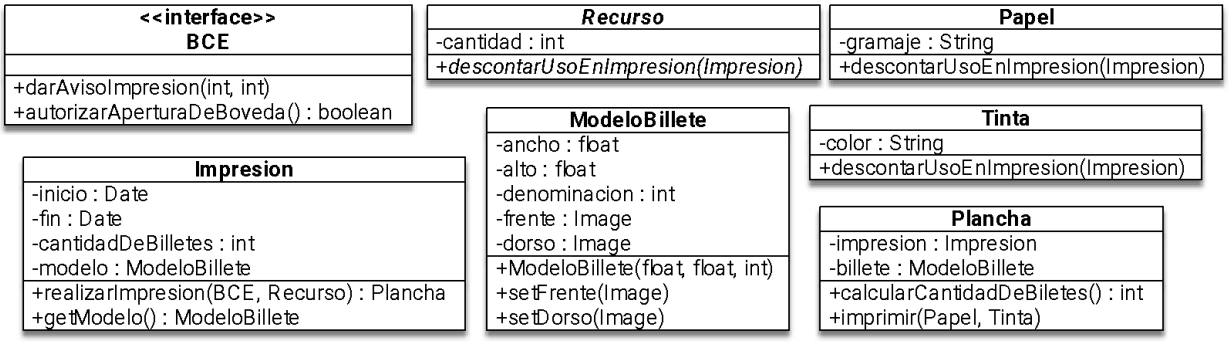
Considere las siguientes listas de términos. La de la izquierda, lista nombres de tipos de relaciones que se pueden dar entre clases. Mientras que la del centro, lista términos utilizados frecuentemente cuando se leen relaciones entre clases a partir de un diagrama. A la derecha se ven las representaciones en los diagramas.

- **Trace la respectiva correspondencia que existe entre los ítems de las listas.**

Especialización / Generalización	"USA"	
Dependencia	"ES PARTE DE"	
Realización	"ES UNA CATEGORÍA DE" … "ES UN"	
Agregación	"IMPLEMENTA"	
Asociación	"CONOCE A"	

Ejercicio 4 Considere el Enunciado 1 del TP Base:

Determine las relaciones faltantes en el diagrama de clases teniendo en cuenta el diagrama dado y las siguientes consideraciones. El método darAvisoImpresion tiene por parámetros dos valores de tipo int que denotan la denominación y cantidad de billetes impresos. Cada impresión se realiza para un solo modelo de billete. Cada plancha de impresión está formada por un único modelo de billete y pertenece a una única impresión. Sin embargo, en cada impresión pueden confeccionarse diferentes planchas. Indique los adornos cuando corresponda.



Ejercicio 5 (Caso de Estudio: Enunciado 3 TP Base)

Dado el siguiente código fuente parcial de la aplicación descrita en la narrativa realice el diagrama de clases correspondiente. Solo tenga en cuenta las clases y métodos indicados en el código fuente. Incluya los nombres de parámetros. Incluya adornos cuando corresponda. No incluya relaciones que no puedan ser deducidas del código fuente.

```
public class AdministradorDeReservas {

        public void crearReserva(String nombreResponsable, String apellidoResponsable, int dniResponsable,
Date fecha, int nroTarjeta, Date vencimientTarjeta, int codSeguridad, Cancha canchaReservada, float total){
                DatosTarjeta tarjeta=new DatosTarjeta(nroTarjeta, vencimientTarjeta, codSeguridad);
                canchaReservada.addReserva(new Reserva(nombreResponsable, apellidoResponsable,
dniResponsable, fecha, tarjeta, canchaReservada, total));
        }

        public void cancelarReserva(Date date, Cancha cancha){
                Reserva reserva=cancha.removeReserva(date);
                reserva.realizarPago(10);
        }
        public void informarNoPresentacion(Date date, Cancha cancha){
                cancha.getReserva(date).realizarPago(50);
        }
}
```

```
public class Cancha {
        private String deporte;
        private Object ubicacion;
        private List<Date> diasApertura;
        private Time horarioApertura;
        private Time horarioCierre;
        private Time duracionTurno;
        private List<Reserva> reservas;
        private long id;

        public void addReserva(Reserva r){
                reservas.add(r);
        }
        private List<Reserva> getReservaProxDias(int nroDias){
                //Do something
        }
        private Object listarCaracteristicas(){
                //Do something
        }
        public Reserva getReserva(Date date){
                for (Iterator<Reserva> iterator = reservas.iterator(); iterator.hasNext();) {
                        Reserva reserva = (Reserva) iterator.next();
                        if(reserva.getFecha().equals(date))
```

```
                return reserva;
            }
            return null;
        }
        public Reserva removeReserva(Date date){
            Reserva ret=getReserva(date);
            reservas.remove(ret);
            return ret;
        }
}
```

```
public class Reserva {
    private String nombreResponsable;
    private String apellidoResponsable;
    private int dniResponsable;
    private Date fecha;
    private DatosTarjeta tarjeta;
    private Cancha canchaReservada;
    private float total;
    private boolean pagado;

    public Reserva(String nombreResponsable, String apellidoResponsable, int dniResponsable, Date fecha,
            DatosTarjeta tarjeta, Cancha canchaReservada, float total) {
        this.nombreResponsable = nombreResponsable;
        this.apellidoResponsable = apellidoResponsable;
        this.dniResponsable = dniResponsable;
        this.fecha = fecha;
        this.tarjeta = tarjeta;
        this.canchaReservada = canchaReservada;
        this.total = total;
        pagado=false;
    }

    public void realizarPago(float porcentaje){
        float montoFinal=total*(porcentaje/100);
        getSistemaBancario().hacerPago(tarjeta.getNro(), tarjeta.getVencimiento(),
tarjeta.getCodSeguridad(), montoFinal);
        pagado=true;
    }

    private SistemaBancario getSistemaBancario(){
        //Do something
    }
    public Date getFecha() {
        return fecha;
    }
}
```

```
public class DatosTarjeta {

    private int nro;
    private Date vencimiento;
    private int codSeguridad;

    public DatosTarjeta(int nro, Date vencimiento,
int codSeguridad) {
        this.nro = nro;
        this.vencimiento = vencimiento;
        this.codSeguridad = codSeguridad;
    }
```

```
public interface SistemaBancario {
    public boolean hacerPago(int nroTarjeta, Date
vencimiento, int codSeguridad, float monto);
}
```

```
    public int getCodSeguridad() {
        return codSeguridad;
    }
    public int getNro() {
        return nro;
    }
    public Date getVencimiento() {
        return vencimiento;
    }
}
```

Ejercicio 6 (Ingeniería Reversa – C++)

Dado el siguiente código fuente escrito en C++:

- **Identifique las clases involucradas y construya el diagrama de clases con sus relaciones, atributos y operaciones.**
- **Si fuera posible, identifique paquetes e interfaces.**

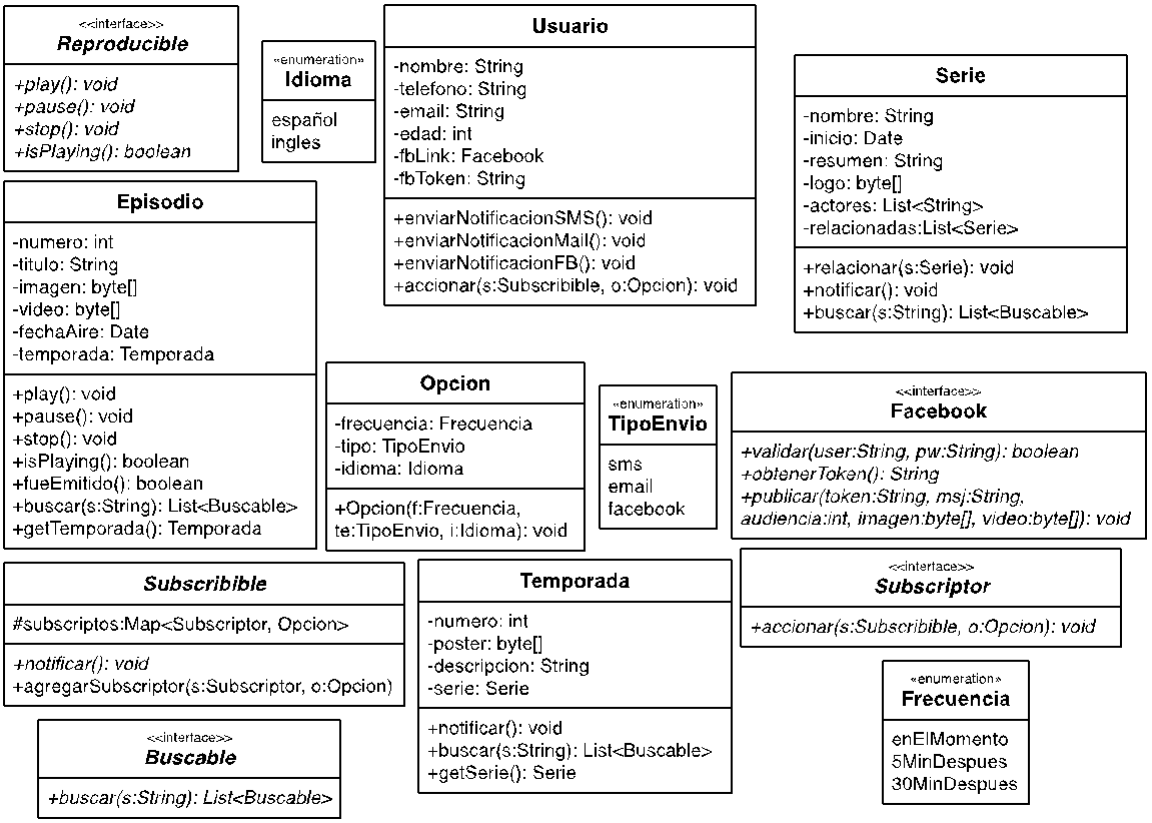
```
#include <iostream.h>
```

```
class Object { public:
```

```
    static int nombre_objetos = 0;
    int numero;
    Object();
    ~Object();
};

class Puerta : public Object { public:
    Puerta ();
    ~ Puerta ();
};

class Edificio : public Object { public:
    Puerta * puerta;
    int superficie;
    Edificio(int);
    ~Edificio();
};

class Torre : public Object { public:
    int altura;
    Muralla * muralla;
    Torre(int, Muralla *);
    ~Torre ();
};

class TorrePrincipal : public Torre { public:
    Castillo * castillo;
    TorrePrincipal (Castillo *, int);
    ~ TorrePrincipal ();
};

class Muralla : public Object { public:
    int longitud;
    Torre * torre1;
    Torre * torre2;
    Castillo * castillo;
    Muralla(int, int, int, Castillo *);
    ~ Muralla ();
};

class Castillo : public Edificio { public:
    TorrePrincipal * torrePrincipal;
    Muralla * muralla;
    Castillo (int, int, int, int, int);
    ~Castillo ();
};

int Object::nombre_objetos = 0 ;

Object::Object() {
    cout << "++ Object inicio" << endl;
    nombre_objetos++;
    numero = nombre_objetos ;
    cout << "++ Object fin" << nombre_objetos <<
endl;
}
Object::~Object() {
    cout << "-- Object inicio " << endl;
    nombre_objetos--;
    cout << "-- Object fin " << endl;
}

Puerta:: Puerta () {
    cout << "++ Puerta inicio" << endl;
    cout << "++ Puerta fin" << endl;
}

Puerta::~ Puerta () {
    cout << "-- Puerta inicio" << endl;
    cout << "-- Puerta fin" << endl;
}

Edificio:: Edificio (int s) {
    cout << "++ Edificio inicio " << endl;
    puerta = new Puerta();
    superficie = s;
    cout << "++ Edificio fin" << supercicie << endl;
}
Edificio::~ Edificio () {
    cout << "-- Edificio inicio " << endl;
    delete puerta;
    cout << "-- Edificio fin" << endl;
}

Torre:: Torre (int h, Muralla * m) {
    cout << "++ Torre inicio " << endl;
    altura = h;
    muralla = m;
    cout << "++ Torre fin " << hauteur << endl;
}

Torre::~ Torre () {
    cout << "-- Torre inicio " << endl;
    cout << "-- Torre fin" << endl;
}

TorrePrincipal:: TorrePrincipal (Castillo * c, int h) :
Torre(h, NULL) {
    cout << "++ TorrePrincipal inicio " << endl;
    castillo = c;
    cout << "++ TorrePrincipal fin" << endl;
}
TorrePrincipal::~ TorrePrincipal () {
    cout << "-- TorrePrincipal inicio " << endl;
    cout << "-- TorrePrincipal fin" << endl;
}

Muralla::Muralla(int l, int h1, int h2, Castillo * c) {
    cout << "++ Muralla inicio " << endl;
    longitud = l;
    torre1 = new Torre(h1, this);
    torre2 = new Torre(h2, this);
    castillo = c;
    cout << "++ Muralla fin " << longitud << endl;
}
Muralla::~Muralla() {
    cout << "-- Muralla inicio " << endl;
    delete torre1;
    delete torre2;
    cout << "—Muralla fin" << endl;
}
Castillo:: Castillo (int l, int h1, int h2, int hd, int s) :
Edificio
    cout << "++ Castillo inicio " << endl;
    muralla= new Muralla(l, h1, h2, this);
    torrePrincipal(this, hd);
    cout << "++ Castillo fin" << endl;
}
Castillo::~Castillo () {
    cout << "-- Castillo inicio" << endl;
    delete muralla;
    delete torrePrincipal;
    cout << "-- Chateau fin" << endl;
}
main() {
    Castillo castillo (400, 40, 60, 80, 10000);
}
```

Ejercicio 7

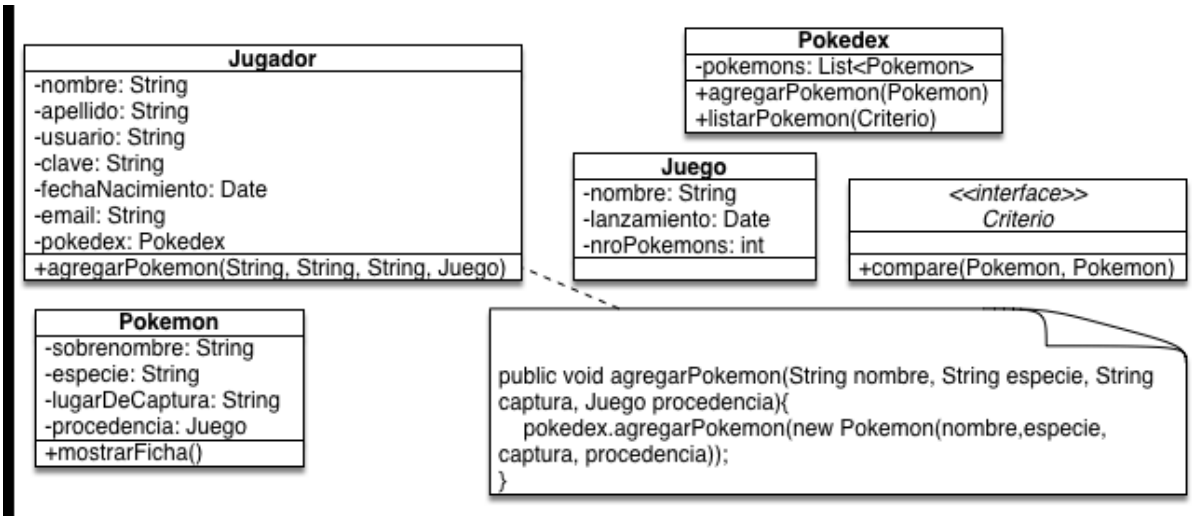
Dado el siguiente conjunto de clases, determine las relaciones entre ellas (asociación, agregación, composición, generalización y realización) a partir de sus atributos y operaciones. Tenga en cuenta las siguientes restricciones. Las series pueden estar formadas por como máximo 10 temporadas. Las temporadas además están formadas solamente por 6, 10, 13 o 22 episodios. También considere que los datos de la subscripción (clase Opcion) emergen como propiedad (es una clase de asociación) de la subscripción entre un Subscriptor y un elemento Subscribible.



Ejercicio 8

En el conjunto de juegos de Pokemon el conjunto de pokemons con los que cuenta un usuario se denomina pokédex. En este sentido, la aplicación deberá presentar la pokédex listando todos los pokémons que la componen ordenada por diferentes criterios (alfabética, número o tipo). Asimismo, en la pokédex se indicará el porcentaje de completitud de captura de los pokémons en cada uno de los juegos (dividiendo el total de pokémons por el número capturado por el jugador). Por cada pokémon del pokédex, se mostrará una breve reseña del mismo, incluyendo su número, nombre de especie y el juego donde fue capturado. Asimismo, si el jugador lo desea, en este momento puede solicitar ver la ficha individual de un pokémon seleccionandolo en la pokédex.

A partir del siguiente diagrama de clases parcial y teniendo en cuenta la narrativa presentada, determine las relaciones faltantes en el diagrama de clases. Incluya los adornos correspondientes cuando corresponda. No incluya relaciones que no puedan ser deducidas del diagrama de clases.



Ejercicio 9

Dado el siguiente código fuente escrito en Java, identificar las clases involucradas, definiendo su nombre, atributos y métodos. Luego, construir el diagrama de clases indicando las relaciones y sus correspondientes adornos. Aclaración: no se deben tener en cuenta las clases que no se encuentran descriptas en código (List, Date, etc.). Tener en cuenta que las varitas no pueden pertenecer a más de un personaje a la vez y; que los cursos son creados sin tener ningún profesor a cargo y luego pueden asignarse múltiples profesores.

```
public abstract class Personaje {
        protected Varita varita;
        private boolean encantado;

        public abstract boolean
registrarIngreso();

        public void Personaje(){
                this.encantado = false;
                this.varita = null;
        }

        public void solicitarVarita(){

this.varita=Cuestionario.elegirVarita();
        }

        public void hechizar(Personaje
objetivo){
                if (!this.encantado)
                  varita.encantar(objetivo);
        }

        public void setEncantado(boolean b) {
                this.encantado = b;
        }

}
```

```
public class Curso {
        private boolean abierto;

        public void Curso(){
                this.abierto = true;
        }

          public void modificarEstado(boolean
b) {
                this.abierto = b;
        }
```

```
public class Estudiante extends Personaje{
        private Hashtable<Curso,Integer> cursos;
        private Estudiante compañeroHabitacion;
        private Casa casa;
        private Ala vivienda;
        private Date proximaReunion;
        private Date ultimoIngreso;

          public void Estudiante(Estudiante
compañero){
            super.Personaje();
              this.cursos = new
Hashtable<Curso,Integer>();
            this.compañeroHabitacion = compañero;
            this.casa = null;
            this.vivienda = null;
            this.proximaReunion = null;
        }

        public void elegirCasa(){
              casa = Cuestionario.responder(this);
        }

        public boolean registrarIngreso(){
                this.ultimoIngreso = new Date();
                  this.varita.setActiva(true);
          }

        public void reunir(Date fecha) {
                this.proximaReunion = fecha;
        }

        public void inscribirCurso(Curso c){
                this.cursos.put(c, null);
        }

        public void calificar(Curso c, int nota) {
                this.cursos.put(c, nota);

        }
```

```
}
```

```
public class Casa {
        private List<Estudiante> estudiantes;

        public void Casa(){
       this.estudiantes = new
ArrayList<Estudiante>();
        }

        public void reunir(Date fecha){
            for(int i=0; i < estudiantes.size();
i++){
                estudiantes.get(i).reunir(fecha);
                }
        }
}
```

```
public class Profesor extends Personaje{
        private Curso c;

        public void Profesor(Curso c){
                super.Personaje();
                this.c = c;
        }

        public boolean registrarIngreso() {
                c.modificarEstado(false);
        }

        public boolean terminarClase(){
                c.modificarEstado(true);
        }

        public void calificar(Estudiante
alumno){
                int nota = (Math.random()*10);
                alumno.calificar(c, nota);
        }
}
```

```
}
```

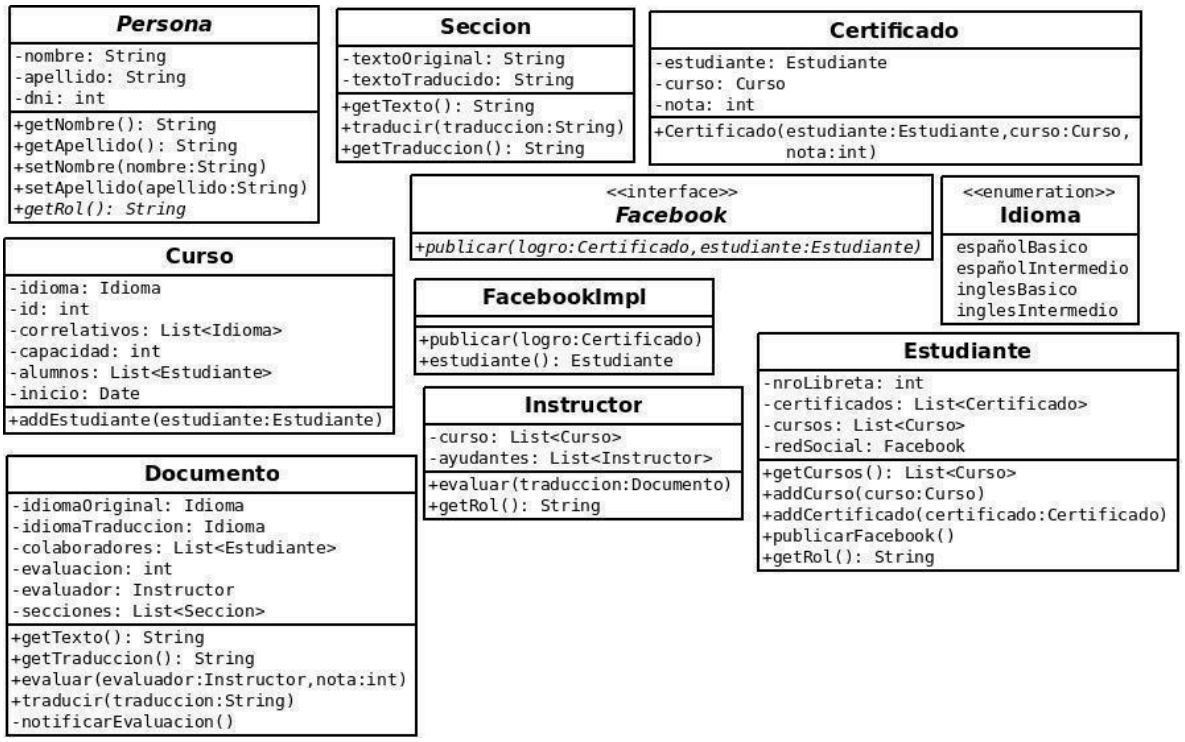
```
public class Varita {
       public void encantar(Personaje objetivo) {
            objetivo.setEncantado(true);
       }
}
```

```
public class Ala {
        private int nroHabitaciones;
        public void Ala (int habitaciones){
             this.nroHabitaciones = habitaciones;
        }

        public void destroy() {
                this.finalize();
        }

}
```

```
public class Edificio {
        private List<Ala> alas;

        public void Edificio(Ala alaOriginal){
                alas = new ArrayList<Ala>();
                alas.add(alaOriginal);
        }

        public void addAla(Ala a){
                alas.add(a);
        }

        public void destroy(){
                for (int i=0; i<alas.size(); i++){
                        alas.get(i).destroy();
                }
                this.finalize();
        }
}
```

Ejercicio 10 (Organización de cursos)

Dado el siguiente conjunto de clases, determine las relaciones entre ellas (dependencia, asociación, agregación, composición, generalización y realización) a partir de sus atributos y operaciones. Tenga en cuenta las siguientes restricciones. Un instructor puede ser ayudante de cero o varios instructores. A su vez, un instructor puede tener a cargo ningún o varios ayudantes de cátedra. Cada instructor puede evaluar cero o múltiples documentos, asignándose el evaluador del documento en cualquier momento (no necesariamente en la creación del documento). Cuando se crea un curso se le asigna a este uno o más instructores pero para ser instructor no es necesario tener un curso a cargo (pudiendo tener de cero a múltiples cursos). Los estudiantes pueden cursar hasta un máximo de 5 cursos simultáneamente. Una única instancia de Facebook se encarga de realizar las publicaciones de los múltiples estudiantes. Un documento está formado por sus correspondientes secciones.



Ejercicio 11 Dado el siguiente código fuente parcial de la aplicación descrita en el Enunciado 11 del TP Base realice el diagrama de clases correspondiente. Solo tenga en cuenta las clases, métodos y atributos indicados en el código fuente. No es necesario incluir los nombres de parámetros. Incluya adornos cuando corresponda. No incluya relaciones que no puedan ser deducidas del código fuente o de la narrativa.

```
public class RegistroAtencion {
  protected Date fechaAtencion;
  private String descripcion;

  public RegistroAtencion(Date
      fechaAtencion, String descripcion) {
    this.fechaAtencion = fechaAtencion;
    this.descripcion = descripcion;
  }
  public Date getFechaAtencion() {
    return fechaAtencion;
  }
  public String getDescripcion() {
    return descripcion;
  }
}
```

```
public class AtencionAmbulatoria extends
                    RegistroAtencion {
  private String dependencia;

  public AtencionAmbulatoria(Date fechaAtencion,
String descripcion, String dependencia) {
    super(fechaAtencion, descripcion);
    this.dependencia = dependencia;
  }

  public String getDependencia() {
    return dependencia;
  }
}
```

```
public class Internacion extends RegistroAtencion {
  private Date fechaAlta;
  private String responsableInternacion;


  public Internacion(Date fechaAtencion, String
descripcion, String responsableInternacion) {
    super(fechaAtencion, descripcion);
    this.responsableInternacion =
                responsableInternacion;
    this.fechaAlta=null;
  }
  public void setFechaAlta(Date fechaAlta)
  {
    this.fechaAlta = fechaAlta;
  }
  public long calcularDiasInternacion() {
    if(fechaAlta!=null)
      return fechaAlta.getTime()-
      fechaAtencion.getTime();
    return -1;
```

```
public class DatosPaciente {
  private String nombre;
  private String apellido;
  private String direccion;
  private Date fechaDeNacimiento;

  public DatosPaciente(String nombre, String apellido,
String direccion, Date fechaDeNacimiento) {
    this.nombre = nombre;
    this.apellido = apellido;
    this.direccion = direccion;
    this.fechaDeNacimiento =
                fechaDeNacimiento;
  }
  public String getNombre(){return nombre;}
  public String getApellido() {
    return apellido;
  }
  public String getDireccion() {
    return direccion;
  }
```

```
  }
}
```

```
  public Date getFechaDeNacimiento() {
    return fechaDeNacimiento;
  }
}
```

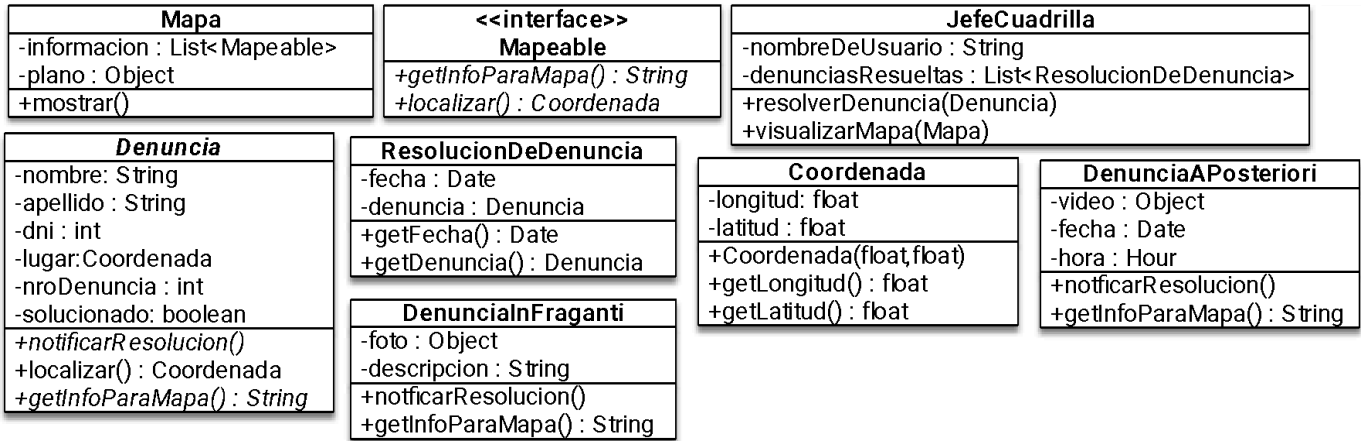
```
public class HistoriaClinica {
  private DatosPaciente datosPaciente;
  private List<RegistroAtencion>
                        atenciones;

  public HistoriaClinica(DatosPaciente datosPaciente,
RegistroAtencion primeraAtencion) {
    this.datosPaciente=datosPaciente;
    atenciones=new
          ArrayList<RegistroAtencion>();
    atenciones.add(primeraAtencion);
  }

  public void addAtencion(RegistroAtencion atencion) {
    atenciones.add(atencion);
  }
}
```

```
public class Cama {
  public static int GENERAL=1;
  public static int TERAPIA_INTERMEDIA=2;
  public static int TERAPIA_INTENSIVA=3;
  private int tipoCama;
  private boolean disponible;
  private DatosPaciente ocupanteActual;

  public Cama(int tipoCama) {
    this.tipoCama = tipoCama;
    disponible=true;
    ocupanteActual=null;
  }
  public boolean isDisponible() {
    return disponible;
  }
  public int getTipoCama() {
    return tipoCama;
  }
  public void setOcupanteActual(
          DatosPaciente ocupanteActual) {
    this.ocupanteActual = ocupanteActual;
  }
  public DatosPaciente getOcupanteActual(){
    return ocupanteActual;
  }
}
```

```
public class AdministradorDeCamas {
        private List<Cama> camas;

        public AdministradorDeCamas() {
                camas=new ArrayList<Cama>();
        }

        public void cargarNuevaCama(int tipoCama) {
                camas.add(new Cama(tipoCama));
        }

        public int camasDisponibles() {
                int acum=0;
                for (Cama cama : camas) {
                        if(cama.isDisponible())
                                acum++;
                }
                return acum;
        }

        public int capacidadCamasUTI() {
                int acum=0;
                for (Cama cama : camas) {
                        if(cama.getTipoCama()==Cama.TERAPIA_INTENSIVA)
                                acum++;
                }
                return acum;
        }
        public List<DatosPaciente> listarOcupantesCamas(){
                List<DatosPaciente> ret=new ArrayList<DatosPaciente>();
                for (Cama cama : camas) {
                        if(!cama.isDisponible())
                                ret.add(cama.getOcupanteActual());
                }
                return ret;
        }
}
```

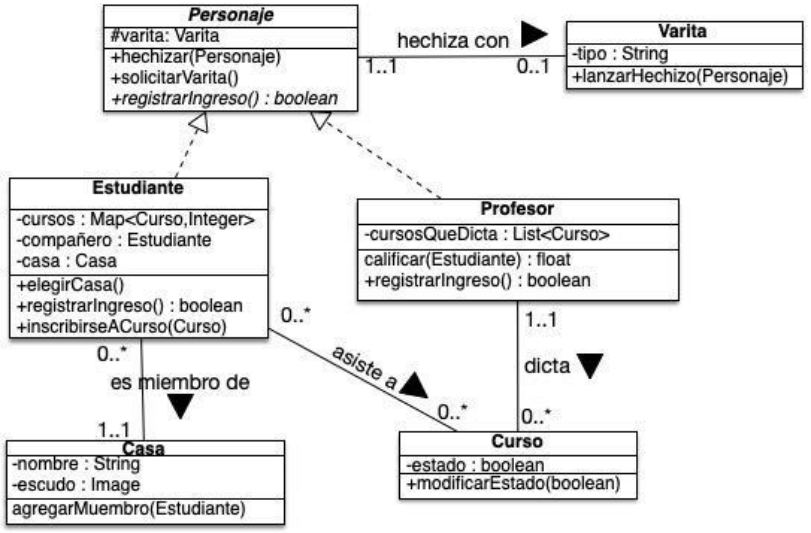
Ejercicio 12 (Enunciado 5 TP Base) Dado el siguiente conjunto de clases, determine las relaciones entre ellas a partir de sus atributos y operaciones. Tenga en cuenta las siguientes restricciones. Cuando se elimina una

denuncia también se eliminan la “coordenada” y “resolución de denuncia” relacionadas a la misma. Un mapa está compuesto por un conjunto de cero o más información “mapeable”.



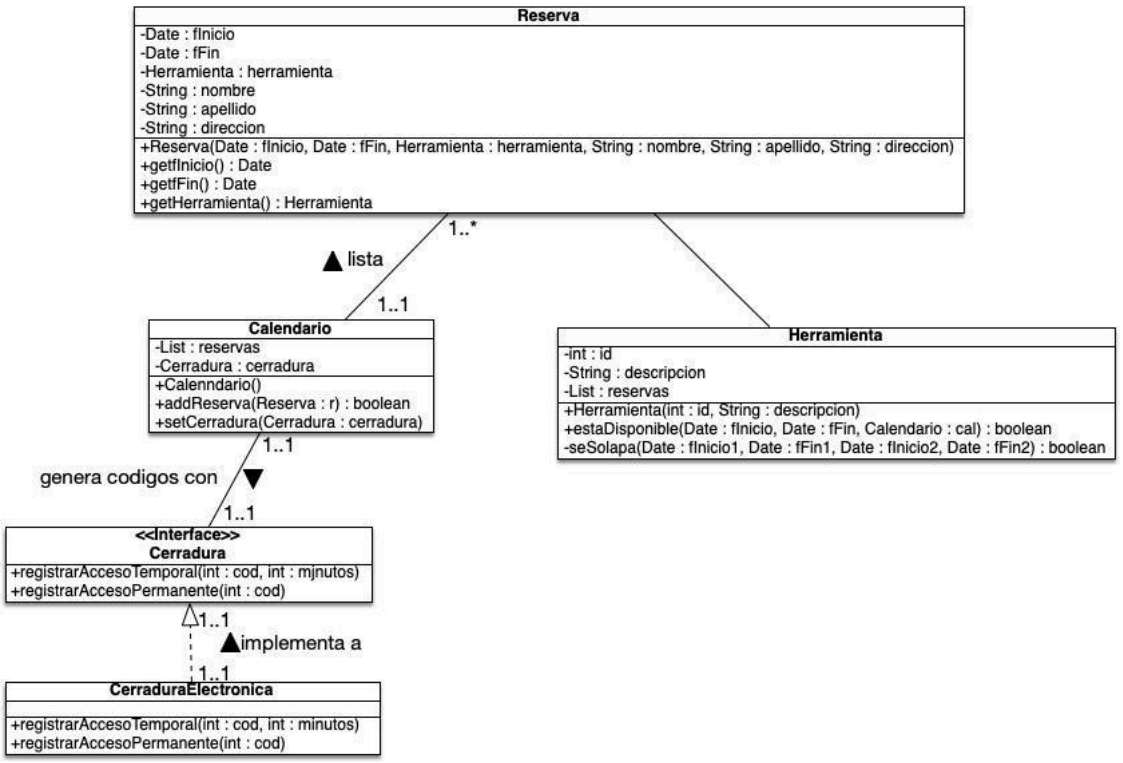
Ejercicio 13. A partir del siguiente diagrama, indique los errores. Solo tenga en cuenta el diagrama y la siguiente información:

- Se esta desarrollando un juego de Harry Potter.
- Los personajes del juego son magos que pueden lanzar hechizos mediante varitas mágicas.
- Hay 2 tipos de personaje: estudiantes y profesores.
- Cada estudiante puede tomar la cantidad de cursos que desee y tiene un compañero de cuarto con el que vive.
- Ademas, todos los estudiantes son miembros de una casa. Cada casa puede estar compuesta por un numero ilimitado de estudiantes.
- Los profesores dictan de 3 a 4 cursos pero cada curso es dictado por un único profesor.



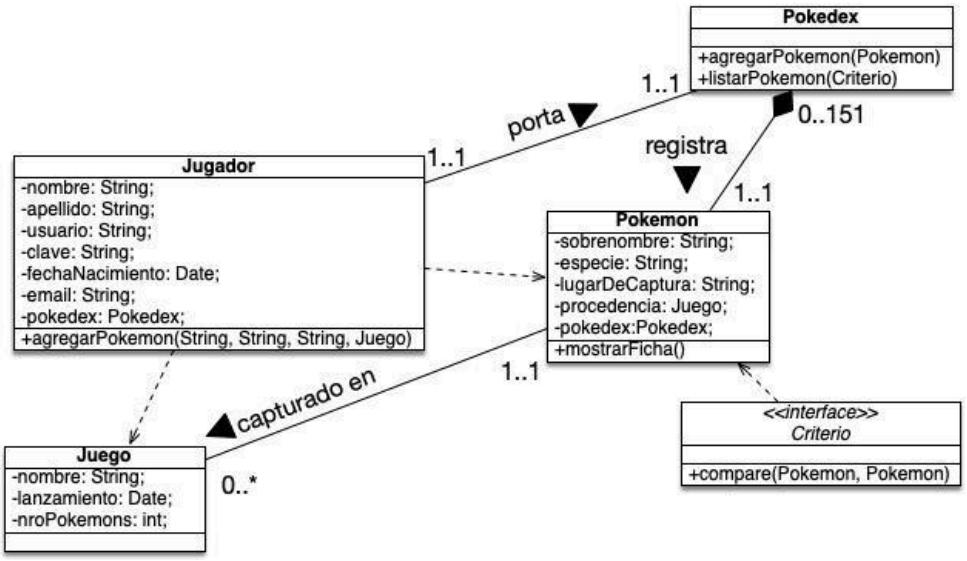
Ejercicio 14 A partir del siguiente diagrama, indique los errores. Solo tenga en cuenta el diagrama y la siguiente información:

- La aplicación permite automatizar reservas de diferentes herramientas mediante una cerradura electrónica que permite acceder a un deposito en el cual se encuentran todas las herramientas.
- La aplicación utiliza la API de la cerradura para generar un código de acceso 5 minutos antes de una reserva.
- El calendario se compone por un número ilimitado de reservas
- Una reserva corresponde a una única herramienta



Ejercicio 15 A partir del siguiente diagrama, indique los errores. Solo tenga en cuenta el diagrama y la siguiente información:

- Se esta desarrollando un sistema que permite llevar un registro de los distintos pokemons capturados por el usuario en diferentes videojuegos de la saga.
- La pokédex es una enciclopedia portátil que se compone de todos los pokemons capturados por un jugador.
- La pokédex puede registrar un máximo de 151 pokemons.



Bibliografía de Apoyo Sugerida

[1] – The Unified Modeling Language – Reference Manual. J. Rumbaugh, I. Jacobson and G. Booch. Addison Wesley Longman, Inc. 1999. ISBN 0-201-30998-X. Código de consulta en Biblioteca Central UNICEN: 001.642 R936-1.

- Resumen de la notación de UML: Appendix B, páginas 519 a 530.
- Clases, Relaciones y Diagramas de Clases: Part 2 - Chapter 4 (Static View), páginas 41 a 59.

- Diagramas de Objetos: Part 2 - Chapter 4 (Static View), páginas 59 a 61.

[2] – The Unified Modeling Language – User Guide. G. Booch, J. Rumbaugh and I. Jacobson. Addison Wesley Longman, Inc. 1999. ISBN 0-201-57168-4. Código de consulta en Biblioteca Central UNICEN: 001.642 B724-3.

- Conceptos generales sobre diagramas de UML: Section 2 – Chapter 7, páginas 91 a 104.
- Resumen de la notación de UML: Appendix A, páginas 435 a 437.
- Clases: Chapter 4 y 9.
- Relaciones: Chapter 5 y 10.
- Diagramas de Clases: Chapter 8.
- Diagramas de Objetos: Chapter 14.
- Paquetes e Intefaces: Chapter 11 y 12.
- Instancias: Chapter 13.