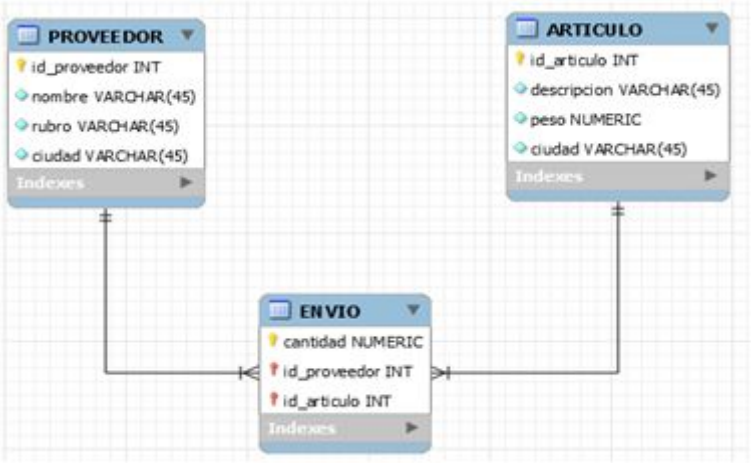
Ejercicios de Repaso para la Certificación

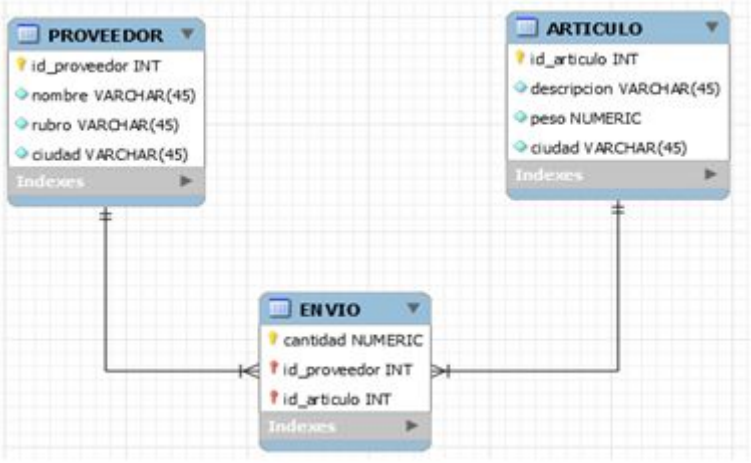
Ejercicios de Bases de Datos

1. **Dado el siguiente de Diagrama de Entidad-Relación y la consulta SQL. Modifique la consulta SQL ya que contiene errores y describa el resultado de la consulta.**



SELECT id_proveedor, id_articulo
FROM Proveedor p JOIN Envio e ON p.nombre = e.id_proveedor

2. **Dado el siguiente de Diagrama de Entidad-Relación y la consulta SQL. Modifique la consulta SQL ya que contiene errores y describa el resultado de la consulta.**



```
SELECT SUM (e.cantidad), e.id_proveedor, e.id_articulo
FROM Proveedor p JOIN Envio e ON p.id_proveedor = e.id_proveedor
AND e.id_articulo IN (SELECT a.descripcion FROM Articulo a)
```

3. **Dada la siguiente consulta SQL y las siguientes tuplas de ejemplo, determinar el resultado de la consulta:**

```
SELECT SUM (e.cantidad), e.id_proveedor, e.id_articulo
FROM Proveedor p JOIN Envio e ON p.id_proveedor = e.id_proveedor
AND e.id_articulo IN (SELECT a.id_articulo FROM Articulo a)
HAVING e.cantidad > 100
GROUP BY e.id_proveedor
ORDER BY e.ciudad DESC
```

Tuplas de ejemplo:

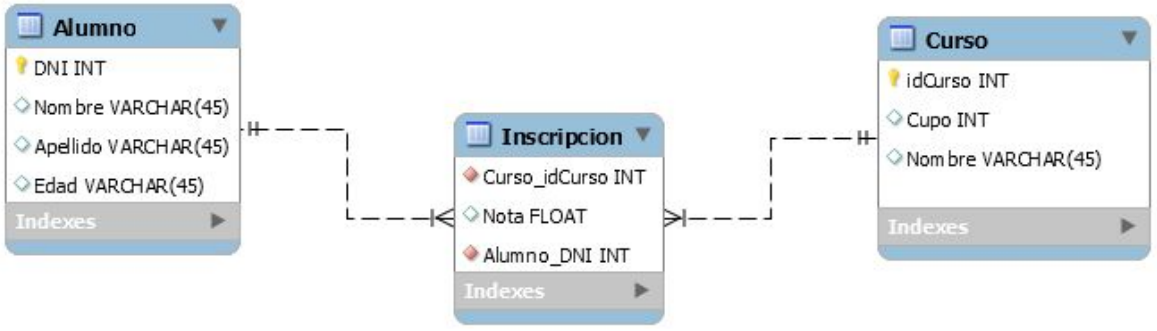
ARTICULO:
1,”articulo A”, 30, “Tandil”
2,”articulo B”, 50, “Balcarce”
3,”articulo C”, 10, “Olavarría”
4,”articulo D”, 60, “Pinamar”

PROVEEDOR:
1, “producto X”, “limpieza”, “Tandil”
2, “producto Y”, “higiene”, “Azul”
3, “producto Z”, “farmacia”, “Bolivar”
4, “producto W”, “limpieza”, “Mar del Plata”

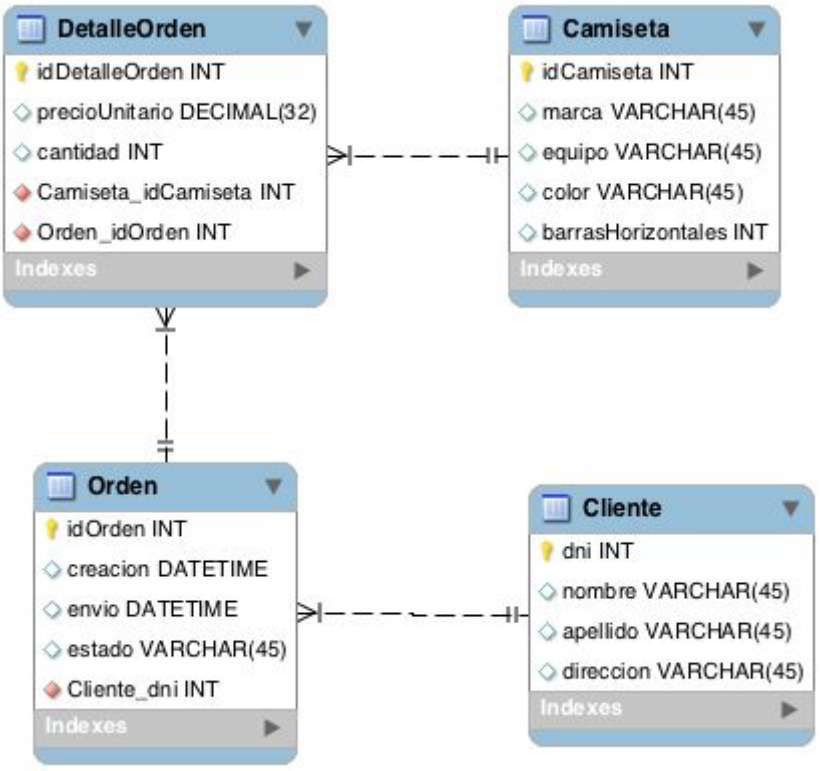
ENVIO:
1,1,120
1,2,160
3,1,90
4,3,100

4. **El equipo administrativo que trabaja con el director de un colegio se encuentra en este momento contabilizando los estudiantes que al menos aprobaron 1 curso de los que ofrece el Programa 111Mil en el Colegio Secundario Nº1. En su base de datos existe información de los alumnos, de los cursos y de las inscripciones a dichos cursos. A partir del DER del ejercicio anterior, el equipo necesita listar el nombre y apellido de cada alumno, con su DNI, edad y el nombre del o los curso/s con nota 7 ó superior. El listado debe estar ordenado alfabéticamente por el apellido del alumno. El equipo necesita que escribas la consulta SQL correspondiente.**

.



5. **Dado el siguiente diagrama de entidad-relación, escriba una consulta SQL que liste los números de orden (id) de todas las órdenes pertenecientes a clientes con apellido Rodriguez.**



6. **Dada la siguiente porción de clase en Java y la sentencia de DDL de creación de la tabla escriba la línea del archivo de mapeo de Hibernate HBM en formato XML correspondiente al mapeo del atributo "monto".**

Infraccion.java

```
public class Infraccion {

   private Integer id;
   private InfraccionNomenclada infraccionNomenclada;
   private Integer cantidadPuntosDescontados;
   private BigDecimal monto;
   private String observacion;
```

Script SQL de creación de la tabla

```
CREATE TABLE `Infraccion` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `cantidadPuntosDescontados` tinyint(3) unsigned DEFAULT NULL,
  `monto` decimal(8,2) unsigned NOT NULL,
  `observacion` varchar(255) DEFAULT NULL,
  `numeroActaConstatacion` int(10) unsigned NOT NULL,
  `codigoInfraccionNomenclada` int(10) unsigned NOT NULL
  PRIMARY KEY (`id`),
  KEY `fk_Infraccion_ActaConstatacion1_idx` (`numeroActaConstatacion`),
  KEY `fk_Infraccion_InfraccionNomenclada1_idx` (`codigoInfraccionNomenclada`),
  CONSTRAINT `fk_Infraccion_ActaConstatacion1`
    FOREIGN KEY (`numeroActaConstatacion`) REFERENCES `ActaConstatacion` (`numero`)
    ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_Infraccion_InfraccionNomenclada1`
        FOREIGN KEY (`codigoInfraccionNomenclada`) REFERENCES `InfraccionNomenclada`
(`codigo`)
    ON DELETE NO ACTION ON UPDATE NO ACTION
)
```

7. **Dada la siguiente porción de clase en Java y la sentencia de DDL de creación de la tabla escriba la línea del archivo de mapeo de Hibernate HBM en formato XML correspondiente al mapeo del atributo "infraccionNomenclada".**

Infraccion.java

```
public class Infraccion {

   private Integer id;
   private InfraccionNomenclada infraccionNomenclada;
   private Integer cantidadPuntosDescontados;
   private BigDecimal monto;
   private String observacion;
```

Script SQL de creación de la tabla

```
CREATE TABLE `Infraccion` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `cantidadPuntosDescontados` tinyint(3) unsigned DEFAULT NULL,
  `monto` decimal(8,2) unsigned NOT NULL,
  `observacion` varchar(255) DEFAULT NULL,
  `numeroActaConstatacion` int(10) unsigned NOT NULL,
  `codigoInfraccionNomenclada` int(10) unsigned NOT NULL
  PRIMARY KEY (`id`),
  KEY `fk_Infraccion_ActaConstatacion1_idx` (`numeroActaConstatacion`),
  KEY `fk_Infraccion_InfraccionNomenclada1_idx` (`codigoInfraccionNomenclada`),
  CONSTRAINT `fk_Infraccion_ActaConstatacion1`
    FOREIGN KEY (`numeroActaConstatacion`) REFERENCES `ActaConstatacion` (`numero`)
    ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `fk_Infraccion_InfraccionNomenclada1`
        FOREIGN KEY (`codigoInfraccionNomenclada`) REFERENCES `InfraccionNomenclada`
(`codigo`)
    ON DELETE NO ACTION ON UPDATE NO ACTION
)
```

Ejercicios de Java

1. **Considere el siguiente método**

```
public void ifElseMisterioso(int x, int y) {
   int z = 4;
   if (z <= x) {
       z = x + 1;
   } else {
       z = z + 9;
   }
   if (z <= y) {
       y++;
   }
   System.out.println(z + " " + y);
}
```

Para cada una de las siguientes invocaciones, determinar la salida que se obtiene:

`ifElseMisterioso(3,20);`	
`ifElseMisterioso(4,5);`	
`ifElseMisterioso(5,5);`	
`ifElseMisterioso(6,10);`	

2. **Considere el siguiente método:**

```
public void misterio(int[] a, int[] b) {
   for (int i = 0; i < a.length; i++) {
       a[i] += b[b.length - 1 - i];
   }
}
```

Dados los siguientes arreglos:

```
int[] a1 = {1, 3, 5, 7, 9};
int[] a2 = {1, 4, 9, 16, 25};
```

Determine los valores de los elementos en el arreglo a1 luego de ejecutar la siguiente invocación al método: misterio(a1,a2)

3. **Determinar los valores almacenados en el arreglo array luego de que se ejecute el siguiente fragmento de código**

```
int [] array = {2,18,6,-4,5,1};
for(int i=0;i<array.length;i++)
    array[i] = array[i] + (array[i] / array[0]);
```

4. **El constructor de la clase Punto tiene dos problemas. Cuáles son? Encontrar y arreglar los problemas.**

```
public class Punto{
    int x;
    int y;

    public void Punto(int xInicial, int yInicial) {
        int x = xInicial;
        int y = yInicial;
    }

}
```

5. **Considere las siguientes invocaciones. Determine la salida que se produce.**

```
public class Raro{
   public static void main(String[] args) {
       Raro.primero();
       Raro.tercero();
       Raro.segundo();
       Raro.tercero();
   }

   public static void primero() {
       System.out.println("Dentro del método primero.");
   }

   public static void segundo() {
       System.out.println("Dentro del método segundo.");
       primero();
   }

   public static void tercero() {
       System.out.println("Dentro del método tercero.");
       primero();
       segundo();
   }
}
```

6. Considere el siguiente programa:

```
1.  public class Ejemplo{

2.      public static void mostrarReglas(){
3.          System.out.println("La primera regla ");
4.          System.out.println("del club de Java es");
5.          System.out.println("");
6.          System.out.println("no se habla del club de Java!");
7.      }

8.      public static void main(String[] args){
9.          System.out.println("Las reglas del club de Java.");
10.         Ejemplo.mostrarReglas();
11.         Ejemplo.mostrarReglas();
12.     }
13. }
```

Qué sucedería si se realizan los siguientes cambios a la clase Ejemplo? Considerar cada cambio de forma independiente de los otros.
Para cada cambio, considerar sólo tres posibilidades:

- "Nada": Si no ocurrirá ningún cambio en el programa.
- "Error". Si produciría que el programa no compile o que de un error durante su ejecución.
- "Salida". Si cambiaría la salida de la ejecución.

Cambiar la línea 1 por `public class Demostracion{`	
Cambiar la línea 8 por `public static void MAIN(String [] args){`	
Insertar una nueva línea debajo de la línea 10 que diga `Ejemplo.mostrarReglas();`	
Cambiar la línea 2 a `public static void imprimirMensaje(){`	
Cambiar la línea 2 a `public static void mostrarMensaje(){` y cambiar las líneas 10 y 11 a `Ejemplo.mostrarMensaje();`	
Reemplazar las líneas 3-4 con `System.out.println("La primera regla del club de Java es, ");`	

7. **Dada la siguiente clase:**

```
public class ListaDeProductos{
	private List<Productos> productos;
	public List<Produtos> getProductos(){
		return productos;
	}
}
```

Cree una ListaDeProductosFiltrada que retorne los productos que empiezan con una letra dada (para ello, puede heredar de la clase dada y extender su funcionalidad).

8. **Extienda el funcionamiento de la siguiente clase, agregando un chequeo por usuario (si el nombre de usuario que intenta abrir el correo no coincide con el nombre de usuario configurado, imprimir un error):**

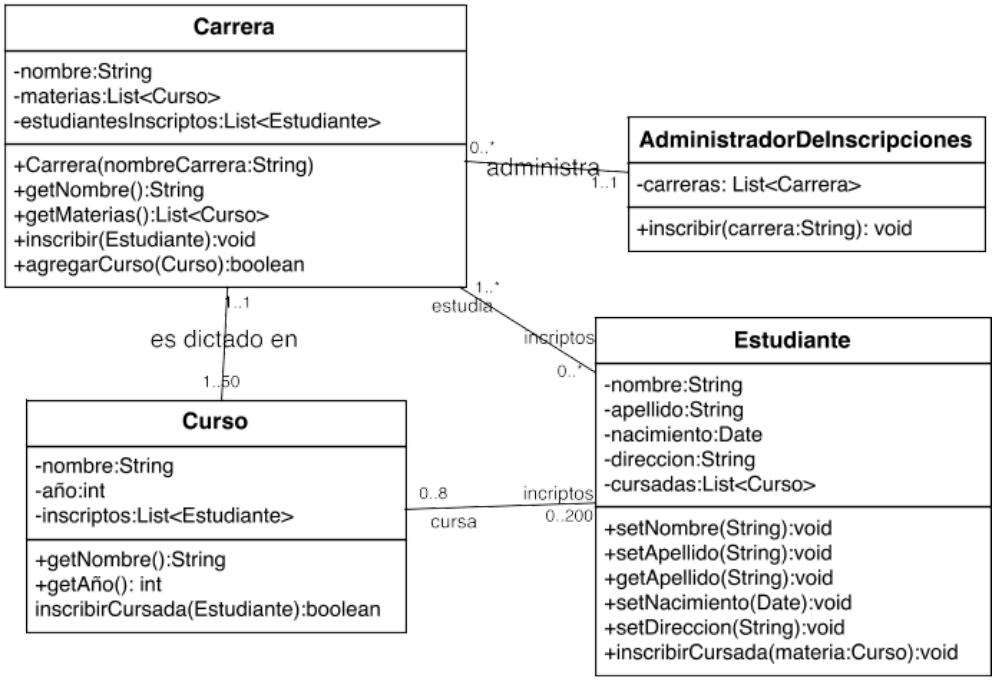
```
public class CorreoPublico {
	String contenido;
	public String leer(String nombreUsuario) {
		return contenido;
	}
}
```

9. **Realice las modificaciones necesarias sobre Noticia que permitan extenderla. El objetivo es crear una subclase llamada NoticiaResumida que solo muestra los primeros 50 caracteres del contenido más "..." (3 puntos suspensivos) que indican que el contenido continúa. Su título también es limitado a 20 caracteres, pero no se colocan puntos.**

```
public class Noticia{
	private String titulo;
	private String contenido;

	public String getTitulo() {
		return titulo;
	}

	public String getContenido() {
		return contenido;
	}

	// Retorna el contenido con la primer letra capital.
	protected String getContenidoConFormato() {
		return contenido.substring(0,1).toUpperCase()
			+ contenido.substring(1);
	}

	// Retorna el título todo en mayúsculas
	protected String getTituloConFormato() {
		return titulo.toUpperCase();
	}

	public String imprimir(){
		return getTituloConFormato() + "\n"
			+ getContenidoConFormato();
	}
}
```

10. Implemente las clases y los métodos Java que se describen en el diagrama de clases teniendo en cuenta los siguientes detalles:

- El método agregarCurso solo agregará el curso pasado por parámetros si se cumple la restricción indicada en la multiplicidad. En caso de poder agregarse satisfactoriamente devuelve true.
- Idem método inscribirCursada de la clase Curso.
- El método inscribirCursada de la clase Estudiante deberá invocar al método inscribirCursada de la clase Curso y sólo inscribirá el curso pasado por parámetros si el resultado del método invocado es true y si se cumple la restricción indicada en la multiplicidad.



10.1. Es necesario incorporar un método para obtener los cursos de todas las carreras de un año en particular dado.

a.) Indiqué en qué clase/s debería definirse esa nueva funcionalidad.
b.) Implemente el/los método/s necesarios en Java. Si considera necesario nuevas variables defínalas y diga en qué clase se implementan.

10.2. Otro desarrollador implementó el siguiente método en la clase Carrera. El método debería devolver todos los estudiantes cuyo apellido coincida por el pasado por parámetros. Sin embargo, no funciona. Encuentre el error.

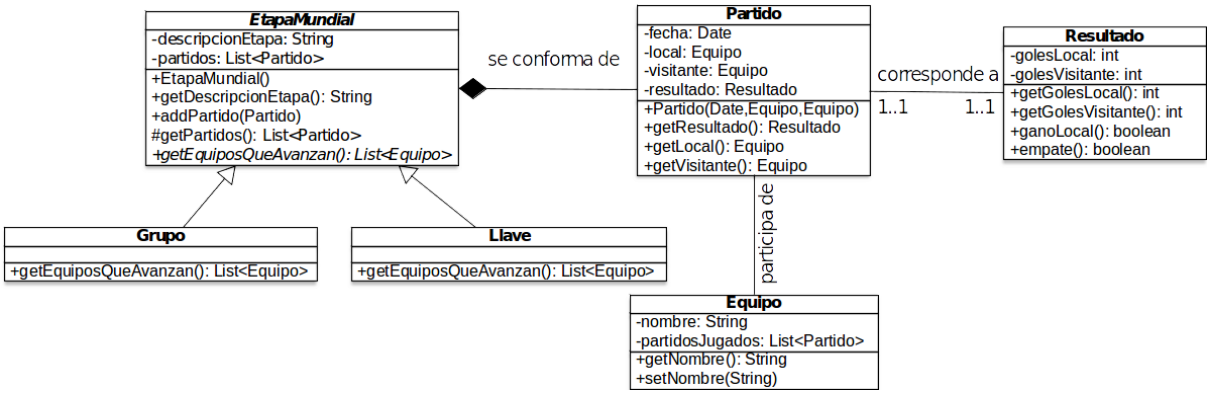
```
public List<Estudiante> getEstudiantesPorApellido(String apellido){
  List<Estudiante> estudiantesConApellido=new ArrayList<>();
  for (Iterator<Estudiante> iterator = estudiantesInscriptos.iterator();
iterator.hasNext();) {
    Estudiante estudiante = iterator.next();
    if(estudiante.getApellido()==apellido)
      estudiantesConApellido.add(estudiante);
  }
  return estudiantesConApellido;
}
```

10.3. Otro desarrollador implementó el siguiente método en la clase Carrera pero no le puso un nombre representativo. Explique qué hace el método.

```
private void ???????(){
  int index=0;
  boolean intercambio = true;
  Curso auxiliar;
  while (intercambio) {
    intercambio = false;
    index++;
    for (int i = 0; i < materias.size() - index; i++) {
      if (materias.get(i).getAño() > materias.get(i + 1).getAño()) {
        auxiliar = materias.get(i);
        materias.add(i, materias.get(i + 1));//Agrega el valor pasado en el
segundo parámetro en la posición i de la lista. El valor que se
encontraba en la posición i se mueve a la derecha
        materias.remove(i+1);
        materias.add(i+1, auxiliar);
        materias.remove(i+2);
        intercambio = true;
    }
  }
}
```

11. Implemente las clases y los métodos Java que se describen en el diagrama de clases teniendo en cuenta los siguientes detalles:

- Un grupo de un mundial está compuesto de 4 equipos. Solo los dos con mayor cantidad de puntos avanzan a la siguiente rueda. Ganador=3 puntos. Empate=1 punto. Suponga que no hay equipos con la misma cantidad de puntos una vez jugados todos los partidos del grupo En las llaves (ej. octavos, cuartos, etc.) avanzan los ganadores (los partidos no pueden terminar en empate).
- Note que el método getEquiposQueAvanzan() de la clase EtapaMundial es abstracto.
- El constructor de EtapaMundial debe inicializar la lista partidos.
- Puede crear métodos si lo considera necesario.



11.1. Para realizar estadísticas, se necesita saber la diferencia de goles de un equipo. Por ejemplo, si en los partidos disputados en el mundial por el equipo A recibió 3 goles y convirtió 10, entonces, la diferencia de goles será 7.

a.) Indique en qué clase se implementará dicha funcionalidad.
b.) Implemente en Java el/los método/s correspondientes. Si considera necesario nuevas variables defínalas y diga en qué clase se implementan.

11.2. El siguiente código no compila. Explique a qué se debe. Suponga que el método obtenerFechaPartido() se encuentra implementado y que devuelve un objeto de tipo Date válido.

```
EtapaMundial etapa=new EtapaMundial();
Equipo e1=new Equipo(); e1.setNombre("Argentina");
Equipo e2=new Equipo(); e2.setNombre("Brasil");
Partido p=new Partido(obtenerFechaPartido(), e1, e2);
etapa.addPartido(p);
```

11.3. Un desarrollador implementó el siguiente método en la clase Equipo pero no usó nombres representativos. Explique qué hace el método.

```
public float xxx(){
  int yyy=0;
  for (Iterator<Partido> iterator = partidosJugados.iterator(); iterator.hasNext();) {
    Partido partido = iterator.next();
    if((partido.getLocal().equals(this)&&partido.getResultado().ganoLocal())||
       (partido.getVisitante().equals(this)&&!partido.getResultado().ganoLocal()
                  &&!partido.getResultado().empate()))
       yyy++;
  }
  return (yyy/partidosJugados.size())*100;
}
```

12. En base al siguiente método de la clase Licencia escriba la documentación del método en formato JavaDoc:

```
public BigDecimal cuantoDebePorInfraccionesNoPagadas(LocalDate fechaDesde, LocalDate fechaHasta) {
   BigDecimal total = BigDecimal.ZERO;

   // iteramos sobre las actas de constatación asociadas a la licencia del conductor
   Iterator<ActaConstatacion> iter = actas.iterator();
   while (iter.hasNext()) {
      // obtenemos el acta actual
      ActaConstatacion acta = iter.next();

      // comprobamos que la fecha sea entre las buscadas
      if (acta.estasEnPeriodo(fechaDesde, fechaHasta) && !acta.estaPagada()) {

         // acumulamos la cantidad de infracciones
         total = total.add(acta.calcularTotalInfracciones());
      }
   }

   return total;
}
```

13. En base al siguiente método de la clase Licencia escriba la documentación del método en formato JavaDoc:

```
public int cuantasInfraccionesEnPeriodo(LocalDate fechaDesde, LocalDate fechaHasta) {
   int infracciones = 0;

   // iteramos sobre las actas de constatación de la licencia del conductor
   Iterator<ActaConstatacion> iter = actas.iterator();
   while (iter.hasNext()) {
      // obtenemos el acta actual
      ActaConstatacion acta = iter.next();

      // comprobamos que la fecha sea entre las buscadas
      if (acta.estasEnPeriodo(fechaDesde, fechaHasta)) {

         // acumulamos la cantidad de infracciones
         infracciones += acta.cuantasInfraccionesContiene();
      }
   }

   return infracciones;
}
```

14. Dadas las siguientes clases, analice el comportamiento en común para definir una interface que lo abstraiga.

```
public class Pez {
 public Pez() {

 }

public void comer() {
   System.out.println("El pez come plancton.");
 }

public void jugar() {
   System.out.println("El PEZ juega");
  }
}

public class Gato {
 public Gato() {

 }

public void comer() {
   System.out.println("El gato come plancton.");
 }

public void jugar() {
   System.out.println("El gato juega");
  }
}
```