

**Teoría ISW**

**Grupo:** \_\_\_\_\_

**Apellidos:** \_\_\_\_\_

**Nombre:** \_\_\_\_\_

# **Evaluación Teoría**

**30-01-2023**

**3h 30 min**

## **Normas a seguir:**

- No abrir el enunciado hasta que lo indique el/la profesor/a del aula.
- Se debe contestar a las preguntas en las hojas proporcionadas (no se darán hojas extra). Solo se podrá arrancar la hoja del enunciado.
- Entrega:
  - El/la alumno/a se debe identificar ante el profesor en el momento de la entrega.
  - El enunciado y las respuestas se depositarán en una caja habilitada para ello.
  - El procedimiento de entrega se realizará de forma ordenada. En cada momento solo podrá estar un/a alumno/a entregando.

**Ingeniería del Software**

ETS Ingeniería Informática

DSIC – UPV

**Curso 2022-2023**

**Nombre:**

**Grupo:**

---

**NOMBRE:**

**GRUPO:**

**Tiempo: 3 horas 30 minutos**

**Cuestiones** (2 puntos)

1. (0.75 puntos) ¿Una metodología software establece únicamente el conjunto de actividades asociadas a la producción y gestión de un producto software? Razone la respuesta.

No, esa definición correspondería al proceso de SW. En la metodología, además del conjunto de actividades a realizar, se indica quien las realiza, en qué momento, qué se obtiene en cada paso, incluso con qué herramientas se realizarán las actividades.

2. (0.5 puntos) ¿Qué beneficios aporta la separación de la capa lógica y la capa de persistencia en una arquitectura de 3 capas? Razona tu respuesta

- Cambios en la persistencia no tienen porqué afectar a la capa lógica y viceversa.
- Aislar la lógica de la aplicación en componentes separados.
- Distribución de capas en diferentes máquinas o procesos (Cliente /Servidor --- niveles vs. capas)
- Posibilidad de desarrollo en paralelo.
- Dedicación de recursos a cada una de las capas.
- REUTILIZACIÓN

3. (0.75 puntos) ¿Qué es un *Data Transfer Object* (DTO) y para qué se puede utilizar?

Representa el objeto portador de los datos. DAO puede devolver los datos al BusinessObject en un DTO. El DAO puede recibir los datos para actualizar la BD en un DTO. De la misma forma, la capa de presentación puede recibir los datos a mostrar en un DTO y puede enviar los datos en un DTO.

Nombre:

Grupo:

---

**Problemas (8 puntos)**

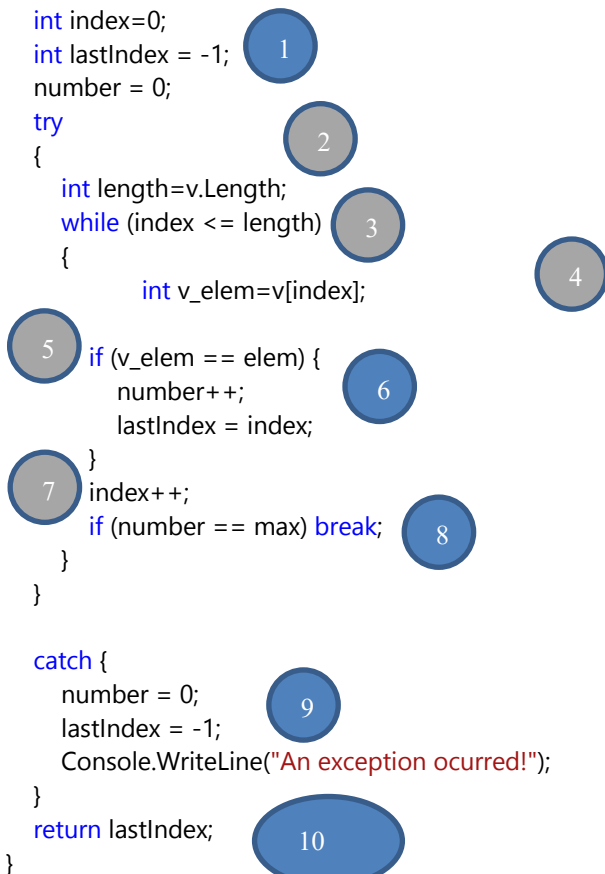
4. (1.5 puntos) Diseñe los casos de prueba para el siguiente fragmento de código siguiendo la técnica del camino básico de caja blanca, dibuje el grafo de flujo, calcule la complejidad ciclomática, especifique los caminos independientes y los casos de prueba asociados a cada camino. La función "findElement" recibe un vector ("v") que puede ser "null", un entero a buscar ("elem"), un número de veces máximo a encontrarlo ("max") y devuelve usando un parámetro variable de salida "number" el número de veces que se ha encontrado "elem" en "v". También retorna la última posición del vector (lastIndex) en la que se ha encontrado el elemento. Asuma que sólo se pueden producir excepciones en las líneas de código indicadas con un comentario.

Solución:

```
int findElement(int[]? v, int elem, int max, out int number){
    int index=0;
    int lastIndex = -1;
    number = 0;
    try
    {
        int length=v.Length;
        while (index <= length)
        {
            int v_elem=v[index];

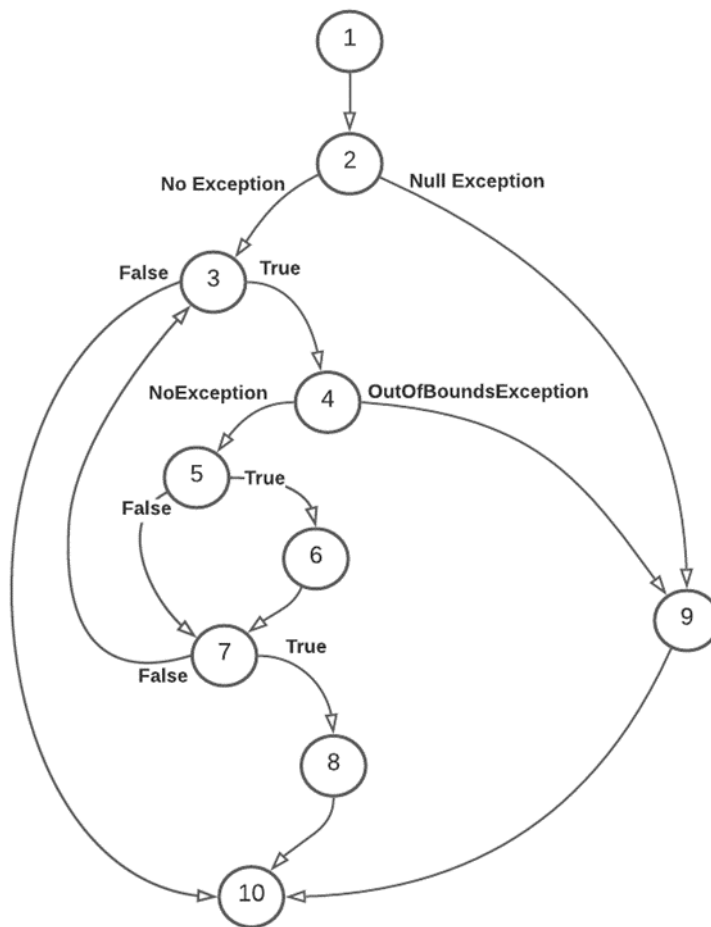
            if (v_elem == elem) {
                number++;
                lastIndex = index;
            }
            index++;
            if (number == max) break;
        }
    }

    catch {
        number = 0;
        lastIndex = -1;
        Console.WriteLine("An exception occurred!");
    }
    return lastIndex;
}
```



Nombre:

Grupo:



Regiones= 6

Predicado+1=5+1=6

A-N+2=14-10+2=6

CC=6

C1=1-2-9-10

C2=1-2-3-10

C3=1-2-3-4-9-10

C4=1-2-3-4-5-7-8-10

C5=1-2-3-4-5-7-3-10

C6=1-2-3-4-5-6-7-8-10

Path	v	elem	max	out number	Return/Consola
C1	null	-	-	0	-1 An exception Ocurred
C2	Impossible Path				
C3	{}	-	-	0	-1 An Exception Ocurred
C4	{2}	3	0	0	-1
C5	Impossible Path				
C6	{3}	3	1	1	0

**Nombre:**

**Grupo:**

---

5. (4.5 puntos) A partir del siguiente enunciado se pide obtener:
- (2.5 puntos) El diagrama de clases en UML que modela el sistema propuesto. En el diagrama de clases no hay que incluir los métodos ni los tipos de los atributos.
  - (2 puntos) El diagrama estructurado de casos de uso en UML que modela el sistema propuesto.

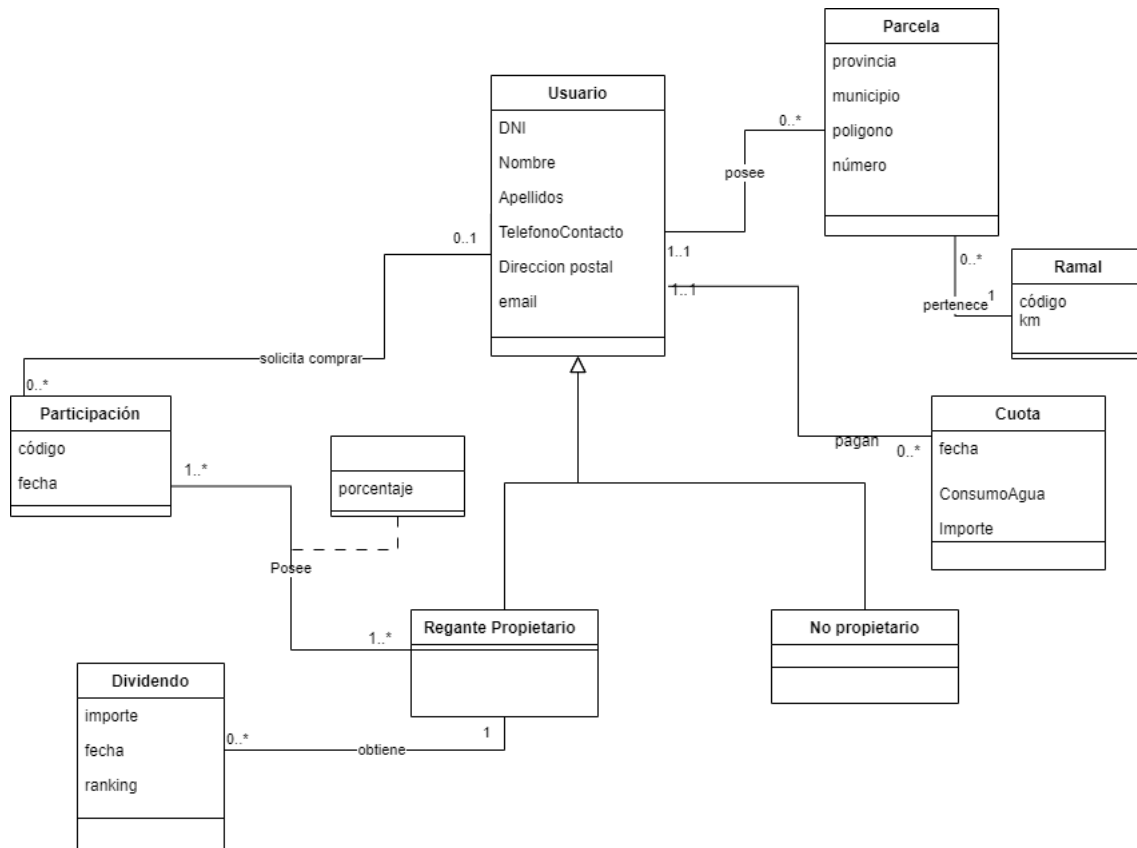
El administrador del Pozo de Collet de Mariola contrata a ISWSoft para iniciar el desarrollo de un software de soporte a la gestión de las operaciones del negocio. El Pozo de Collet de Mariola está constituido como una comunidad de usuarios que riegan sus parcelas. Existen dos tipos de usuarios, los regantes que son propietarios y los regantes que no son propietarios. La diferencia fundamental es que los primeros (regantes propietarios) han adquirido unas participaciones sobre la propiedad del pozo (que tienen un código único que las identifica y la fecha de adquisición) y por eso obtienen un dividendo o descuento sobre las cuotas que se pagan por los servicios de suministro de agua, mientras que los segundos pagan las cuotas sin ningún tipo de bonificación. El dividendo depende del porcentaje de participación adquirido por el usuario-propietario y se calcula con un conjunto de reglas que tienen en cuenta el número de participaciones, el consumo de agua del usuario. Así pues, a los propietarios se les aplica un dividendo diferente en cada ejercicio.

Los usuarios regantes (propietarios o no propietarios) pueden tener una o más parcelas contempladas en la superficie de riego. Las parcelas de riego se agrupan en ramales según la tubería principal que les suministra el agua. Actualmente el Pozo del Collet tiene 8 ramales, pero podrían aparecer nuevos si hay demanda de agua de nuevas parcelas que requieran construir nuevas tuberías principales. Cada ramal viene identificado por un código y tiene una longitud en kilómetros.

El software por desarrollar debe dar soporte a la gestión de altas de usuarios, los cambios de tipo propietario/no propietario y la compra de participaciones. En concreto, un usuario potencial puede realizar una solicitud de alta en el sistema. Para ello facilitará sus datos personales (DNI, nombre, apellidos, dirección postal, teléfono de contacto, email) y opcionalmente el número de participaciones sobre la propiedad que quiere adquirir y/o la(s) parcela(s) que quiere regar (provincia, municipio, polígono y número de parcela). En caso de que el alta incluya parcelas de riego, la solicitud será revisada por el encargado del mantenimiento del pozo que realizará un estudio de viabilidad necesario para que el alta se confirme. Una vez dado de alta el usuario puede solicitar en cualquier momento la adquisición de participaciones (pudiendo pasar de no propietario a propietario). Una participación puede ser adquirida por varios usuarios y en ese caso se debe especificar el porcentaje de la participación que tendrá cada usuario. También se podrá solicitar el riego de nuevas parcelas (pudiendo pasar de no regante a regante), previa revisión por el encargado del mantenimiento, quien deberá autorizar la solicitud. Todas estas acciones las puede ejecutar el encargado del mantenimiento del pozo en nombre de los usuarios.

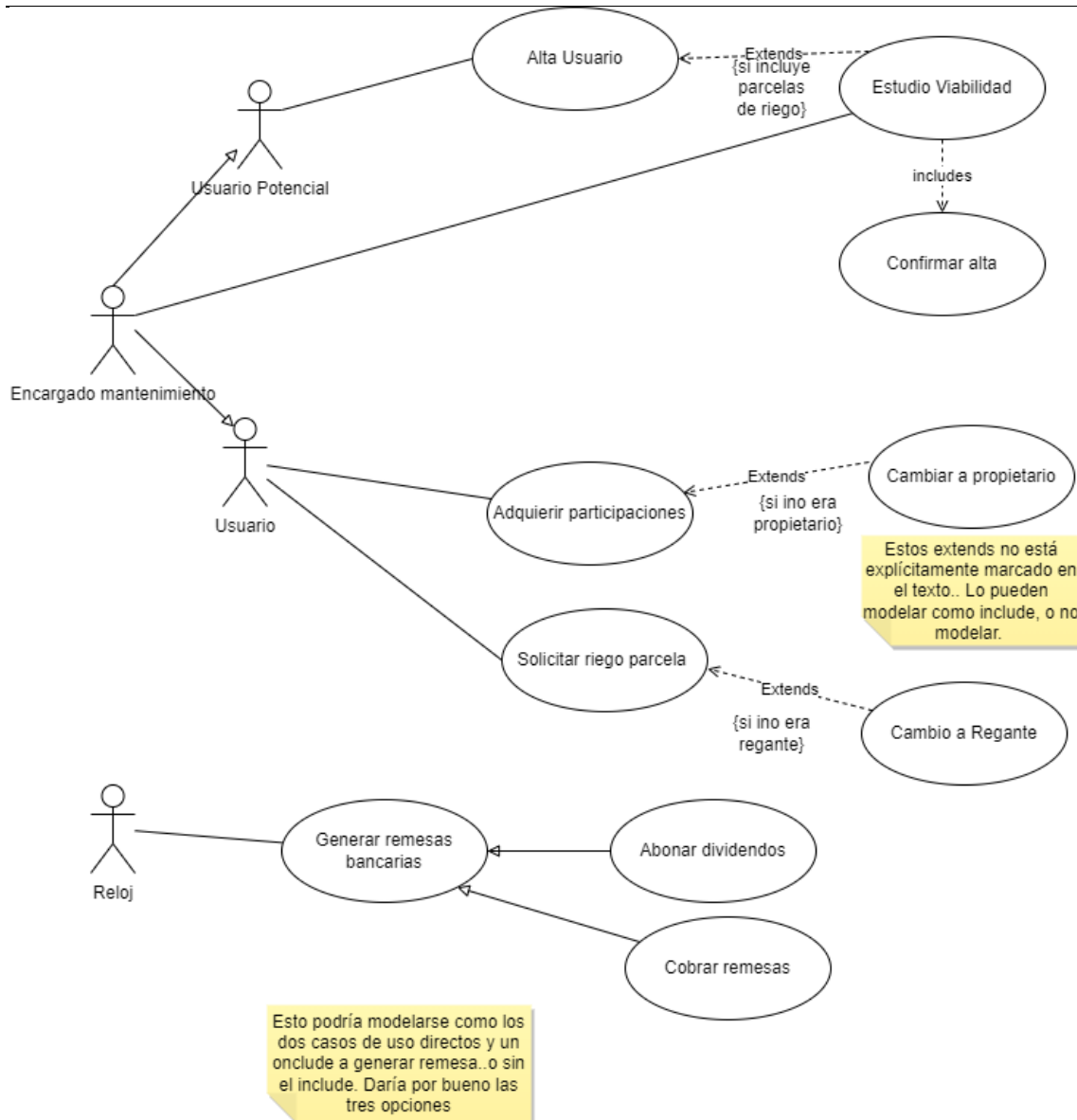
**Grupo:**

Por último, el sistema generará de forma automática a final de mes las remesas bancarias para el cobro de cuotas a los usuarios o abono de los dividendos teniendo en cuenta el tipo de usuario. Las cuotas generadas para todos los regantes incluirán la fecha, el importe total y el consumo de agua en metros cúbicos. Los dividendos generados para cada regante propietario contendrán la fecha, el importe del dividendo y un número que indica la posición en el ranking de ganancias de los propietarios.



Nombre:

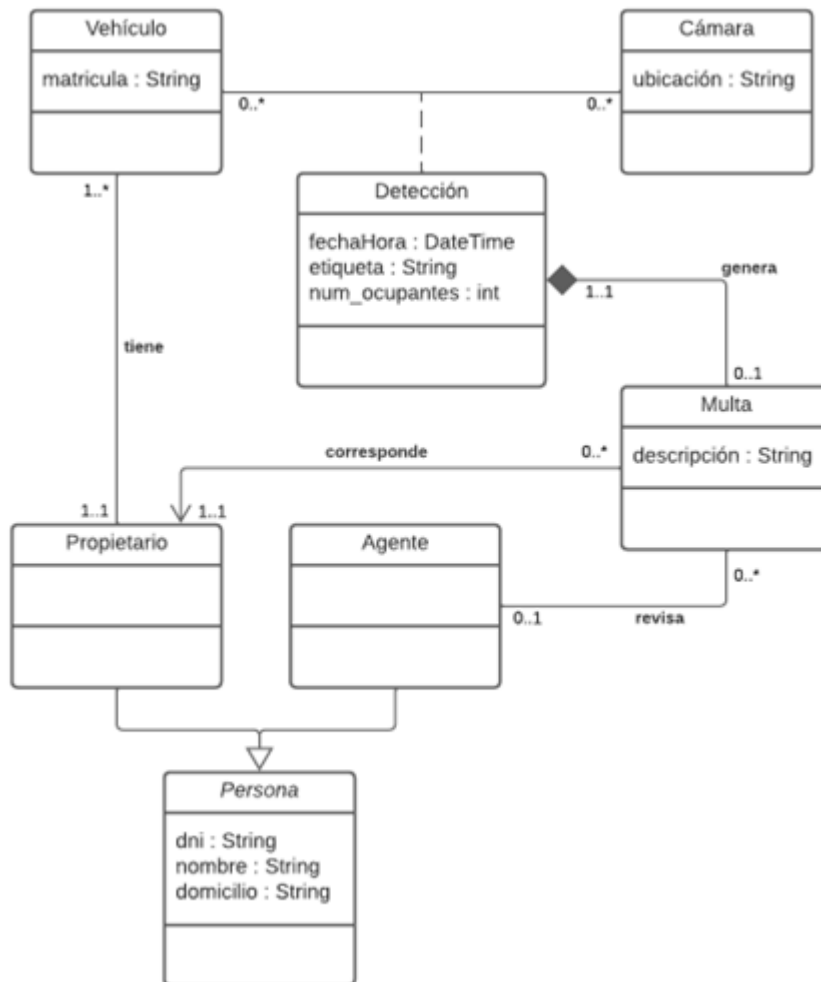
Grupo:



Nombre:

Grupo:

(2 puntos) A partir del siguiente diagrama de clases en UML:



**Nota:** La clase **Persona** es abstracta y hay una restricción de navegación entre **Multa** y **Propietario**.

- a) (1 punto) Realice el diseño de objetos en C# siguiendo las guías de diseño vistas (no indicar ningún método).

```
public class Camara
{
    private String ubicación;

    private ICollection<Deteccion> detecciones;
}

public class Vehículo
{
    private String matricula;
```



Nombre:

Grupo:

---

```
        private ICollection<Deteccion> detecciones;
        private Propietario pertenece;
    }

    public class Deteccion
    {
        private DateTime fechaHora;
        private String etiqueta;
        private int num_ocupantes;

        private Vehiculo vehiculo;
        private Camara camara;
        private Multa genera;
    }

    public class Multa
    {
        private String descripción;

        private Deteccion produce;
        private Propietario corresponde;
        private Agente revisa;
    }

    public abstract class Persona
    {
        private String dni;
        private String nombre;
        private String domicilio;
    }

    public class Propietario : Persona
    {
        private ICollection<Vehiculo> tienes;
        //private ICollection<Multa> multas;
        //No hace falta por la restricción de navegación
    }

    public class Agente : Persona
    {
        private ICollection<Multa> revisadas;
    }
}
```

- b) (0.5 puntos) Realice el diseño en C# del constructor con parámetros de cada clase (sólo la cabecera, no los implemente. No defina el constructor sin parámetros).

```
public Camara(string ubicación)
public Vehiculo(string matricula, Propietario pertenece)
public Deteccion(DateTime fechaHora, string etiqueta, int num_ocupantes,
                 Vehiculo vehiculo, Camara camara)
public Multa(string descripción, Deteccion produce, Propietario
             corresponde)
public Persona(string dni, string nombre, string domicilio)

public Propietario(string dni, string nombre, string domicilio)
                 //, Vehiculo vehiculo) //SE RELAJA ESTE CONSTRUCTOR
: base(dni, nombre, domicilio)
```

**Nombre:**

**Grupo:**

---

```
public Agente(string dni, string nombre, string domicilio)
    :base(dni, nombre, domicilio) {}
```

- c) (0.5 puntos) Usando los constructores del apartado anterior, implemente el código necesario para tener un vehículo que haya sido detectado por una cámara y que se haya generado una multa. Use los valores que desee para el resto de los atributos.

```
Propietario propietario = new Propietario("12345678Z", "Juan", "La univerd, 3");
Vehiculo vehiculo = new Vehiculo("123ABC", propietario);
propietario.AddVehiculo(vehiculo); // Por el constructor relajado
```

```
Camara camara = new Camara("Norte");
Deteccion deteccion = new Deteccion(DateTime.Now, "C", 1, vehiculo, camara);
vehiculo.AddDeteccion(deteccion); // Por consistencia del modelo
camara.AddDeteccion(deteccion); // Por consistencia del modelo
```

```
Multa multa = new Multa("No puedes circular por VAO", deteccion, propietario);
deteccion.SetMulta(multa); // Por consistencia del modelo
//propietario.AddMulta(multa); // No se pone por la restricción de navegación
```