Universidad de Oviedo

ACUERDO DE FECHA 16 DE ABRIL DE 2024, ADOPTADO POR LA COMISIÓN CALIFICADORA DESIGNADA PARA RESOLVER EL CONCURSO-OPOSICIÓN LIBRE CONVOCADO POR RESOLUCIÓN DE LA UNIVERSIDAD DE OVIEDO, DE FECHA 12 DE JUNIO DE 2023 (BOE DE FECHA 26 DE JUNIO DE 2023), PARA LA PROVISIÓN DE PLAZAS DE PERSONAL LABORAL CON LA CATEGORÍA DE PROGRAMADOR, GRUPO II, APARTADO "A" DE PLAZAS (PUESTOS CON NÚMEROS DE ORDEN 202, 203, 204, 206, 221 Y 228)

Reunida el 16 de abril de 2024, la Comisión Calificadora designada para resolver el concurso-oposición libre convocado para la provisión de plazas de **Programador**, Grupo II, apartado "A" de plazas, puestos con números de orden 202, 203, 204, 206, 221 y 228 de la RPT de personal laboral de la Universidad de Oviedo, adopta el siguiente

ACUERDO

Único.- Hacer pública en la página web de la Universidad de Oviedo la plantilla de respuestas correctas del segundo ejercicio de la fase de oposición, para el apartado "A" de plazas de Programador, que se acompaña como Anexo.

Las respuestas dadas a algunas de las preguntas del ejercicio son orientativas, quedando a criterio de la Comisión Calificadora la apreciación de otras respuestas posibles como igualmente válidas.

Oviedo, 16 de abril de 2024

LA SECRETARIA

MERCBP+o

Fdo. María Mercedes Palacio Menéndez

Universidad de Oviedo

Proceso selectivo para la provisión de plazas de personal laboral con la categoría de Programador, Grupo II.

Apartado A - Segundo ejercicio

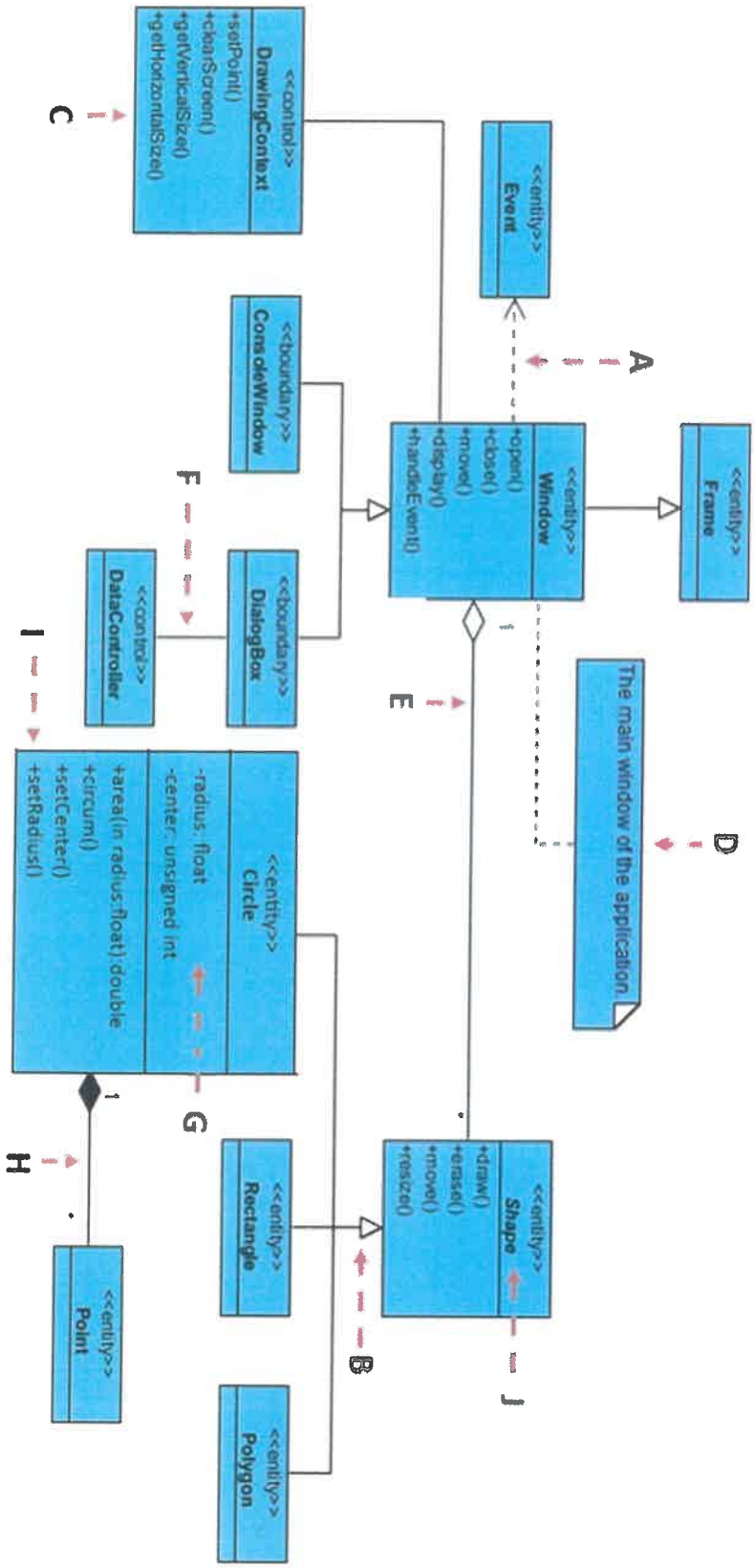
16 de abril de 2024

NO ABRA EL CUESTIONARIO HASTA QUE SE LE INDIQUE

Ejercicio 1

Puntuación máxima: 1,5 puntos.

Dado el siguiente diagrama UML de clases, responda a las siguientes preguntas:



Cada una de estas preguntas se valora con un máximo de 0,3 puntos.

a) Dentro de la clasificación de diagramas UML de clases, ¿se corresponde a un diagrama de comportamiento o de estructura?

Estructura

b) Del anterior diagrama identifique al menos una relación con multiplicidad 1 a *, indicando las entidades implicadas.

Window - shape
Circle - point

c) Identifique si hay algún atributo público. Si existe, indique el nombre.

No hay

d) Identifique si existe alguna clase abstracta. Si existe, indique el nombre.

Shape

e) Identifique los siguientes elementos en el diagrama anterior, indicando la letra correspondiente:

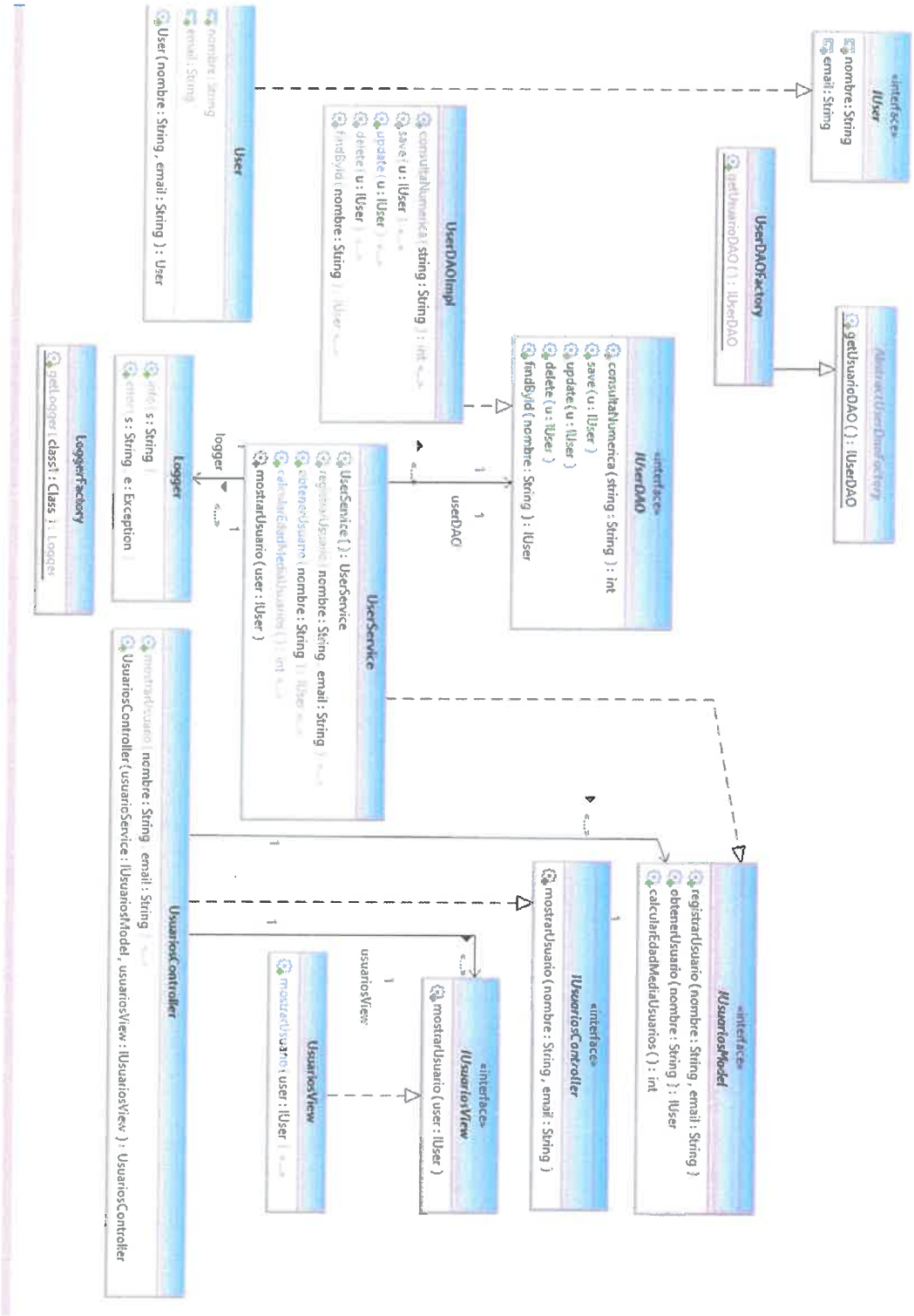
Dependencia	A
Clase de control	C
Asociación	F
Agregación	E
Nota	D
Atributo	G
Clase abstracta	J
Generalización	B
Operación	I
Composición	H

Ejercicio 2

Puntuación máxima: 1,5 puntos.

Se muestra un diagrama de clases de la fase de diseño representando la utilización de varios patrones de diseño. Posteriormente se muestran trozos de código de varias partes de la implementación.

Observe atentamente el siguiente diagrama y los bloques de código adjuntos.



Bloque 1

```
package EJ1;
public class UserService implements IUsuariosModel {
   private IUserDAO userDAO;
   private Logger logger;
   public UserService() {
         this.userDAO = XXX-1-XXX
         this.logger = XXX-2-XXX
    }
[...]
```

Bloque 2

```
package EJ1;
public class Main {
    public static void main(String[] args) {
        IUsuariosModel usuariosmodel = XXX-3-XXX;
        IUsuariosView usuariosView = new UsuariosView();
        IUsuariosController usuariosController = XXX-4-XXX;
        XXXXXXXXX.mostrarUsuario("Juan", "juan@example.com");
    }
}
```

Teniendo en cuenta el correcto uso de los patrones de diseño utilizados en el diagrama y una correcta compilación y ejecución, conteste a las siguientes preguntas:

Cada una de estas preguntas se valora con un máximo de 0,3 puntos.

a) ¿Qué sentencia va en el campo oculto XXX-1-XXX?

UserDAOFactory.getUsuarioDAO();

b) ¿Qué sentencia va en el campo oculto XXX-2-XXX?

LoggerFactory.getLogger(UserService.class);

c) ¿Qué sentencia va en el campo oculto XXX-3-XXX?

new UserService();

d) ¿Qué sentencia va en el campo oculto XXX-4-XXX?

new UsuariosController(usuariosmodel, usuariosView);

e) ¿Qué línea de código incluirías al final del método *main* para que se cree un usuario de nombre "pepe" y de email "pepe@uniovi.es"?

usuariosmodel.registrarUsuario("pepe", "pepe@uniovi.es");

Ejercicio 3

Puntuación máxima: 2 puntos.

A continuación, se muestra el código fuente de un programa Java, basado en Spring y Maven, que utiliza las dependencias necesarias para compilar y ejecutarse correctamente y sin errores sobre una máquina virtual OpenJDK 11.

Persona.java

```java
package es.uniovi.sic.examen.gestoraccesos;

import java.util.Objects;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
public class Persona {
   private String nombre;
   private String apellidos;
   private String dni;

   public Persona(String nombre, String apellidos, String dni) {
      this.nombre = nombre;
      this.apellidos = apellidos;
      this.dni = dni;
   }

   @Override
   public int hashCode() {
      return Objects.hash(dni);
   }

   @Override
   public boolean equals(Object obj) {
      if (this == obj)
         return true;
      if (obj == null)
         return false;
      if (getClass() != obj.getClass())
         return false;
      Persona other = (Persona) obj;
      return Objects.equals(dni, other.dni);
   }
}
```

GestorAccesos.java

```java
package es.uniovi.sic.examen.gestoraccesos;

import java.util.HashSet;
import java.util.Set;

import org.springframework.stereotype.Service;

import lombok.Getter;

@Service
@Getter
public class GestorAccesos {
   private int contadorAccesos;
   private Set<Persona> personasQueHanAccedido;
```

```
    public GestorAccesos() {
        contadorAccesos = 0;
        personasQueHanAccedido = new HashSet<Persona>();
    }

    public void registrarNuevoAcceso(Persona persona) {
        contadorAccesos++;
        personasQueHanAccedido.add(persona);
    }
}
```

GestorTaquilla.java

```
package es.uniovi.sic.examen.gestoraccesos;

import java.util.Set;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class GestorTaquilla {

    public static void main(String[] args) {
        ApplicationContext contexto = new
ClassPathXmlApplicationContext("applicationContext.xml");

        GestorAccesos gestorAccesos1 = (GestorAccesos)
contexto.getBean("gestorAccesos");
        gestorAccesos1.registrarNuevoAcceso(new Persona("María", "Fernández
García", "12345678A"));
        gestorAccesos1.registrarNuevoAcceso(new Persona("Marcos", "Martínez
Rodríguez", "98765432B"));
        gestorAccesos1.registrarNuevoAcceso(new Persona("Carmen", "López
González", "11222333C"));

        System.out.println("1.- Valor del atributo contadorAccesos en
gestorAccesos1: " + gestorAccesos1.getContadorAccesos());
        System.out.println("2.- Tamaño de la colección personasQueHanAccedido en
gestorAccesos1: " + gestorAccesos1.getPersonasQueHanAccedido().size());

        boolean resultado = (gestorAccesos1.getContadorAccesos() > 2) ? true :
false;
        System.out.println("3.- Resultado: " + resultado);

        Set<Persona> personas = gestorAccesos1.getPersonasQueHanAccedido();
        personas.forEach(p -> System.out.println("4.-" + p.getDni()));

        GestorAccesos gestorAccesos2 = (GestorAccesos)
contexto.getBean("gestorAccesos");
        Persona persona = new Persona("Manuel", "Sánchez Pérez", "44555666D");
        gestorAccesos2.registrarNuevoAcceso(persona);
        gestorAccesos2.registrarNuevoAcceso(persona);
        gestorAccesos2.registrarNuevoAcceso(persona);

        System.out.println("5.- Valor del atributo contadorAccesos en
gestorAccesos2: " + gestorAccesos2.getContadorAccesos());
        System.out.println("6.- Tamaño de la colección personasQueHanAccedido en
2: " + gestorAccesos2.getPersonasQueHanAccedido().size());

        String cadena1 = new String("cadena");
        String cadena2 = new String("cadena");
        System.out.println("7.- " + (cadena1 == cadena2));
        System.out.println("8.- " + (cadena1.equals(cadena2)));
    }
}
```

Los dos siguientes ficheros se muestran como referencia, pero no tienen impacto en la funcionalidad del programa, ya que en el enunciado se asegura que el programa compila y funciona correctamente.

Fichero de configuración de Spring applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
   http://www.springframework.org/schema/context
   http://www.springframework.org/schema/context/spring-context-3.0.xsd">

   <context:component-scan base-package="es.uniovi.sic.examen.gestoraccesos"/>
</beans>
```

Fichero de configuración de Maven pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>es.uniovi.sic.eadmon</groupId>
  <artifactId>examen</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Examen</name>

  <dependencies>
   <dependency>
       <groupId>org.springframework</groupId>
       <artifactId>spring-beans</artifactId>
       <version>5.3.32</version>
   </dependency>
   <dependency>
       <groupId>org.springframework</groupId>
       <artifactId>spring-context</artifactId>
       <version>5.3.32</version>
   </dependency>
   <dependency>
       <groupId>org.projectlombok</groupId>
       <artifactId>lombok</artifactId>
       <version>1.18.24</version>
       <scope>provided</scope>
   </dependency>
  </dependencies>
</project>
```

En base a todos estos ficheros se deben indicar los valores que se muestran por la salida estándar a través de las sentencias *System.out.println* del método *main* de la clase *GestorTaquilla* cuando esta clase es ejecutada.

Cada una de estas preguntas se valora con un máximo de 0,25 puntos.

a) 1.- Valor del atributo contadorAccesos en gestorAccesos1:

3

b) 2.- Tamaño de la colección personasQueHanAccedido en gestorAccesos1:

3

c) 3.- Resultado:

true

d) 4.-

11222333C
98765432B
12345678A

e) 5.- Valor del atributo contadorAccesos en gestorAccesos2:

6

f) 6.- Tamaño de la colección personasQueHanAccedido en 2:

4

g) 7.-

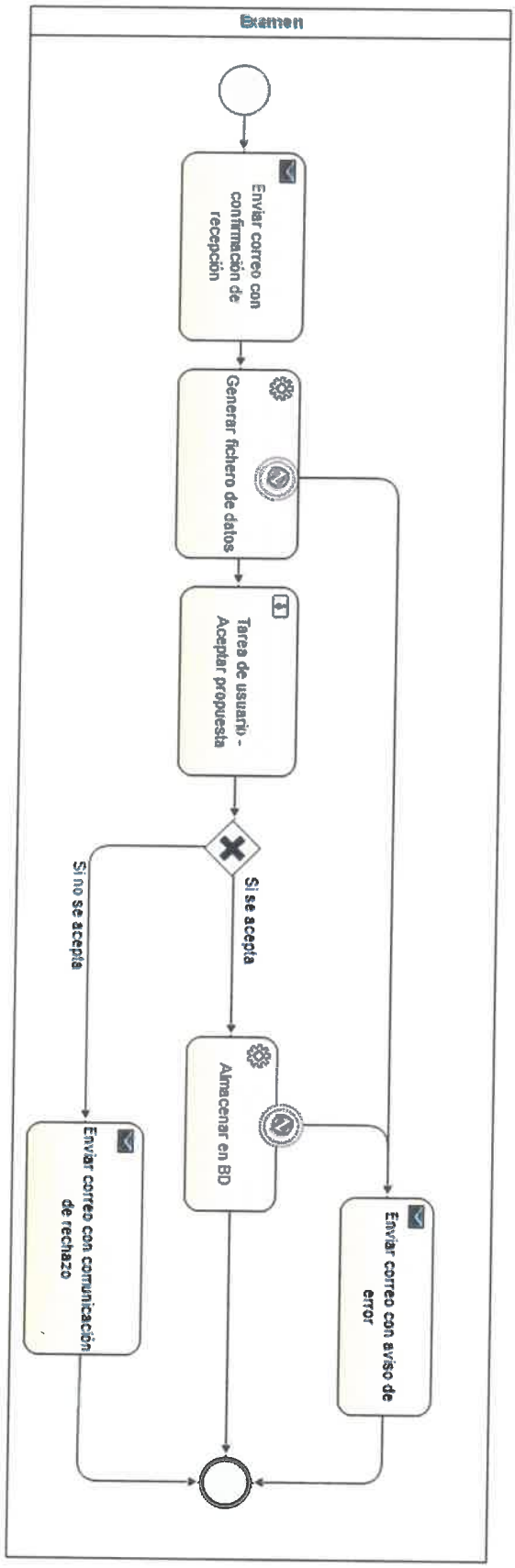
false

h) 8.-

true

Ejercicio 4

Puntuación máxima: 1,5 puntos.



Sobre el diagrama Activiti BPMN v5 anterior responda a las siguientes preguntas:

a) Una vez que se inicia la ejecución del flujo Activiti anterior, el proceso se ejecuta de forma ininterrumpida hasta llegar a un punto del proceso. ¿En qué punto se para la ejecución de este flujo hasta que se requiere la intervención humana para continuar? Indique el nombre de la tarea Activiti en la que se para la ejecución.

Puntuación máxima: 0,4 puntos.

Tarea de usuario - Aceptar propuesta

b) ¿Cuántos correos electrónicos se envían durante el proceso completo si el usuario acepta la propuesta y no se lanza ninguna excepción durante todo el proceso?
Puntuación máxima: 0,3 puntos.

1

c) ¿Cuántos correos electrónicos se envían durante el proceso completo si se lanza una excepción durante la ejecución de la tarea "Generar fichero de datos"?

Teniendo en cuenta que:

- La excepción lanzada es de tipo *org.activiti.engine.delegate.BpmnError*.
- La tarea "Generar fichero de datos" es de tipo *Service Task* y está configurada de la siguiente forma:
  - *Task type*: *Delegate expresión*
  - *Delegate expression*: *${generadorFicheroDeDatos}*
- Siendo *generadorFicheroDeDatos* un bean de Spring que se corresponde con un objeto de una clase que implementa *org.activiti.engine.delegate.JavaDelegate*.

Puntuación máxima: 0,4 puntos.

2

d) ¿Qué usuarios pueden ejecutar la tarea "Tarea de usuario - Aceptar propuesta" teniendo en cuenta que tiene configuradas las siguientes propiedades?:

- *activiti:candidateUsers="kermit, gonzo"*
- *activiti:candidateGroups="management, accountancy"*

Teniendo en cuenta que la asignación de usuarios a grupos es la siguiente:

- El usuario *kermit* pertenece al grupo *management*.
- El usuario *gonzo* pertenece al grupo *it*.
- El usuario *piggy* pertenece al grupo *accountancy*.
- El usuario *animal* pertenece al grupo *it*.

Puntuación máxima: 0,4 puntos.

kermit, gonzo, piggy

Ejercicio 5

Puntuación máxima: 2 puntos.

Dadas las siguientes tablas, con los scripts de creación que se indican y los siguientes datos. Responda a las siguientes preguntas:

```
create table alumnos (
id  INT8,
dni VARCHAR(12) not null,
nombre VARCHAR(20) not null,
primerApellido VARCHAR(20) not null,
segundoApellido VARCHAR(20),
fechaNacimiento DATE );
```

```
create table profesores (
id  INT8,
dni VARCHAR(12) not null,
nombre VARCHAR(20) not null,
primerApellido VARCHAR(20) not null,
segundoApellido VARCHAR(20),
fechaNacimiento DATE,
sueldoMes SMALLFLOAT);
```

Tabla *Alumnos*

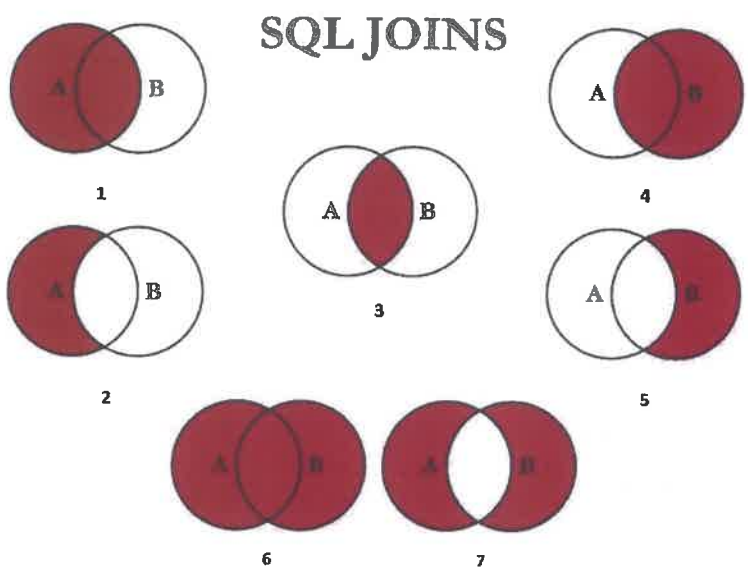
ID	DNI	NOMBRE	PRIMERAPELLIDO	SEGUNDOAPELLIDO	FECHANACIMIENTO
1	11	ANGEL	MARTINEZ	GARCIA	2002-10-10
2	12	MARIA	FLOREZ	GARCIA	2001-10-10
3	13	ANA	FLOREZ	GARCIA	1960-10-10
4	14	CARMEN	FLOREZ	GARCIA	1970-10-10
5	15	COVADONGA	FLOREZ	PEREZ	1990-10-10
6	16	BELEN	FLOREZ	JIMENEZ	1999-10-10
7	17	INES	FLOREZ	GONZALEZ	1996-10-10

Tabla *Profesores*

ID	DNI	NOMBRE	PRIMERAPELLIDO	SEGUNDO APELLIDO	FECHA NACIMIENTO	SUELDOMES
1	11	ANGEL	MARTINEZ	GARCIA	2002-10-10	NULL
2	12	MARIA	FLOREZ	GARCIA	2001-10-10	5
3	14	CARMEN	FLOREZ	GARCIA	1970-10-10	4
4	15	COVADONGA	FLOREZ	GARCIA	1990-10-10	2
5	99	LUIS	MAGIA	GARCIA	1975-10-10	2
0	88	DAVID	MELOSO	GOMEZ	1986-10-10	NULL
0	18	JAVIER	DIAZ	GOMEZ	1986-10-10	2

Cada una de estas preguntas se valora con un máximo de 0,4 puntos.

a) Dada la siguiente gráfica de las distintas joins entre las tablas *Alumnos* A y *Profesores* B, identifique cada gráfica con su correspondiente consulta:



SELECT * FROM ALUMNOS A LEFT JOIN PROFESORES B ON A.DNI=B.DNI WHERE B.DNI IS NULL	2
SELECT * FROM ALUMNOS A INNER JOIN PROFESORES B ON A.DNI=B.DNI	3
SELECT * FROM ALUMNOS A LEFT JOIN PROFESORES B ON A.DNI=B.DNI	1
SELECT * FROM ALUMNOS A FULL OUTER JOIN PROFESORES B ON A.DNI=B.DNI	6
SELECT * FROM ALUMNOS A RIGHT JOIN PROFESORES B ON A.DNI=B.DNI WHERE A.DNI IS NULL	5
SELECT * FROM ALUMNOS A RIGHT JOIN PROFESORES B ON A.DNI=B.DNI	4
SELECT * FROM ALUMNOS A FULL OUTER JOIN PROFESORES B ON A.DNI=B.DNI WHERE A.DNI IS NULL OR B.DNI IS NULL	7

b) Indique la sentencia SQL para conocer la media del sueldo de los profesores.

```
select avg(sueldoMes) from profesores
```

c) Indique la sentencia SQL para conocer los alumnos que son también profesores para los que no se conozca su sueldo.

```
select * from alumnos a, profesores b
where a.dni=b.dni and sueldoMes is null
```

d) Indique la sentencia SQL para conocer la estadística de cuantos alumnos hay por año de nacimiento.

```
select year(fechaNacimiento),count(*) from alumnos
group by 1
```

e) Indique la sentencia SQL para conocer los profesores que no son alumnos.

```
select * from alumnos a right join profesores b on a.dni = b.dni where a.dni is null
```

Ejercicio 6

Puntuación máxima: 1,5 puntos.

Dado el siguiente fragmento de un fichero *pom.xml*. Responda a las preguntas sobre él.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <parent>
        <groupId>***</groupId>
        <artifactId>***</artifactId>
        <version>***</version>
    </parent>
    <modelVersion>***</modelVersion>
    <groupId>***</groupId>
    <artifactId>***</artifactId>
    <packaging>***</packaging>
    <version>***</version>
    <name>***</name>
    <description>***</description>
    ...
    <properties>
        <maven.javadoc.skip>true</maven.javadoc.skip>
    ...
    <scm>
        <developerConnection>
    ...
    <modules>
      <module>s1</module>
    ...
    <distributionManagement>
    ...
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-checkstyle-plugin</artifactId>
    ...
    <dependencies>
        <dependency>
            <groupId>***</groupId>
            <artifactId>build-tools</artifactId>
            <version>***</version>
        </dependency>
    </dependencies>
</project>
```

Cada una de estas preguntas se valora con un máximo de 0,3 puntos.

a) ¿A través de qué etiqueta se declara la información sobre el proyecto padre? ¿Qué otras etiquetas se engloban en la información sobre el padre?

<parent> <groupId> <artifactId> <version>

b) Según lo indicado en el *xlmns* del pom anterior, ¿qué se debe incluir en la etiqueta *<modelVersion>*?

4.0.0

c) ¿Dentro de qué etiqueta se incluyen las definiciones de las propiedades?

<properties>

d) Si se define la *property <java.source.version>1.8</java.source.version>*, ¿cómo se puede acceder a esa *property* posteriormente desde el propio fichero *pom.xml*?

${java.source.version}

e) ¿Con qué etiqueta se especifica la lista de dependencias?

<dependencies>