**Hay un error de compilación en este código. ¿Cuál es el motivo de este error?**

**EntityManagerFactory factoria =**

**Persistence.*createEntityManagerFactory*("PrimerProyectoJPA");**

**EntityManager em = factoria.createEntityManager();**

**Cliente cli = em.find("43434343A");**

**System.out.println(cli.getNombre());**



La variable cli debería ser declarada como Object.



Falta hacer una conversión o cash. Sería así: *Cliente cli = (Cliente) em.find("43434343A");*



Falta un argumento en el método *find()*para especificar el tipo de objeto que devolverá la función. Debería ser así: *Cliente cli = em.find(Cliente.class, "43434343A");*

Falta un argumento en el método find() para especificar la clase a la que pertenece el objeto que será devuelto.

--Bien--

**La clase Java que se encarga de la administración de los objetos de entidad es:**



*EntityManager*



*EntityManagerFactory*



*TypedQuery*



*Query*

Un objeto de la clase EntityManager es el encargado de la administración de los objetos de entidad, permitiendo realizar sobre ellos las operaciones CRUD.

--Bien—

**Identifica las condiciones que debe cumplir una clase para ser utilizada como *JavaBean*.**



Tener un constructor sin argumentos.



Las propiedades privadas y accesibles mediante métodos *get/set*.



No debe implementar ninguna interfaz.



No debe heredar de ninguna otra clase.



Debe implementar la interfaz *Serializable*.



Debe implementar la interfaz *Serializar*.

--Bien—

**La configuración de la unidad de persistencia...**



Se encuentra en el archivo JPA.xml.



Se encuentra en el archivo persistence.xml.



Se encuentra distribuida en las distintas clases de entidad.

La configuración de cada unidad de persistencia se encuentra en el archivo persistence.xml.

--BIEN---

**Dada la siguiente línea de código:**

**EntityManagerFactory factoria =**

**Persistence.*createEntityManagerFactory*("UPFerreteria");**



*UPFerreteria* es el nombre del proyecto.



*UPFerreteria* es el nombre de un fichero xml que contiene la configuración de conexión a la base de datos.



*UPFerreteria* es el nombre de una unidad de persistencia cuya configuración está en el archivo *persistence.xml*.

El método createEntityManagerFactory() recibe como argumentos el nombre de la unidad de persistencia que encierra los valores de conexión a una base de datos con la que queremos trabajar.

--BIEN--

**Indica las anotaciones que sirven para configurar una clase de entidad.**



*@PrimaryKey*



*@id*



*@Entity*



*@EntityClass*



*@OneToMany*



*@1ToN*



*@JoinColumn*



*@ManyColumns*

Son anotaciones para configuración de clases de entidad @Id, @Entity, @OneToMany y @JoinColumn.

--MAL--

**Dado el siguiente código:**

**EntityManagerFactory factoria =**

**Persistence.createEntityManagerFactory("PrimerProyectoJPA");**

**EntityManager em = factoria.createEntityManager();**

**String sql = "SELECT pro FROM Producto pro WHERE pro.codigo =**

**?1";**

**TypedQuery query = em.createQuery(sql,**

**Producto.class);**

**// Aquí falta algo**

**Producto p = query.getSingleResult();**

**System.out.println(p.getDescripcion() + "     -      " +**

**p.getPrecio()+"€");**

**¿Qué habría que insertar en la posición del comentario para que ejecutara sin errores?**



*query.set (1, "TOR7");*



*query.setParameter(1, "TOR7");*



*query.setParameter("TOR7", 1);*

Dentro de la sentencia JPQL aparece ?1, lo que significa que hay que suministrar un parámetro en la posición 1. La sentencia correcta sería query.getParameter(1, “TOR7”);

--Bien—

**Selecciona las afirmaciones correctas sobre el mapeo objeto-relacional.**



Pretende acercar el mundo de las bases de datos al mundo de los objetos Java.



Pretende acercar el mundo de los objetos Java a los sistemas de gestión de base de datos.



Trabaja sobre una base de datos orientada a objetos virtual, es decir, en memoria.



El *framework* ORM es el encargado de mapear las clases Java con las tablas físicas de base de datos.



Trabaja contra una base de datos orientada a objetos virtual, basada en unas clases especiales que heredan de otra clase base, definida con ese objetivo.

En el proceso de mapeo objeto-relacional, por un lado tenemos una base de datos con su estructura relacional, con sus tablas físicas y sus asociaciones, y por otro lado, tenemos clases Java que imitan la estructura de la base de datos y con las que interactúa la aplicación.

--MAL—

**Un Framework ORM... (varias respuestas correctas)**



Es un marco de trabajo que ofrece soporte para el desarrollo de aplicaciones con bases de datos.



Se basa en el procedimiento de Mapeo objeto-relacional.



Es un marco de trabajo para aplicaciones web con bases de datos.



Toma sus iniciales de las palabras *Object Relational Modeling*.



Toma sus iniciales de las palabras *Object Relational Mapping*.

Un framework ORM es un marco de trabajo que ofrece un soporte completo para el desarrollo de aplicaciones con bases de datos relacionales y se basa en el procedimiento de Mapeo Objeto-Relacional.

--MAL--

**El método *getResultList*...**



Devuelve un array de objetos*String*.



Devuelve una colección de un objeto que pertenece a una clase de entidad.



Devuelve una colección de objetos que podrá ser de cualquier tipo.

--MAL--

**Dadas las siguientes líneas de código:**

**TypedQuery query = em.createQuery**

**("SELECT pro.codigo, pro.descripcion FROM Producto pro",**

**Object[].class);**

**List resultados = query.getResultList();**



La variable *resultados* contendrá una colección de objetos de la clase de entidad *Producto*.



La variable *resultados* contendrá una colección de arrays, cada uno de ellos con dos elementos.



La variable resultados contendrá una colección de objetos *String*.

--Mal— Creo que es la fosforita

**La perspectiva Database Development de Eclipse...**



Permite crear conexiones a bases de datos, pero carece de editor SQL, así que no es posible realizar consultas.



Permite crear conexiones a bases de datos y ejecutar consultas SQL sobre ellas.



Permite crear conexiones a bases de datos e inspeccionar los datos desde las herramientas gráficas, pero no dispone de editor SQL.

La perspectiva Database Development de Eclipse permite crear conexiones a base de datos. A través de dichas conexiones, podemos inspeccionar las tablas y ejecutar consultas SQL.

---Bien--

**Dado el siguiente código:**

**EntityManagerFactory factoria =**

**Persistence.createEntityManagerFactory("PrimerProyectoJPA");**

**EntityManager em = factoria.createEntityManager();**

**EntityTransaction et = em.getTransaction();**

**et.begin();**

**Query query = em.createQuery**

**("UPDATE Producto p SET p.precio = p.precio\*0.1");**

**int valor = query.executeUpdate();**

**et.commit();**

**¿Qué valor tendrá la variable valor tras la ejecución?**



Un valor numérico igual al número de filas de la tabla Producto.



1, si la operación se ejecutó con éxito y 0, en caso contrario.



Un valor numérico igual al número de filas de la tabla*Producto*, o -1 si la operación falló.

El método executeUpdate() devuelve el número de filas afectadas por una operación de tipo INSERT, UPDATE o DELETE.

--Bien--

**¿Con qué letra de la palabra CRUD podríamos relacionar el método *merge()* de la clase*EntityManager*?**



C



R



U



D

Con la U de Update, ya que merge() permite actualizar una fila de la base de datos a partir de los datos suministrados por el objeto pasado como argumento.

--Bien--

**Marca la afirmación correcta.**



JPA (*Java Persistence Aplication*) es el *framework* estándar proporcionado por Java Enterprise Edition (Java EE) para la capa de persistencia, que implementa el concepto de ORM.



JPA (*Java Persistence API*) es el *framework* estándar proporcionado por Java Standard Edition para la capa de persistencia, que implementa el concepto de ORM.



JPA (*Java Persistence API*) es el *framework* estándar proporcionado por Java Enterprise Edition (Java EE) para la capa de persistencia, que implementa el concepto de ORM.

---Bien--