<https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00006_Representaci%C3%B3n%20de%20clases>

Llegó el momento de poner a prueba nuestros conocimientos. Para ello, nos han propuesto el siguiente desafío:

Primero vamos a tener 20 preguntas teóricas que integrarán todo lo visto hasta ahora. ¡Así que es importante que hayan leído la teoría de las guías!

Segundo, vamos a tener 4 desafíos prácticos: uno en el que tendrás que hacer una consulta en SQL y tres en los que tendrás que trabajar en Java, con funciones y resultados esperados. Esto significa que le pasaremos un dato a la función de Java y, según ese dato, la función devolverá otro dato esperado. Igualmente, no te preocupes porque quedará más claro cuando llegues a esa parte.

¡Mucha suerte!

[01\_Collections Framework](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00001_Collections%20Framework)

¿Cuál de estas clases **NO** es parte del Collections Framework? Examen EGG Pregunta Real

LinkedList

Iterator

Collection

Matriz

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

POO Examen EGG Pregunta Real

¿Qué es un constructor?

Es un método que permite instanciar un objeto de la clase donde se encuentra

Un método void

Un método que nos define qué atributos tiene la clase

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

POO Examen EGG Pregunta Real

¿Cómo accedo a los atributos de un objeto?

Con el símbolo ! luego del nombre del objeto

Con el símbolo \_ luego del nombre del objeto

Con el símbolo . luego del nombre del objeto

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[02\_Static](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00002_Static)

¿Cuál de los siguientes se crean anteponiendo la palabra static a su declaración?

a) Atributos de objeto

b) Atributos de clase estáticos

c)Variables finales

d) Ninguna de las anteriores

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[03\_Relaciones](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00003_Relaciones)

¿Cuál de las siguientes relaciones es la más simple?

Asociación

Agregación

Composición

Ninguna de las anteriores

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[Relaciones](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00003_Relaciones) Examen EGG Pregunta Real

¿Cuál de las siguientes no es una relación de clases?

Navegación

Composición

Asociación

Agregación

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[04\_UML](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00004_UML) Examen EGG Pregunta Real

¿Qué significa el acrónimo UML?

Ninguna de las anteriores

Lenguaje de Modelo Único

Lenguaje de Modelado Unificado

Lenguaje de Mapas sobre Uniones

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[05\_Herencia](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00005_Herencia)

Para decir que una clase hereda a otra se usa la palabra...

Implements

Inherits

Extends

Ninguna de las anteriores

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[06\_Representación de clases](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00006_Representaci%C3%B3n%20de%20clases)

Las clases en UML se representan con un...

Triángulo

Rectángulo

Círculo

Ninguna de las anteriores

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[07\_Herencia en Java](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00007_Herencia%20en%20Java) Examen EGG Pregunta Real

¿Cuál de estas **NO** es un tipo de herencia en Java? (Cuál de estas herencias no es soportada en java?)

Herencia Multinivel

Herencia Jerárquica

Herencia Simple

Herencia Múltiple

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[08\_Excepciones y errores](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00008_Excepciones%20y%20errores)

La clase padre de todas las excepciones y errores es la clase...

Exception

Error

Throwable

Ninguna de las anteriores

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[09\_LIKE](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00009_LIKE)

En una cláusula LIKE, ¿cómo se obtienen todos los nombres de personas que comienzan con "Juan"?

LIKE "Juan%"

LIKE "Juan\*"

LIKE "Juan$"

LIKE "Juan&"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[10\_JDBC](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00010_JDBC)

¿Cuál es la clase encargada de obtener los datos de la base de datos en JDBC?

Statement

ResultSet

Connection

DriverManager

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[11\_JPA](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00011_JPA)

¿Cuál de estas clases **NO** es parte de la arquitectura de JPA?

EntityManager

PersistenceUnit

EntityManagerFactory

Entity

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[12\_Tipos de relación](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00012_Tipos%20de%20relaci%C3%B3n)-JPA

Elegir qué anotación se aplica a la siguiente relación: "un Perro pertenece a un Dueño":

"@OneToOne"

"@ManyToMany"

"@ManyToOne"

"@OneToMany"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## [13\_GitHub](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00013_GitHub)

¿Qué comando usamos para clonar un repositorio de GitHub?

Git pull

Git clone

Git download

Git remote

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[14\_CSS](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00014_CSS)

En relación al atributo class de CSS, ¿cuál de las siguientes afirmaciones es verdadera?

No se puede repetir en dos elementos

Es reutilizable

Debe ir siempre

Ninguna es cierta

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[15\_HTML](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00015_HTML)

En HTML, ¿qué atributo debemos ponerle a un input para tener texto oculto?

Ejemplo de texto oculto: \*\*\*\*\*\*\*

Hidden

HideText

Password

HiddenValue

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[16\_HTTP](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00016_HTTP)

¿Qué indica el código HTTP 301?

OK, petición procesada correctamente.

Indica al browser que visite otra dirección.

Acceso prohibido, por falta de permisos.

No encontrado, cuando el documento no existe.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[17\_Arquitectura](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00017_Arquitectura)/Spring

¿Durante Spring qué arquitectura implementamos?

XML

UML

MVC

EVC

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[18\_Thymeleaf](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00018_Thymeleaf)-Spring

¿Qué es Thymeleaf?

Un motor de vistas

Un lenguaje de programación basado en Java

Un motor de plantillas

Ninguna de las anteriores

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[19\_Anotaciones](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00019_Anotaciones)-JPA

¿Cuál de estas **NO** es una anotación de Spring?

"@Qualifier"

"@AutoWired"

"@Service"

"@Entity"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[20\_Maven](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00020_Maven) Examen EGG Pregunta Real

¿Cómo se llama el archivo que usa Maven para construir sus dependencias?

Application-context.xml

Application-propperties.yml

POM.xml

Dependencies-propperties.xml

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Parte práctica

[21\_Query](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00021_Query)- MySQL

Seleccionar la cantidad de países donde su población en 2020 esté entre 30 millones y 50 millones.

CREATE TABLE paises (id\_paises INTEGER PRIMARY KEY, nombre\_paises TEXT, codigo\_paises TEXT, poblacion\_2018 INT, poblacion\_2019 INT, poblacion\_2020 INT); INSERT INTO paises VALUES(32, "Argentina", "AR", 44000000, 45000000, 45000000),(858, "Uruguay", "UY", 3000000, 3000000, 3000000),(152, "Chile", "CL", 18000000, 19000000, 19000000),(600, "Paraguay", "PY", 6000000, 7000000, 7000000),(76, "Brasil", "BR", 209000000, 211000000, 212000000),(68, "Bolivia", "BO", 11000000, 11000000, 12000000),(170, "Colombia", "CO", 49000000, 50000000, 50000000),(862, "Venezuela", "VE", 29000000, 29000000, 30000000);

Podés usar la tabla paises que está en el esquema por defecto. Por ejemplo:

SELECT \*

FROM paises;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[22\_El último superviviente](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00022_El%20%C3%BAltimo%20superviviente)

class Examen {

public String ultimoSuperviviente(String palabra, int[] numeros) {

// Acá escribí la lógica necesaria

}

}

Tenemos una cadena de letras y un arreglo vector de números.

* Los numeros indican la posición, en orden, de las letras que deben ser removidas, desde el principio del vector.
* Luego de remover una letra, el tamaño de la cadena disminuye (no quedan espacios vacíos).

Definí una función que devuelva la última letra superviviente.

Por ejemplo, si el String original fuera "zbk" y el array [0, 1], el paso a paso sería así:

str = "zbk", arr = [0, 1]

str = "bk", arr = [1]

str = "b", arr = []

return "b";

Resultados esperados:

* *Si se ingresan el String****"ASD"****y el vector****[1, 0]****, deberá devolver la letra****"D"****.*
* *Si se ingresan el String****"Burger"****y el vector****[4,3,2,1,0]****, deberá devolver la letra****"r"****.*
* *Si se ingresan el String****"Milanga"****y el vector****[6,4,2,1]****, deberá devolver****"Mag"****.*
* *Si se ingresan el String****"Vaso"****y el vector****[]****, deberá devolver****"Vaso"****.*
* *Si se ingresan el String****"ExamenFinal"****y el vector****[10,9,8,7,6,5,4,3,2,1]****, deberá devolver la letra****"E"****.*

Examen pr;

@Before

public void before() {

pr = new Examen();

}

@Test

public void ASD\_con\_vector\_1\_0\_devuelve\_D() {

Assert.assertEquals("D", pr.ultimoSuperviviente("ASD", new int[] {1, 0}));

}

@Test

public void Burger\_con\_vector\_4\_3\_2\_1\_0\_devuelve\_r() {

Assert.assertEquals("r", pr.ultimoSuperviviente("Burger", new int[] {4,3,2,1,0}));

}

@Test

public void Milanga\_con\_vector\_6\_4\_2\_1\_devuelve\_Mag() {

Assert.assertEquals("Mag", pr.ultimoSuperviviente("Milanga", new int[] {6,4,2,1}));

}

@Test

public void Vaso\_con\_vector\_vacio\_devuelve\_Vaso() {

Assert.assertEquals("Vaso", pr.ultimoSuperviviente("Vaso", new int[] {}));

}

@Test

public void ExamenFinal\_con\_vector\_10\_9\_8\_7\_6\_5\_4\_3\_2\_1\_devuelve\_E() {

Assert.assertEquals("E", pr.ultimoSuperviviente("ExamenFinal", new int[] {10,9,8,7,6,5,4,3,2,1}));

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[23\_Funciones heredadas](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00023_Funciones%20heredadas)

class A {

protected int func(int a) {

return ;

}

}

class B {

protected int func(int a) {

return ;

}

}

class C {

protected int func(int a) {

return ;

}

}

class D {

protected int func(int a) {

return ;

}

}

Te dejamos cuatro clases definidas: A, B, C y D. Cada una tendrá la implementación y sobreescritura de func(int a).

D deberá heredar de C, C de B, y B de A. A su vez, se define func(int a) de la siguiente forma:

* En la clase A, func() multiplica un valor pasado por parámetro por 2.
* En la clase B, func() multiplica el valor a pasado por parámetro por lo que devuelva func(a) en la clase padre A.
* En la clase C, func() multiplica el valor a pasado por parámetro por lo que devuelva func(a) en la clase padre B.
* En la clase D, func() multiplica el valor a pasado por parámetro por lo que devuelva func(a) en la clase padre C.

Modificá las 4 clases para que retornen el producto correspondiente según lo dicho anteriormente.

Por ejemplo:

D d = new D();

System.out.println(d.func(3)); //162

Resultados esperados:

* *Si ingresa el número****3****en la clase D, el valor final sería****162****.*
* *Si ingresa el número****4****en la clase D, el valor final sería****512****.*
* *Si ingresa el número****10****en la clase D, el valor final sería****20000****.*

Recordá el uso de extends y super. También se sugiere hacer un diagrama de herencia para facilitar el entendimiento. 😉

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[24\_Excepciones](https://github.com/MumukiCertificaciones/mumuki-guia-java-examen-integrador-egg-2/tree/master/00024_Excepciones)

class MiCalculadora {

public double potencia(double n, double p) throws Exception {

// Acá escribí la lógica necesaria

}

}

Se te pide computar la potencia de un número implementando una calculadora.

Vamos a usar la clase MiCalculadora que consiste de un solo método: potencia(int,int). Este método toma dos enteros n y p como parámetros y devuelve la p-potencia de n (es decir, n elevado a la p).

Observaciones:

* Si *n* o *p* es negativo, el método debe tirar una excepción que diga: "n o p no pueden ser negativos".
* Si *n* o *p* son 0 debe tirar otra excepción que diga "n o p no pueden ser 0".

Completá la función de la clase MiCalculadora para que retorne el valor del cálculo de la potencia y las excepciones correspondientes según lo dicho anteriormente.

Resultados esperados:

* *Si ingresa el****n = 8****y****p = 2****, devolverá****64****.*
* *Si ingresa el****n = 0****y****p = 2****, devolverá****"n o p no pueden ser 0****.*
* *Si ingresa el****n = -8****y****p = 2****, devolverá****"n o p no pueden ser negativos"****.*

MiCalculadora pr;

@Before

public void before() {

pr = new MiCalculadora();

}

@Test

public void si\_n\_es\_0\_lanza\_excepcion() throws Exception{

try{

pr.potencia(0,2);

Assert.fail("Debería haber lanzado una excepción");

}catch(Exception e){

Assert.assertEquals("n o p no pueden ser cero", e.getMessage());

}

}

@Test

public void si\_p\_es\_0\_lanza\_excepcion() throws Exception{

try{

pr.potencia(3,0);

Assert.fail("Debería haber lanzado una excepción");

}catch(Exception e){

Assert.assertEquals("n o p no pueden ser cero", e.getMessage());

}

}

@Test

public void si\_n\_es\_negativo\_lanza\_excepcion() throws Exception{

try{

pr.potencia(-5,2);

Assert.fail("Debería haber lanzado una excepción");

}catch(Exception e){

Assert.assertEquals("n o p no pueden ser negativos", e.getMessage());

}

}

@Test

public void si\_p\_es\_negativo\_lanza\_excepcion() throws Exception{

try{

pr.potencia(3,-4);

Assert.fail("Debería haber lanzado una excepción");

}catch(Exception e){

Assert.assertEquals("n o p no pueden ser negativos", e.getMessage());

}

}

@Test

public void para\_8\_a\_la\_2\_devuelve\_64() throws Exception {

Assert.assertEquals(64, pr.potencia(8, 2), 0);

}

@Test

public void para\_2\_a\_la\_10\_devuelve\_1024() throws Exception {

Assert.assertEquals(1024, pr.potencia(2, 10), 0);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*