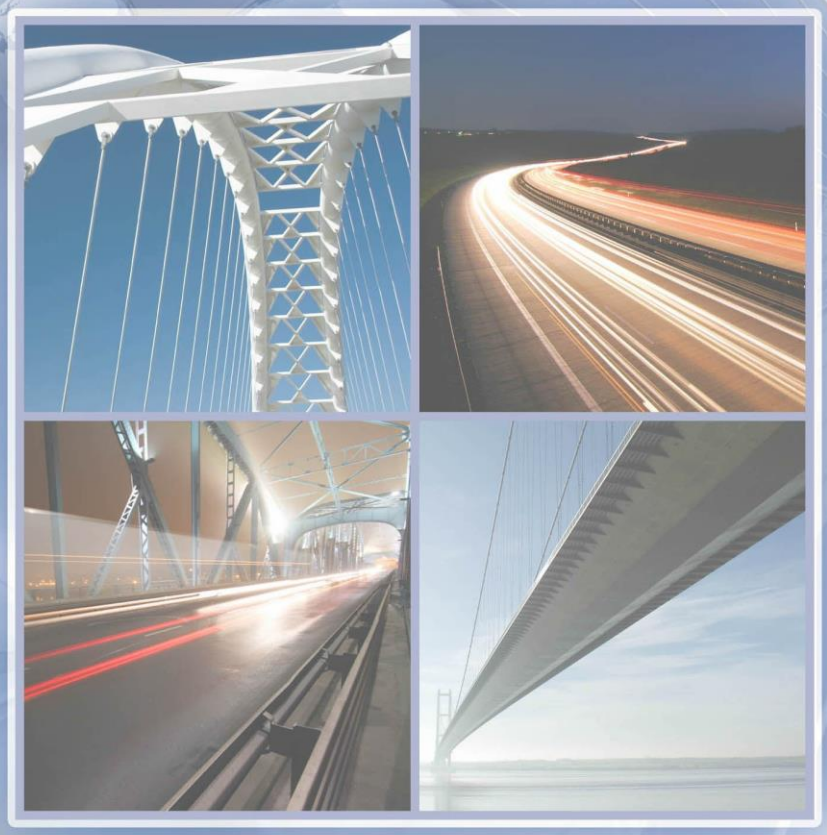
LATINIA



Test de Evaluación JAVA SE/EE

Enero de 2023

Software Infrastructure Innovation anywhere

A continuación se presenta una lista de ejercicios que el candidato deberá resolver, presentando el código fuente y documentando la solución empleada en caso de ser necesario.

Los ejercicios basados en JAVA S.E. deben incluir test unitarios (JUnit, TestNG, etc.) que verifiquen su correcto funcionamiento.

Se valorará positivamente que los ejercicios basados en Java EE empleen EJB 3.0 e incluyan tests unitarios que verifiquen su correcto funcionamiento.

Para la **entrega** de los ejercicios, se creará un archivo ZIP para cada uno: ejercicio1.zip, ejercicio2.zip, etc. Cada archivo ZIP contendrá una carpeta "proyecto" con la implementación de la solución, y un documento PDF (ejercicio1.pdf, ejercicio2.pdf, etc.) explicando la solución adoptada.

Ejercicio 1:

1) Definir una clase abstracta Persona con las siguientes características:
    a. Atributos: nombre y edad.
    b. Setters y getters para todos los atributos.
    c. Método abstracto "getSaludo". Las diferentes implementaciones de este método devolverán un String con el contenido "Hola. Me llamo <nombre> y soy un/a <tipo_persona> de <edad> años."
2) Definir cuatro clases heredadas de Persona: Niño, Niña, Hombre y Mujer.
3) Definir una última clase Main.java que será el punto de entrada del programa, y en ella se realizarán las siguientes acciones:
    a. Construir un Vector de personas con los datos del fichero "personas.csv"
    b. Iterar el vector para que todas las personas saluden.

- El fichero "personas.csv" contendrá una persona en cada línea. Una línea se compone de los campos: nombre,sexo,edad
- El sexo podrá ser "M" (masculino) o "F"(femenino).
- Si una persona es mayor de 18 años se considerará un hombre (o mujer), y si es menor de 18 será un niño o niña.

Ejemplo de fichero "personas.csv":

```
Ana,F,35
Pedro,M,42
Juan,M,39
Marc,M,8
Andrea,F,14
```

El fichero anterior debe producir la salida:

```
Hola. Me llamo Ana y soy una mujer de 35 años.
Hola. Me llamo Pedro y soy un hombre de 42 años.
Hola. Me llamo Juan y soy un hombre de 39 años.
Hola. Me llamo Marc y soy un niño de 8 años.
Hola. Me llamo Andrea y soy una niña de 14 años.
```

Ejercicio 2:

1) Desarrollar un servlet que busque en una base de datos una lista de nombres de persona e imprima en pantalla sus datos si se encuentra la persona.
2) La tabla sobre la que hay que buscar tendrá la siguientes estructura:

Tabla: PERSONAS	
Dni	VARCHAR2(9)
nombre	VARCHAR2(40)
edad	NUMBER

3) El datasource a usar estará publicado en el JNDI, en la ubicación: `jdbc/dsPersonas`
4) La lista de nombres, separados por comas, la recibirá el servlet vía el parámetro "lista".

Ejemplo de invocación de servlet:

http://host:port/evaluacion/busca_personas?lista=marta,rafael,alberto,pilar

Ejemplo de salida en el navegador:

Rafael,123456789,30
Pilar,123456789,23

Ejercicio 3:

1) Realizar un programa que invoque al servlet del ejercicio anterior y muestre el resultado devuelto.

Ejercicio 4:

1) Dada la siguiente clase Jugador:

```
class Jugador{
    String nombre;
    int puntuacion;

    Jugador(String nombre, int puntuacion){
        this.nombre = nombre;
        this.puntuacion = puntuacion;
    }
}
```

2) Escribe un código que cree un *array* de 20 jugadores asignando puntuaciones y nombres de forma aleatoria. Los nombres se obtendrán de un array de 15 nombres, de tipo:

```
String[] nombres = new String[]{"Juan", "Marta", "Lucas",
"Miriam", "Pedro", "Raul", "Cristina", …};
```

3) Ordenar el array en función de la puntuación de forma descendente. Si dos jugadores tienen la misma puntuación, ordenarlos alfabéticamente de forma ascendente.
4) Condiciones del ejercicio:
    a. La puntuación debe estar entre 0 y 1000, y ser divisible por 10.
5) Imprimir en pantalla la lista ordenada de jugadores y puntuaciones, por ejemplo:

    Miriam 13412
    Raul, 5139
    Pedro, 1234
    Cristina, 500
    Lucas, 500
    Etc.

Ejercicio 5:

Queremos implementar un clasificador de residuos que viajan en una cinta transportadora, y separar papel, cristal y metal. Los residuos llegaran por una cola JMS llamada entradaResiduos, y deberán enviarse a las colas salidaPapel, salidaCristal o salidaMetal según sea necesario.

La recepción de residuos se hará implementando un MDB, llamado ReciclajeBean, que recibirá los residuos en el cuerpo de un mensaje JMS TextMessage. Si el mensaje no es del tipo TextMessage hay que descartarlo. Además, para asegurarnos de que solamente obtenemos residuos de la cinta transportadora, los mensajes JMS correspondientes llevaran la propiedad JMS: *tipo='residuo'*.

El cuerpo del mensaje JMS nos indicará qué residuo es, conteniendo el valor "papel", "cristal", o "metal". Una vez identificado el residuo hay que enviarlo al Clasificador.

El Clasificador es un Bean de sesión, llamado ClasificadorBean, que expondrá un método en su interfaz local:

```
void clasifica(String residuo);
```

Este método debe recibir los residuos ("papel", "cristal" o "metal") y enviarlos a la cola JMS correspondiente (salidaPapel, salidaCristal o salidaMetal).