

Preguntas APX y java examen 02-Septiembre-2024

APX

1. Las dependencias circulares son permitidas...* Respuesta correcta Están absolutamente prohibidas
2. Son dos de los protocolos físicos que APX admite en una solicitud. JMS y REST
3. En caso de que una transacción deba invocar a otra, solo se puede realizar de forma Asincrónica
4. ¿Cuántas veces es posible invocar Mainframe? 2
5. Una transacción o librería, dentro de su lógica, debe evitar invocar más de ____ librerías 9
6. No se debe manejar el objeto Datasource directamente
7. En un Batch, ¿Cuál es el step donde el desarrollador introducirá la lógica empresarial necesaria? TASK
8. Implementación de PowerCurve en APX? Verdadero falso?
9. Los componentes de seguridad lógica que dan servicio a la arquitectura no están acoplados a la Arquitectura APX. Verdadero
10. DEBUG, sirve para información de muy bajo nivel solo útil para el debug de la aplicación
11. No es una característica de the blob
12. En APX, cómo es que se tiene el acceso a datos en Oracle
 - a. JDBC
 - b. ORM
 - c. JPA
 - d. SQL
13. Componente que se encarga de encapsular toda la lógica empresarial y el acceso a datos. **Respuesta: Librería.**
14. La creación de hilos o su gestión por las aplicaciones está permitida.
 - a. Verdadero
 - b. Falso**
15. El consumo de transacción asincronica pierde el control de la ejecución y solo se puede utilizar cuando la operación realizada de forma asíncrona ____ para el proceso de negocio, ya que el consumidor de esa transacción no sabrá si ha concluido con un commt OK o con rollback. **Respuesta: No es crucial**
16. Son dos de las características ofrece la Arquitectura Batch para ampliar la funcionalidad que ofrece el framework. **Respuesta: Soporte Multi-BBDD, Reading/Writing de multiples formatos de archivo**

17. ¿Quién es el responsable de iniciar un Job?
- a. JobExecution
 - b. JobLauncher
 - c. JobInstance
 - d. JobBegginer
18. Los archivos objeto del procesamiento Batch de APX son de naturaleza estrictamente ____, y este archivo no es, en ningún caso, un mecanismo de persistencia de información a largo plazo. **Respuesta: Temporal.**
19. ¿Cuál es la ubicación de archivos temporales utilizados en Batch?
- a. /fichtemporal/datent
 - b. /fichtemcomp/datent
 - c. /temporalFiles/datent
 - d. /datentJobs/datent
20. ¿Cómo solucionamos un Blob?
- a. Mover la funcionalidad a la transacción reduciendo la carga en la librería.
 - b. Mover el comportamiento a sus librerías relacionadas, reduciendo el acoplamiento entre librerías y simplificando los mantenimientos.
 - c. Moviendo la lógica a diferentes métodos y clases dentro de la misma librería
 - d. Mover el comportamiento a DTOs relacionados, reduciendo el acoplamiento entre los componentes y simplificando los mantenimientos
21. Es una contradicción al usar el patrón CRUD
- a. Mantener todas las operaciones sobre la entidad encapsuladas en la librería
 - b. Tener varias librerías parciales para una única entidad
 - c. Centralizar en una librería de todas las operaciones básicas sobre una entidad
 - d. Tener una implementación ligera, evitando extender la lógica más allá del objetivo de la operación
22. Nivel de log que muestra información superior que permite monitorear la ejecución normal. **Respuesta: Info**
23. En una librería, ¿Cuál es el archivo en donde se define la lógica del negocio?
- a. UUAAR000Impl
 - b. UUAAR001-app-osgi.xml y UUAAR001-app.xml
 - c. UUAAR001-arq-osgi.xml y UUAAR001-arc.xml
 - d. UUAAR000Abstract
24. En primer lugar, APX recibe la solicitud en uno de los protocolos físicos que admite (HTTP, JMS, REST,). Luego, crea un identificador único para la _____ y valide si los encabezados de solicitud están bien formateados de acuerdo con el protocolo lógico definido por Arquitectura
- a. Pila de ejecución
 - b. Solicitud
 - c. Transacción

d. Ejecución

25.

Java

1. The BUILDER pattern is used to: Simplify the creation of complex objects.
2. In Java the difference between throws and throw is: throws indicates the type of exception that the method does not handle and throw an exception.
3. A method is declared to take three arguments. A program calls this method and passes only two arguments. What is the result? Compilation fails
4. Which three implementations are valid?

```
interface SampleCloseable {
    public void close() throws java.io.IOException;
}

class Test implements SampleCloseable { public void close() throws java.io.IOException { // do something }}

class Test implements SampleCloseable { public void close() throws Exception { // do something }}

class Test implements SampleCloseable { public void close() throws FileNotFoundException { // do something }}

class Test extends SampleCloseable { public void close() throws java.io.IOException { // do something }}

class Test implements SampleCloseable { public void close() { // do something }}
```

Respuesta correcta

```
class Test implements SampleCloseable { public void close() throws java.io.IOException { // do something }}
class Test implements SampleCloseable { public void close() throws FileNotFoundException { // do something }}
class Test implements SampleCloseable { public void close() { // do something }}
```

5. Which three lines will compile and output "Right on!"?

```
13. public class Speak {
14.     public static void main(String[] args) {
15.         Speak speakIT = new Tell();
16.         Tell tellIt = new Tell();
17.         speakIT.tellItLikeltis();
18.         (Truth) speakIT.tellItLikeltis();
19.         ((Truth) speakIT).tellItLikeltis();
20.         tellIt.tellItLikeltis();
21.         (Truth) tellIt.tellItLikeltis();
22.         ((Truth) tellIt).tellItLikeltis();
23.     }
24. }
```

```
class Tell extends Speak implements Truth {
    @Override
    public void tellItLikeltis() {
        System.out.println("Right on!");
    }
}
```

```
interface Truth {
    public void tellItLikeltis();
}
```

6. What changes will make this code compile? **SELECCIONA 2**

```
class X {  
    X() { }  
    private void one() { }  
}  
public class Y extends X {  
    Y() { }  
    private void two() {  
        one();  
    }  
    public static void main(String[] args) {  
        new Y().two();  
    }  
}
```

Adding the public modifier to the declaration of class X.
Adding the protected modifier to the X() constructor.
Changing the private modifier on the declaration of the one() method to protected.

Removing the Y() constructor.

7. ¿Qué imprime?

```
public class Main {  
    public static void main(String[] args) {  
        int x = 2;  
        for(;x<5;){  
            x=x+1;  
            System.out.println(x);  
        }  
    }  
}
```

8. ¿Cuál es el resultado?

```
public class Test {  
    public static void main(String[] args) {  
        int[][] array = { {0}, {0,1}, {0,2,4}, {0,3,6,9}, {0,4,8,12,16} };  
        System.out.println(array[4][1]);  
        System.out.println(array[1][4]);  
    }  
}
```

9. Which two possible outputs?

```

public class Main {
    public static void main(String[] args) throws Exception {
        doSomething();
    }
    private static void doSomething() throws Exception {
        System.out.println("Before if clause");
        if (Math.random() > 0.5) { throw new Exception();}
        System.out.println("After if clause");
    }
}

```

10. ¿Cuál es la salida?

```

public class Main {
    public static void main(String[] args) {
        int x = 2;
        if(x==2) System.out.println("A");
        else System.out.println("B");
        else System.out.println("C");
    }
}

```

11. How many times is 2 printed?

```

class Menu {
    public static void main(String[] args) {
        String[] breakfast = {"beans", "egg", "ham", "juice"};
        for (String rs : breakfast) {
            int dish = 1;
            while (dish < breakfast.length) {
                System.out.println(rs + "," + dish);
                dish++;
            }
        }
    }
}

```

Resultado: beans,2, beans,3, egg,2, egg,3, ham,2, ham,3, juice,2, juice,3
4 veces se imprime 2

12. What is the result?

```

class Person {
    String name = "No name";
    public Person (String nm) {name=nm}
}
class Employee extends Person {
    String empID = "0000";
    public Employee(String id) { empID " //18
}

```

```

}
public class EmployeeTest {
    public static void main(String[] args) {
        Employee e = new Employee("4321");
        System.out.println(e.empID);
    }
}

```

- A. 4321.
- B. 0000.
- C. An exception is thrown at runtime.
- D. Compilation fails because of an error in line 18.

13. What is the result? (No me apareció esta pregunta tal cual, pero era muy parecida)

```

class Atom {
    Atom() {System.out.print("atom ");}
}
class Rock extends Atom {
    Rock(String type) {System.out.print(type);}
}
public class Mountain extends Rock {
    Mountain(){
        super("granite ");
        new Rock("granite ");
    }
    public static void main(String[] a) {new Mountain();}
}

```

- A. Compilation fails.
- B. Atom granite.
- C. Granite granite.
- D. Atom granite granite.
- E. An exception is thrown at runtime.
- F. Atom granite atom granite.

14. What is the result?

```

import java.text.*;
public class Align {
    public static void main(String[] args) throws ParseException {
        String[] sa = {"111.234", "222.5678"};
        NumberFormat nf = NumberFormat.getInstance();
        nf.setMaximumFractionDigits(3);
        for (String s : sa) {
            System.out.println(nf.parse(s));
        }
    }
}

```

- A. 111.234 222.567

- B. 111.234 222.568
- C. 111.234 222.5678
- D. An exception is thrown at runtime

15. What is the result?

```
interface Rideable {
    String getTicket(); //No recuerdo muy bien el nombre del método, pero
    era diferente
}
public class Camel implements Rideable {
    int weight = 2;
    String getGait() {
        return mph + ", lope";
    }
    void go(int speed) {
        ++speed;
        weight++;
        int walkrate = speed * weight;
        System.out.print(walkrate + getGait());
    }
    public static void main(String[] args) {
        new Camel().go(8);
    }
}
```

16. Tema: Lambdas con predicate

17. Which two actions, used independently, will permit this class to compile?

```
import java.io.IOException;
public class Y {
    public static void main(String[] args) {
        try {
            doSomething();
        } catch (RuntimeException e) {
            System.out.println(e);
        }
    }

    static void doSomething() {
        if (Math.random() > 0.5) {
            throw new IOException();
        }
        throw new RuntimeException();
    }
}
```

- A. Adding throws IOException to the main() method signature and to the doSomething() method.
- B. Adding throws IOException to the main() method ... IOException.
- C. Adding throws IOException to the doSomething() method signature.

- D. Adding throws IOException to the main() method signature.
- E. Adding throws IOException to the doSomething() method signature and changing the catch.

18. ¿Cuál es el resultado? (Este código no es el original, si no uno similar)

```
public class LicenciaSoftware {  
    public static void main(String[] args) {  
        LocalDateTime fechaEmision = LocalDateTime.of(2024, 1, 1, 0, 0);  
        Period periodoValidez = Period.of(1, 2, 3);  
        LocalDateTime fechaVencimiento = fechaEmision.plus(periodoValidez);  
        System.out.println("La licencia vence el: " + fechaVencimiento);  
    }  
}
```

19. ¿Cuáles de las siguientes opciones son válidas?

- A. El constructor predeterminado proporcionado por el compilador puede ser llamado utilizando this().
- B. Un constructor puede ser invocado desde un método de instancia.
- C. Una variable de instancia puede ser accedida dentro de un método de clase (static).
- D. Un constructor puede ser llamado dentro de otro constructor utilizando la palabra clave this().

20. Cual es el resultado

What is the result? *

1 punto

```
class X {  
    static void m(int i) {  
        i += 7;  
    }  
    public static void main(String[] args) {  
        int i = 12;  
        m(i);  
        System.out.println(i);  
    }  
}
```

- ☐ 7
- ☒ 12
- ☐ 19
- ☐ Compilation fails.
- ☐ An exception is thrown at run time

21. 33?

What is the result if the integer value is 33? *

```
public static void main(String[] args) {
    if (value >= 0) {
        if (value != 0) {
            System.out.print("the ");
        } else {
            System.out.print("quick ");
        }
        if (value < 10) {
            System.out.print("brown ");
        }
        if (value > 30) {
            System.out.print("fox ");
        } else if (value < 50) {
            System.out.print("jumps ");
        } else if (value < 10) {
            System.out.print("over ");
        } else {
            System.out.print("the ");
        }
        if (value > 10) {
            System.out.print("lazy ");
        } else {
            System.out.print("dog ");
        }
        System.out.print("... ");
    }
}
```

- ☐ The fox jump lazy ?
- ☒ The fox lazy ?
- ☐ Quick fox over lazy ?

What is the result if you try to compile Truthy.java and then run it with assertions enabled?

```
public class Truthy{
    public static void main(String[] args){
        int x = 7;
        assert (x == 6) ? "x == 6" : "x != 6";
    }
}
```

- Truthy.java compiles and the output is x != 6
- Truthy.java compiles and an AssertionError is thrown with x != 6 as additional output.
- Truthy.java does NOT compile. ←
- Truthy.java compiles and an AssertionError is thrown with no additional output

22. APX

Las utilidades APX implementadas tanto para la arquitectura Online como * 1 punto para la arquitectura Batch son:

- ☐ Communication Manager: Se crea una librería con la capacidad de invocar a G.U.C para el envío de notificaciones. Interbackend proxy: Se crea una librería con la capacidad de invocar a los conectores IMS (para acceso a Host).
- ☒ Compresión/Descompresión: Se crea una librería con la capacidad de comprimir y descomprimir archivos en zip. Generador de documentos: Se crea una librería con la capacidad de generar documentos.

23. APX

¿Qué es APX Online? *

1 punto

- ☐ Basada en Java, diseñada y construida por BBVA con ámbito local para ser utilizada en el Banco.
- ☐ Esta plataforma contiene capacidades Front y está dirigida a aplicaciones que necesiten crear lógica de negocio sin acoplamiento con la presentación. Es un Sistema de alto rendimiento, con fácil escalabilidad y disponibilidad garantizada
- ☐ Ejecución en entornos normales tipo R2 es la misma que la arquitectura diseñada y planteada para entornos Cloud. Las versiones que se generan son no duales por lo que cambia el empaquetado de la Arquitectura y la estrategia de aprovisionamiento de componentes aplicativos.
- ☒ Habilita los servicios bancarios transaccionales fuera del entorno de mainframe, pero brinda las mismas capacidades en el mundo distribuido, reduce los costos generales del sistema, se basa en tecnologías de código abierto, permite a los desarrolladores aprovechar todos los servicios extendidos provistos en la plataforma para construir servicios lo más rápido posible

24. APX

Definición del singleton

Diferencias entre clases abstractas e interfaces

Condicionales con if y varios else

Multihilos