Temario ASO - Parte 1

Código de colores:

- Azul: la respuesta que encontré.
- Verde: es la opción más probable.
- Amarillo: puede ser.
- Rojo: no es una opción válida.

Sólo en algunas preguntas pondré todos los incisos, aquellas que requieran una mayor justificación.

1. ¿Cuál es el comando utilizado para deshacer el último commit en git?
   a. `git reset` ← deshace los cambios en el directorio actual y regresa a un commit específico mientras descarta los commits hechos después de dicho commit específico.
   b. `git revert` ← revierte los cambios introducidos por el commit original creando un nuevo commit con el contenido que se quiere invertir. Es la correcta ya que la pregunta especifica el último commit.
   c. `git amend` ← no es un comando, es un *flag* que acompaña al comando `git commit` que modifica el commit más reciente; combina los cambios actuales con los del commit anterior o para cambiar el mensaje del commit sin modificar el *snapshot*.
   d. `git checkout` ← permite cambiarse entre ramas.

2. ¿Cuál es la diferencia entre una clase abstracta y una interfaz en Java 8?

   Una clase abstracta tiene estado mutable y constructores, cosa que la interfaz no tiene. Además, una clase puede implementar varias interfaces pero no puede extender varias clases abstractas.

   a. Una clase abstracta puede contener implementaciones de métodos, mientras que una interfaz no puede. ← Si consideramos los métodos *default*, entonces es incorrecta, lo cual es posible porque la pregunta especifica Java 8.
   b. Una clase abstracta puede contener variables de instancia, mientras que una interfaz no puede. ← Es una buena opción pero la última es mejor.
   c.
   d.
   e. Una interfaz puede ser implementada por múltiples clases, mientras que una clase abstracta sólo puede ser subclasificada. ← Es la más acertada.

3. De los siguientes, ¿qué tipos de declaraciones se debe usar para contar la cantidad de monedas de 5 centavos en una matriz de cadenas de varias monedas? Elige todas las correctas.

   a. Conditional ← Si nos referimos a las declaraciones condicionales (if, switch, etc.), podría usarse para preguntar si la moneda es de 5 centavos.

   b. Assertion ← Ésto tiene que ver con las excepciones, y para los fines del programa descrito parece ser equivocada.

   c. Assignment ← Entendiendo ésto como el hecho de declarar una variable de referencia que apunte a un objeto, entonces ésto es vital ya que necesitamos declarar en una variable la matriz.

   d. Iteration ← Es importante para iterar la matriz.

4. ¿Qué es un archivo `.jar` en Java?

   Es un formato de archivo comprimido que nos permite empaquetar múltiples archivos `.class` y recursos en un sólo archivo.

   a.

   b.

   c. Un archivo que contiene una clase Java compilada. ← Es la más acertada

5. ¿Qué es la sobrecarga (*overloading*) de métodos de Java?

   Es el término que se refiere a cuando en una clase a un método se le dan varias definiciones donde se diferencian en el número de parámetros que reciben y en el tipo que regresa.

   a.

   b.

   c. Cuando un método tiene múltiples definiciones con diferentes tipos de cantidades de parámetros. ← Es la más acertada.

6. ¿Cuál es la diferencia entre un `ArrayList` y un `LinkedList` en Java?

   Un `ArrayList` es una implementación de `List` donde los elementos tienen un índice. Visto de otra forma, es un arreglo de tamaño dinámico. Por otro lado, `LinkedList` es una lista ligada donde cada elemento tiene una referencia a su elemento anterior y el siguiente.

   1. `ArrayList` es más rápido que `LinkedList` para agregar y eliminar elementos. ← Falso: agregar un elemento en `ArrayList` tiene complejidad lineal, mientras que en `LinkedList` tiene complejidad constante.

   2. `ArrayList` es más eficiente en el uso de memoria que `LinkedList`. ← Falso: el uso de memoria es mejor en `LinkedList` ya que no desperdicia memoria, mientras que en `ArrayList` puede ocupar más memoria de la que necesita.

   3. `LinkedList` es una clase abstracta mientras que `ArrayList` es una clase concreta. ← Falso: ambas son clases concretas.

4. `LinkedList` es más rápido que `ArrayList` para acceder a elementos aleatorios. ← Falso: buscar un elemento en `LinkedList` tiene complejidad lineal, mientras que `ArrayList` tiene tiempo constante ya que cada elemento tiene un índice

7. ¿Cuándo se debe usar un bloque *finally* en una declaración try regular (no una prueba con recursos)?

   a. Cuando no hay bloques catch en una declaración try. ← Es la más coherente.

8. ¿Cuál es el propósito principal de los test unitarios?

   Verificar que ciertas porciones del código se comporten como es esperado.

   a.

   b.

   c.

   d. Asegurar la calidad del software. ← Es la más acertada

9. Selecciona la respuesta correcta con respecto al resultado del bloque de código.

```
public class Test3 {
    public static void main(String[] args) {
            String cad1 = "hola";
            String cad2 = new String("hola");
            String cad3 = "hola";

            if (cad1 == cad2)
                    System.out.println("cad1 es igual a cad2");
            else System.out.println("cad1 diferente a cad2");

            if (cad1 == cad3)
                    System.out.println("cad1 es igual a cad3");
            else
                    System.out.println("cad1 diferente a cad3");
        }
}
```

   a. cad1 diferente a cad2

      cad1 es igual a cad3 ← Es la salida correcta.

10. ¿Cuál es la salida al ejecutar el siguiente código?}

```
01. class Mammal {
02.         public Mammal(int age) {
03.                 System.out.println("Mammal");
```

```
04.         }
05. }
06. public class Platypus extends Mammal {
07.         public Platypus() {
08.                 System.out.println("Platypus");
09.         }
10.
11.         public static void main(String[] args) {
12.                 new Mammal(5);
13.         }
14. }
```

Hay un error de compilación en la línea 7 porque el constructor de `Platypus` no llama al constructor de `Mammal`.

a.

b.

c. El código no compila en la línea 8 ← Es la que más se acerca.

11. ¿Cómo se manejan las excepciones en Java?

De manera general, con el uso de bloques try-catch, declaraciones *throw* y los bloques *finally*.

a. Con la instrucción try-catch ← Es la única coherente.

12. ¿La anotación `@Ignore` es usada para omitir un test, por lo que no se ejecuta?

La anotación `@Ignore` en JUnit se utiliza para ignorar una prueba unitaria en particular.

a. Verdadero

b. Falso

13. ¿Cuál es el resultado de compilar y ejecutar el siguiente código?

```
01. public class Tester {
02.     static {
03.             int x = 3;
04.     }
05.     static int x;
06.     public static void main(String[] args) {
07.             x--;
08.             System.out.println(x);
09.     }
10. }
```

Dado que `x = 3` se encuentra dentro de un bloque estático, no está al alcance del método `main`, por lo cual no la tomaremos en cuenta. En cuanto a `static int x`, es inicializado con el valor 0;

ahora bien, el método `main` es el que toma en cuenta y le reduce su valor una unidad. Por lo tanto, el método `main` imprime -1.

a. Error de compilación en la línea 7, `x` no se inicializa. ← cualquier variable de tipo `int` por defecto se inicializa por defecto con 0.

b. -1

14. ¿Qué es un operador de *short circuit*?

Los operadores son `||` y `&&`. A diferencia de los operadores `|` y `&`, éstos no hacen la operación del lado derecho si encuentran una condición `true` o `false` respectivamente del lado izquierdo.

a. Sirve para realizar más eficientes las operaciones condicionales evitando ejecutar operaciones si estas ya no son necesarias.

15. ¿Qué es el patrón de diseño DAO y cómo se implementa en Java?

Es un patrón que propone separar por completo la lógica de negocio de la lógica para acceder a los datos. Concretamente, proporciona los métodos necesarios para insertar, actualizar, borrar y consultar la información. Sólo se utiliza para interactuar con la fuente de datos.

a. El patrón de diseño DAO es un patrón que se utiliza para abstraer la capa de acceso a datos en una aplicación. Se puede implementar en Java utilizando interfaces y clases concretas. ← Es la más acertada.

16. ¿Qué es un endpoint en una API REST?

Es la URL que recibe las solicitudes de los clientes que recibe una petición para acceder a una API y, posteriormente, obtener una respuesta.

a. Un endpoint es la URL que se utiliza para acceder a una API REST.

17. ¿Qué hace el siguiente programa?

```
public class Palabra {
    public static void main(String[] args) {
        String sPalabra = "palabra";
        int inc = 0;
        int des = sPalabra.length() - 1;
        boolean bError = false;
        while ((inc < des) && (!bError)) {
            if (sPalabra.charAt(inc) == sPalabra.charAt(des)) {
                inc++;
                des--;
            } else {
                bError = true;
            }
        }
```

```
    }
}
```

El método `main` inicializa una cadena `sPalabra = "palabra"`, un entero `inc = 0` ,otro entero `des = 6` (la cantidad de caracteres en `sPalabra` menos 1) y un booleano `bError = false`. El código dentro del ciclo *while* se ejecutará siempre y cuando `bError` sea falso e `inc` sea menor a `des`; y lo que pasa dentro del ciclo es que se compara el primer con el último caracter: en caso de que coincidan comparará el siguiente con el penúltimo y así sucesivamente. Por otro lado, si no son iguales, entonces hará el booleano `bError = true`. En resumen, comprobará si la palabra es palíndroma, aunque debería dar una salida que le muestre al usuario el resultado.

a. El programa no compila.

b. Cuenta las letras que hay en una palabra.

c. Verifica si una palabra es un palíndromo.

18. ¿Cuál de las siguientes opciones son verdaderas? Elija todas las correctas.

    a. Java es un lenguaje orientado a objetos. ← Considerando el paquete Functional y los Streams, además de que se le pueden poner etiquetas a ciertos bloques de código (con lo cual ya entramos en el terreno procedimental), Java no es puramente orientado a objetos, pero de manera popular se le etiqueta de esa forma.

    b. El código Java compilado en Windows puede ejecutarse en Linux. ← Esto se puede gracias a la JVM.

    c. Java permite la sobrecarga de operadores. ← Sólo se permite la sobrecarga de constructores y métodos.

    d. Java es un lenguaje de programación funcional. ← Si se considera Functional, entonces sería una respuesta válida.

    e. Java es un lenguaje procedimental. ← Tiene cosas de un lenguaje procedimental pero no es popularmente conocido por eso.

    f. Java tiene punteros a ubicaciones específicas en la memoria. ← Técnicamente una variable de referencia apunta a una dirección en memoria que contiene el objeto, pero no es un puntero como tal como el de C++.

19. ¿Qué es Maven y para qué se utiliza en el desarrollo de aplicaciones?

    Es una herramienta de compresión y de gestión de proyectos. Es usada para la construcción y mantenimiento de proyectos programados en Java.

    a.

    b.

    c.

    d. Maven es una herramienta de gestión de dependencias. Se utiliza en el desarrollo de aplicaciones en el proyecto. ← Es la más acertada.

20. ¿Cuál de lo siguiente es cierto? Elige todas las correctas.

    a. `javac` compila un archivo `.java` en un archivo `.bytecode`. ← La pregunta sería afirmativa si no tuviera punto `.bytecode`.

    b. Java toma el nombre del archivo `.bytecode` como parámetro.

    c. `javac` compila un archivo `.java` en un archivo `.class`.

    d. Java toma el nombre de la clase como parámetro. ← Es probable que sea.

    e. Java toma el nombre del archivo `.class` como parámetro.

    f. `javac` compila un archivo `.class` como archivo Java.

21. ¿Qué es Git y cuáles son algunos de sus comandos básicos?

    Es un sistema de control de versiones distribuido (DCVS abreviado por sus siglas en inglés). Almacena archivos y registra los cambios hechos en un repositorio.

    a.

    b.

    c. Git es un sistema de control de versiones. Algunos de sus comandos básicos incluyen `commit` y `push`.

22. Dados los siguientes segmentos de código, ¿qué respuesta no es una implementación de Java válida?

    a.
    ```
    int variableA = 10;
    float variableB = 10.5;
    int variableC = variableA + variableB;
    ```

    Esta es la opción correcta porque la suma entre un float y un int no cabe en un int.

    b.
    ```
    byte variableA = 10;
    double variableB = 10.5f;
    double variableC = variableA + variableB ;
    ```

    c.
    ```
    byte variableA = 10;
    float variableB = 10.5f;
    float variableC = variableA + variableB;
    ```

23. ¿Qué escenario es el mejor uso de una excepción?

    a. La computadora se incendió. ← No existe una excepción implementada para ese caso.

    b. No sabe cómo codificar un método. ← No se le entiende a esta respuesta.

    c. No se encuentra un elemento al buscar en una lista. ← No se me ocurre qué excepción podría lanzarse pero en cierto contexto podría ser correcta.

    d. Se pasa un parámetro inesperado a un método. ← En este caso, podría lanzarse `IllegalArgumentException`.

e. Quiere recorrer una lista. ← Si sólo se hace esa acción sin darnos contexto de la lista, no podemos sacar ninguna excepción.

24. ¿Qué es un bean en Spring?

Los beans son los objetos que forman la columna de una aplicación y son administrados por el contenedor Spring IoC.

a.

b. Una instancia de una clase que se administra por el contendedor de Spring. ← Es la más acertada.

25. Selecciona la respuesta correcta con respecto al resultado del bloque de código.

```
public class Test1 extends Concreate {
    Test1() {
        System.out.println("t ");
    }

    public static void main(String[] args) {
        new Test1();
    }
}

class Concreate extends Send {
    Concreate() {
        System.out.println("c ");
    }

    private Concreate(String s) {
    }
}

abstract class Send {
    Send() {
        System.out.println("s ");
    }
}
```

La salida es:

```
s
c
t
```

Y ningún inciso la contiene.

26. ¿Cuáles de las siguientes afirmaciones sobre el polimorfismo son verdaderas? Elige todas las correctas.

    a. Si un método toma una superclase de 3 objetos, cualquiera de esas clases puede pasarse como parámetro del método.

    b. Un método que toma un parámetro con tipo `java.lang.Object` tomará cualquier referencia. ← Sí, porque todas las clases heredan de `Object`.

    c. Una referencia a un objeto se puede convertir a una subclase de objetos en una conversión explícita. ← Sí, estamos aplicando un *cast*.

    d. Todas las excepciones de conversión se pueden detectar en tiempo de compilación. ← No, el `ClassCastException` se da en tiempo de ejecución.

    e. Al definir un método de instancia pública en la superclase, garantiza que el método específico se llamará al método en la clase principal en tiempo de ejecución.

27. Son patrones de diseño de software estructural

    - Adapter es un patrón de diseño estructural que permite la colaboración entre objetos con interfaces incompatibles.
    - Bridge es un patrón de diseño estructural que permite dividir una clase grande o un grupo de clases relacionadas en dos jerarquías separadas que pueden implementarse independientemente una de otra.
    - Proxy es un patrón de diseño estructural que permite proporcionar un sustituto para otro objeto. Controla el acceso al objeto original, lo que permite hacer algo antes o después de que la solicitud llegue al objeto original.
    - Composite es un patrón de diseño estructural que permite componer objetos en estructuras de árbol y trabajar con estas estructuras como si fueran objetos individuales.

    a.

    b. Adapter, Bridge, Proxy, Composite. ← Contiene cuatro patrones de diseño estructurales.

28. Seleccione la respuesta que considere correcta dado el siguiente bloque de código.

```
import java.util.Arrays;
import java.util.List;

public class Example {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);
        double result = numbers.stream()
                                    .maptoInt(n -> n)
                                    .average()
                                    .orElse(0);
        System.out.println(result);
```

```
    }
}
```

En resumen, el método `main` saca el promedio del `ArrayList`, el cual es 15/5 = 3.0 porque requerimos regresar un `double`.

a. 3.0

29. ¿Qué son las pruebas de integración?

Son pruebas que evalúan la forma en que interactúan y operan varios módulos de aplicaciones de software y de forma cohesiva. Cada módulo es responsable de una tarea específica y se busca comprobar que todos los módulos funciones juntos.

1.
2.
3. Pruebas que comprueban el funcionamiento de varias unidades de código juntas.

30. ¿Qué comando se utiliza para enviar los cambios confirmados en un repositorio local al remoto?

Una vez hecho el commit, el siguiente paso para que todos los cambios locales pasen al repositorio remoto es utilizar el comando `git push`.

a. `git push`

31. Seleccione la respuesta correcta, dado el siguiente bloque de código.

```
class ClassX {
    static int y = 2;

    Class(int x) {
        this();
        y = y * 2;
    }

    Class() {
        y++;
    }
}

public class Class2 extends ClassX {
    Class2() {
        super(y);
        y = y + 3;
    }

    public static void main(String[] args) {
        new Class2();
```

```
            System.out.println(y);
    }
}
```

Consideremos la variable estática `y = 2`. Ahora bien, el método `main` llama al constructor de la clase `Class2()`, y lo primero que hace es llamar al constructor de su clase padre `ClassX` que recibe un entero, el cual, a su vez, llama a su constructor que no recibe parámetros. Ahí, se le incrementa 1 a `y`, por lo cual vale 3 en ese momento. Después, el constructor que recibe un entero multiplica a `y` por 2, por lo cual su valor se duplica y queda en 6. Finalmente, ocurre la suma en el constructor de `Class2`. Por lo tanto, el método `main` imprime 9.

a. Error de compilación

b. 9

32. ¿Cuál es el comando utilizado para crear una nueva rama en Git?

    Se utiliza el comando `git branch <nombre>`, en donde debemos colocar el nombre la rama nueva.

    a. `git branch`

33. ¿Cuál es el resultado de compilar la siguiente clase?

```
public class Book {
    private int ISBN;
    private String title, author;
    private int pageCount;

    public int hashCode() {
        return ISBN;
    }

    public boolean equals(Object obj) {
        if(!(obj instanceof Book)) {
            return false;
        }

        Book other = (Book) obj;
        return this.ISBN == other.ISBN;
    }
}
```

a.

b.

c.

d. La clase compila satisfactoriamente.

34. ¿Cuál es la primer línea en fallar al compilar?

```
class Tool {
    private void repair() {} //r1
    void use(){}             //r2
}
class Hammer extends Tool {
    private int repair() { return 0; } //r3
    private void use(){}               //r4
    public void bang(){}               //r5
}
```

El error se encuentra al sobrescribir `use()` ya que no se puede restringir más el acceso de un método o variable en una subclase, lo cual se considera un error en compilación. El orden del más restrictivo al menos es `private` < `default` < `protected` < `public`.

a. r5

b. r4

35. ¿Qué es Git? (Pregunta repetida)

Es un sistema de control de versiones distribuido (DCVS abreviado por sus siglas en inglés). Almacena archivos y registra los cambios hechos en un repositorio.

a.

b.

c.

d. Una herramienta de control de versiones que se utiliza para almacenar y administrar el código fuente de un proyecto.

36. ¿Cuál de las siguiente excepciones lanza la JVM? Elija todas las correctas.

Por más información que busqué, encontré que cierto sector de la comunidad divide algunas excepciones como *JVM exceptions* y *Programmatic exceptions*, pero realmente no es un término oficial y no va ligado a las *checked* y *unchecked*. Dentro de las *Programmatic*, se encuentran aquellas que implementan `IllegalArgumentException`, en donde se encuentra `NumberFormatException` y algunas de `Exception` que no implementan `RuntimeException`, como lo es `java.io.IOException`. Por otro lado, las *JVM exceptions* son, en su mayoría, `NullPointerException`, `IndexOutOfBoundsException`, entre otras. Pero aún así es una pregunta muy ambigua que no tiene un sustento en ninguna documentación oficial.

a. `ArrayIndexOutOfBoundsException`

b. `NumberFormatException`

c. `ExceptionInInitializerError`

d. `java.io.IOException`

e. `NullPointerException`

37. ¿Cuál es el comando utilizado para fusionar una rama en Git?

    El comando `git merge` permite tomar dos líneas independientes de desarrollo creadas por `git branch` e integrarlas en una sola rama. La fusión se hará en la rama actual y al comando se le debe indicar con qué rama se quiere fusionar.

    a.

    b. `git merge`

38. ¿Qué es REST y cuál es su relación con las API web?

    La trasferencia de estado representacional o (*REpresentational State Transfer*) es un estilo de arquitectura para el diseño de software de servicios web. Se puede ver como un conjunto de reglas que buscan que diferentes sistemas de la web funcionen entre sí. Una API REST es una API web que sigue los principios de convenciones de REST para darle una funcionalidad, se ajusta a los límites de la arquitectura y permite la interacción con los servicios web.

    a.

    b.

    c.

    d. REST es una arquitectura para aplicaciones. Su relación con las API web es que utiliza para definir la estructura y funcionalidades de una API.

39. ¿Cuál es el comando utilizado para actualizar la rama local con los cambios de la rama remota en Git?

    Con el comando `git pull` "jalas" todos los cambios del repositorio remoto al repositorio local.

    a.

    b.

    c.

    d. `git pull`

40. ¿Qué es un microservicio?

    De manera general, la arquitectura de microservicios se utiliza en un proyecto para separar todas las funcionalidades que no tienen una relación estrecha en servicios independientes. Un microservicio es, entonces, una funcionalidad separada que cumple un único propósito.
    En cuanto a la pregunta, considero todas las opciones muy ambiguas.

    1. Es un componentes que se pueden desplegar de forma independiente en función múltiple, es decir, englobando endpoint que no necesariamente están relacionados. ← Podría ser ya que menciona que no están relacionados pero menciona función múltiple.
    2. Es un componente que se puede desplegar de forma independiente y que suelen ser de función única, es decir, englobando endpoint y están estrechamente relacionados. ← Tengo duda si puede ser porque menciona una estrecha relación pero también función única.

3. Es el conjunto de endpoints contenidos en múltiples desarrollos que se despliegan en conjunto y que están estrechamente relacionados. ← Un microservicio contiene endpoints, mas no es un endpoint.

4. Ninguna de las anteriores. ← Por descarte, pondría esta como respuesta ya que las anteriores las considero equivocadas

11. Dado el siguiente código:

```
public class Main {
    public static void main(String[] args) {
        int[] numeros = {1, 2, 3, 4, 5};
        int suma = 0;

        for(int i = 1; i <= numeros.length; i++) {
            suma += numeros[i];
        }

        System.out.println("La suma de los números es: "
                                            + suma);
    }
}
```

¿Este código sin compila sin errores?

El arreglo de enteros `numeros` tiene una longitud de 5 y el índice de cada uno va del 0 al 4. Así que el código contenido en el ciclo *for* tendrá un error en tiempo de ejecución al momento de querer acceder a `numeros[5]`. Concretamente, un `ArrayIndexOutOfBoundsException`. Sin embargo, el código no tiene ningún error en tiempo de compilación ya que está todo bien definido en ese aspecto.

a. Sí, compila sin errores.

12. ¿Qué método se utiliza para obtener el mensaje de una excepción en Java?

a. `getClass()` ← Regresa la clase en tiempo de compilación del objeto.

b. `printStackTrace()` ← Imprime el rastro de la pila de errores de la excepción que fue atrapada.

c. `toString()` ← Regresa una representación del objeto como cadena.

d. `getMessage()` ← Regresa una cadena que contiene un mensaje detallado de la excepción lanzada.

13. ¿Cuál de siguientes afirmaciones son verdaderas? Elije todas las correctas.

a. Puede declarar solo excepciones no comprobadas (unchecked). ← También se pueden declarar comprobadas.

b. Las excepciones en tiempo de ejecución son lo mismo que las excepciones no comprobadas. ← Sí, ya que las *unchecked* son las *Runtime Exceptions*.

c. Las excepciones en tiempo de ejecución son lo mismo que las excepciones comprobadas. ← No, contradice el punto anterior.

d. Solo puede declarar excepciones comprobadas (checked). ← También se pueden declarar no comprobadas.

e. Solo puede manejar subclases de Exception. ← No, se pueden manejar subclases de Error, como `IOError`.

14. ¿Cuál es el resultado de ejecutar el siguiente código?

```
String s = "hello";
s.toUpperCase();
System.out.println(s);
```

El *snippet* imprime `“hello”`, ya que la referencia de `s` jamás cambió.

a.

b. hello

15. ¿Cuál es el paquete de importación necesario para usar la clase `ArrayList`?

a.

b.

c.

d. `import.java.util.*`; ← No debería llevar punto entre `import` y `java`, pero es la más acertada.

16. ¿Cuál es el formato de los datos que se envían y reciben en una API REST?

a. YAML

b. XML

c. JSON

d. Todos los anteriores ← Se pueden enviar y recibir esos formatos sin problema.

17. ¿Cuál es la función del operador de doble dos puntos (::) en Java 8?

El operador es azúcar sintáctica de las lambdas en Java 8, cuyo uso más común es omitir el uso de variables que se ocupan en las lambdas y sólo definir la lambda utilizando la clase, el operador doble dos puntos y el método que regresará el tipo que pide la función de la interfaz a la que corresponde la lambda.

a.

b.

c. Se utiliza para acceder a métodos estáticos en Java 8. ← Con las lambdas es posible hacer eso pero no es el foco principal. Sin embargo, de todas las opciones, es que más probable.

d. Se utiliza para acceder a métodos no estáticos en Java 8 ← También es posible pero nuevamente, no es el foco principal.

48. ¿Qué palabra clave se utiliza para definir una excepción personalizada en Java?

    a. try

    b. throw ← Se escogió ésta por descarte.

    c. finally

    d. catch

49. ¿Cuál de los siguientes comandos elimina el directorio `target` antes de iniciar el proceso de construcción?

    El comportamiento por defecto del comando `mvn clean` es borrar el directorio `target`.

    a.

    b.

    c.

    d. `mvn clean`

50. ¿Cuál es el comando utilizado para ver el historial de cambios en Git?

    Si con cambios nos referimos a aquellos hechos con los commits, entonces `git log` muestra el historial de éstos.

    a. `git log`

51. ¿Qué es una expresión lambda en Java 8?

    Es azúcar sintáctica de una clase anónima.

    a. Es una forma de escribir una clase anónima en Java 8. ← La más acertada al término oficial.

    b. Una forma de escribir una función anónima en Java 8. ← Podría ser, pero al especificar Java 8, le da más peso a la primera opción.

52. ¿Qué muestra el siguiente código fuente por pantalla?

```
int x = 1;
switch(x) {
    case 1:
        System.out.println("Uno");
    case 2:
        System.out.println("Dos");
    case 3:
        System.out.println("Tres");
    default:
```

```
            System.out.println("Otro número");
    }
```

a.

b. Uno Dos Tres Otro número ← A pesar de que en la respuesta deben considerarse los saltos de línea, ésta es la más acertada ya que no hay `break` en cada caso del `switch`.

53. De los siguientes paquetes, ¿cuáles contienen clases para construir una interfaz gráfica. Elige todas las que correspondan.

a. `java.net` ← Permite realizar conexiones y transacciones a través de la red

b. `java.io` ← Proporciona un sistema de entrada y salida a través de transmisión de datos, serialización y el sistema de archivos.

c. `javax.swing` ← Biblioteca de clases que permite crear interfaces gráficas de usuario.

d. `java.util` ← Contiene varias clases e interfaces de utilidad general.

e. `java.awt` ← Contiene todas las clases para crear interfaces gráficas de usuario.

54. ¿Cuál de las siguientes líneas deben ir en el espacio en blanco para que el código compile?

```
public class News <___> {}
```

a. Solo `N`.

b. Solo `?`.

c. Ninguna de las anteriores.

d. `News` y `Object`.

e. `?` y `N`

f. `N`, `News` y `Object`. ← Puede ir cualquiera de éstas sin ningún problema, y aplica lo mismo para cualquier clase que utilice genéricos.

55. ¿Qué es un *stream* en Java 8 y para qué se utiliza?

Es una secuencia de elementos que soporta operaciones de agregado secuenciales y paralelas utilizado que contiene funciones del paradigma declarativo para usarse a partir de Java 8.

a.

b. Un objeto que representa una secuencia de elementos y se utiliza para procesar colecciones de forma declarativa. ← Es la más acertada.