**Revisión y Validación de conocimientos**

1. El comando "mvn install" es un comando utilizado en Maven, una herramienta de gestión de proyectos de software en Java, para construir y empaquetar proyectos de software.
2. Sí, el archivo principal de configuración de un proyecto gestionado por Maven tiene extensión XML y se llama "pom.xml". Este archivo contiene información sobre el proyecto, como sus dependencias, plugins, configuraciones, versiones, etc.
3. La estructura del archivo "pom.xml" se basa en etiquetas XML anidadas que definen diferentes elementos del proyecto. Por ejemplo, la etiqueta "groupId" define el identificador del grupo al que pertenece el proyecto, la etiqueta "artifactId" define el identificador único del proyecto, la etiqueta "version" define la versión actual del proyecto, y así sucesivamente.
4. a)"New Maven Project": Este comando se utiliza para crear un nuevo proyecto Maven en Eclipse. Al seleccionar este comando, se abrirá un asistente que guiará al usuario a través del proceso de creación del proyecto. El usuario debe proporcionar información como el nombre del proyecto, el identificador del grupo, la versión, etc.

b)"Update Project": Este comando se utiliza para actualizar el proyecto Maven existente en Eclipse. Al seleccionar este comando, se actualizarán todas las dependencias y plugins del proyecto, de acuerdo con lo especificado en el archivo "pom.xml".

c)"Run As Maven Build": Este comando se utiliza para ejecutar una tarea específica de Maven, como la compilación del código fuente, la ejecución de pruebas, la generación de informes, etc. Al seleccionar este comando, se abrirá una ventana donde el usuario puede configurar los parámetros de la tarea.

d)"Maven Clean": Este comando se utiliza para limpiar el proyecto, eliminando todos los archivos generados por Maven, como los archivos de compilación, los archivos de registro, etc.

e)"Maven Install": Este comando se utiliza para instalar el proyecto en el repositorio local de Maven. Al seleccionar este comando, se compilará el proyecto y se generará un archivo "jar" o "war", que se instalará en el repositorio local.

f)"Actualización de Maven.

1. Git init, git clone, git add, git commit , git push, git pull, git status, git log

El código no funciona ya que contiene un error. La firma del método 'mainString[] args), pero la declaración de la variable opcion en el cuerpo del método 'mainmain tiene un tipo de String.

Además, el método 'reservarboolean. Esto dará lugar a un error en tiempo de compilación.

Por lo tanto, este código no se puede ejecutar correctamente y producirá un error de compilación. Para corregir el código, se debería cambiar la firma del método 'main.

Si corregimos los errores en el código y lo ejecutamos, la mecánica del programa sería la siguiente:

El método 'main

La cadena 'opcion

Se llama al método 'reservar

El método 'reservar

Se imprime un mensaje final indicando que la reserva se ha realizado correctamente.

public class Main {

public static void main(String opcion) {

try {

boolean opcionBooleana = opcion.equalsIgnoreCase("Si");

reservar(opcionBooleana);

System.out.println("Turno fue reservada correctamente");

} catch (Exception e) {

System.out.println("Se ha producido un error al reservar el turno: " + e.getMessage());

}

}

static void reservar(boolean opcion) {

try {

if (opcion) {

System.out.println("La reserva se ha realizado correctamente.");

} else {

throw new Exception("La reserva no se ha podido realizar.");

}

} catch (Exception e) {

System.out.println("Se ha producido un error al reservar el turno: " + e.getMessage());

}

}

}

El código intenta reservar una habitación, pero no maneja adecuadamente las excepciones y puede proporcionar información engañosa al usuario.

1 try{

2 habitacion.reservar()

3 } catch (Exception ex) {}

4 System.out.printn(“La habitación fue reservada correctamente”)

1. **Responda: Si se tiene un objeto xx de clase HashMap**

d. xx.get(“algo”) si la clave existe debe retornar un objeto tipo String

1. Java . Metodos

«Localnow() es un método estático

'LocalTime.nowLocalTime de la librería estándar de Java llamada java.time.

1. Java – Arrays

En el array "numeros", el índice comienza en cero. Por lo tanto, "numeros[2]" hace referencia al tercer elemento del array, que es "16".

Entonces, el valor de "numeros[2]" es "16".

1. Junit – Unit Test

Los métodos básicos de un test unitario son:

**assertEquals(expected, actual)** - Comprueba que el valor esperado (expected) y el valor real (actual) son iguales. Si no son iguales, el test falla.

**assertTrue(boolean)** - Comprueba que la expresión booleana dada es verdadera. Si es falsa, el test falla.

**assertFalse(boolean)** - Comprueba que la expresión booleana dada es falsa. Si es verdadera, el test falla.

**assertNotNull(Object)** - Comprueba que el objeto dado no es nulo. Si el objeto es nulo, el test falla.

**assertNull(Object)** - Comprueba que el objeto dado es nulo. Si el objeto no es nulo, el test falla.

**assertSame(expected, actual)** - Comprueba que el valor esperado (expected) y el valor real (actual) son el mismo objeto. Si no son el mismo objeto, el test falla.

**assertNotSame(expected, actual)** - Comprueba que el valor esperado (expected) y el valor real (actual) no son el mismo objeto. Si son el mismo objeto, el test falla.

**@Test** - Anotación que indica que un método es un test unitario. El framework de prueba ejecutará todos los métodos anotados con @Test.

**@Before** - Anotación que indica que un método se ejecuta antes de cada método **@Test.** Se utiliza para inicializar datos que son necesarios para la ejecución de cada test.

**@After** - Anotación que indica que un método se ejecuta después de cada método **@Test**. Se utiliza para limpiar o liberar los recursos utilizados en cada test.

**@BeforeClass** - Anotación que indica que un método se ejecuta antes de que se ejecute cualquier método @Test en la clase. Se utiliza para inicializar datos que son necesarios para la ejecución de todos los tests.

**@AfterClass** - Anotación que indica que un método se ejecuta después de que se hayan ejecutado todos los métodos @Test en la clase. Se utiliza para limpiar o liberar los recursos utilizados en todos los tests.

1. Si suponemos que el atributo “tieneHambre” del objeto “hogar” es falso, entonces:

El test que tiene la sentencia Assertions.assertTrue (!hogar. tieneHambre) no pasa satisfactoriamente.

1. **Java – POO**

¿La palabra reservada “this” tiene sentido en un método “static”?

En un método static, no se puede hacer referencia a un objeto específico de la clase porque el método se ejecuta a nivel de clase, y no a nivel de instancia. Por lo tanto, la palabra reservada "this" no tiene sentido en un método static.

¿Una clase puede heredar de varias clases pero implementar una sola interfaz?

Sí, en Java una clase puede heredar de varias clases pero solo puede implementar una interfaz.

Dada la siguiente clase y el objeto “carrito1”, indique la opción correcta:

Para funcionar, en la declaración, en lugar de usar List<ItemCarrito> se debe usar ArrayList<ItemCarrito>

¿Main() que tipo de método es? ¿Para qué sirve?

Es un método estático que es el punto de entrada de cualquier programa

**Public**: El modificador de acceso "public" se utiliza para declarar campos, métodos y constructores que pueden ser accesibles desde cualquier lugar, es decir, desde dentro de la clase en sí misma, desde otras clases en el mismo paquete y desde cualquier otra clase en cualquier paquete.

**Private**: El modificador de acceso "private" se utiliza para declarar campos, métodos y constructores que solo pueden ser accesibles desde dentro de la clase en sí misma. Esto significa que no se pueden acceder a los campos o métodos privados desde otras clases, incluso si están en el mismo paquete.

**Protected**: El modificador de acceso "protected" se utiliza para declarar campos, métodos y constructores que solo pueden ser accesibles desde dentro de la clase en sí misma y desde sus subclases. Los campos y métodos protegidos no son accesibles desde otras clases en el mismo paquete, a menos que esas clases también sean subclases de la clase en la que se declaran los campos y métodos protegidos.

¿Cómo podemos crear los getter and setter de forma casi automáticas en Eclipse? ¿Para qué sirven?

En Eclipse, se pueden crear getters y setters automáticamente para los campos de una clase utilizando la función "Generate Getters and Setters". Para hacerlo, se debe seleccionar el campo que se desea crear el getter y setter, hacer clic derecho y seleccionar "Source" y luego "Generate Getters and Setters".

¿Cómo instancio una Clase?

Se abre el archivo donde desea instanciar la clase. Se escribe el nombre de la clase que desea instanciar.

Se escribe el nombre de la variable que se utilizará para referirse a la instancia de la clase.

Se utiliza el operador "new" para crear una nueva instancia de la clase.

Se asigna la instancia a la variable que se creó en el paso 3.

¿Cómo heredo de una Clase Abstracta?

Se debe crear una nueva clase: En el Package Explorer de Eclipse, se puede hacer clic derecho en el paquete donde se quiere crear la clase y seleccionar "New" y luego "Class". Escribir el nombre de la clase y seleccionar la clase abstracta de la que se quiere heredar: En el cuadro de diálogo "New Java Class", se debe escribir el nombre de la clase en el campo "Name". Luego, se debe seleccionar la clase abstracta de la que se quiere heredar en el campo "Superclass".

¿Cómo heredo una interfaz?

Crear una nueva clase o abrir la clase existente en la que se desea implementar la interfaz.

Hacer clic en el menú "Source" en la barra de menú y seleccionar "Implement Interface".

Seleccionar la interfaz que se desea implementar en el cuadro de diálogo y hacer clic en "OK".

Implementar todos los métodos de la interfaz en la clase.

¿Cómo utilizo los métodos de una clase?