**Clase n°9, ejercicios (guía ejercicio 9)**

**Autor: Vargas Diego**

Ejercicio 1.

**Maven:**

1. El comando “mvn install”, ¿para qué sirve?

Ejecuta todos los pasos anteriores (compile, test, etc) y además junta el código compilado en un archivo .jar y deja la biblioteca disponible en mi sistema para que la pueda utilizar algún otro proyecto.

2. ¿El archivo principal de configuración de un proyecto gestionado por Maven tiene extensión XML? ¿Cómo es su estructura?

Maven – pom.xml

<project>

<modelVersion>4.0.0</modelVersion>

<groupId>proyecto</groupId>

<artifactId>proyecto</artifactId>

<version>0.0.1-SNAPSHOT</version>

<build>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

3. Liste los comandos básicos de Maven para Elipse. Describa brevemente cada uno.

* ***mvn clean***: Borra todos los archivos compilados que existan.
* ***mvn compile***: Compila todo el código fuente.
* ***mvn test***: Primero compila todo el código fuente si no estuviera compilado y luego ejecuta los tests.

**Git – GitHub**

1. Liste los comandos básicos de Git. Describa brevemente cada uno.

**git init <nombre\_proyecto>:** Inicia un repositorio local con el nombre.

**git init** : Dentro de la carpeta del proyecto, inicia un repositorio en esta carpeta.

**git clone <url>** : Clona el repositorio que haya en la url.

**git init <nombre\_proyecto>:** Inicia un repositorio local con el nombre.

**git init** : Dentro de la carpeta del proyecto, inicia un repositorio en esta carpeta.

**git clone <url>** : Clona el repositorio que haya en la url.

**git merge <nombre\_de\_rama>** : Trae los cambios de la rama mencionada a la rama en la que nos encontramos.

**git pull** : Baja el historial del repo remoto e incorpora los cambios. Es necesario hacer esto antes de subir nada.

**git push <nombre\_de\_rama>** : Sube los commits que tengamos en la rama local a la rama con ese nombre del repositorio remoto. Puede ser necesaria la palabra origin.

2. Realice un diagrama de flujo donde muestre como usaría los comandos de Git para subir un proyecto desde Eclipse a GitHub.

Por Ejemplo:

Realizo el git commit -m

“mensaje”

Realizo el git push

Realizo el git add < archivo\_especifico> o git add . para varios archivos

Utilizo comando git init

Creo proyecto  
Creo repositorio

Existe proyecto?

Existe repo?

Proyecto agregado al repo(staging area)

Luego del git add

Subido al repo remoto

**Excepciones:**

Analice el código siguiente:

1. ¿Funciona o se produce una excepción?

-Se produce una excepcion

2. Explique narrativamente la mecánica en caso que funcione o Explique donde y porque se la excepción en caso que no funcione.

-Se produce una excepción debido a que la variable “opcion” es de tipo String, y cuando se llama al método reservar(boolean opcion), recibe un String, cuando espera una variable del tipo boolean.

3. En cualquier caso: Agregue los bloques de control Try-Catch para atrapar las excepciones y muestre un mensaje adecuado por pantalla.

Try{

String opcion= “si”;

Reservar(opcion);

Sout(“reserve realizada correctamente”);

}catch(Exception e){

Sout(“Lo siento, debe enviar al método reservar() una variable del tipo boolean”);

}

4. Si tengo el siguiente código y en la 2da línea se arroja una Exception, describir que sucede.

No sucede nada, ya que, el bloque catch(){} atrapa la excepción pero dentro de su bloque de código, no hay ninguna instrucción, entonces continua a la siguiente instrucción mas próxima en el programa, la cual corresponde al system.out.println y muestra el mensaje correspondiente.

**Colecciones - clase HashMap**

1. Liste los métodos para agregar, leer y borrar un objeto HashMap.

Para agregar vamos a utilizar el método “put”, para leer el método “get” y para borrar el método “remove”

2. Responda:

Si se tiene un objeto xx de clase HashMap<String, Producto>

1. La clave es un producto y el valor un texto

(falso, la clase es del tipo String y el valor es del tipo producto)

b .xx.put(“algo”,137) no debería compilar

(falso, ya que el valor es de tipo producto)

1. xx.get(“algo”) si la clave existe debe retornar un objeto tipo producto o null

(verdadero, ya que la clave es correcta y puede devolver null en el valor o del tipo de objeto en este caso)

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

(falso, debe retornar un objeto del tipo producto)

**Java – Literales, caracteres especiales, palabras reservadas**

1. De los siguientes caracteres ( < , \_ , - , @, | , / , // , \ , \\ , # , ¡ , ! , { , [ , ( , ) , ] , } , >), cual son reservados o no se deberían usar por ser carácter de escape.

Los siguientes caracteres son reservados: <,-,@,|,/,//,#, ¡ , ! , { , [ , ( , ) , ] , } , >

Los siguientes caracteres no se deberían utilizar por ser caracteres de escape: \,\\

2. Describa brevemente en qué casos de usan y muestre un ejemplo.

\n = nueva línea,el cursor de pantalla se coloca al principio de la siguiente linea ejemplo: system.out.println(“hola\n”);

\\= se utiliza para imprimir un carácter de barra inversa. Ej: systemm.out.println(“hola\\”);

**Java – Métodos**

1. ¿LocalTime.now() es un método o variable?

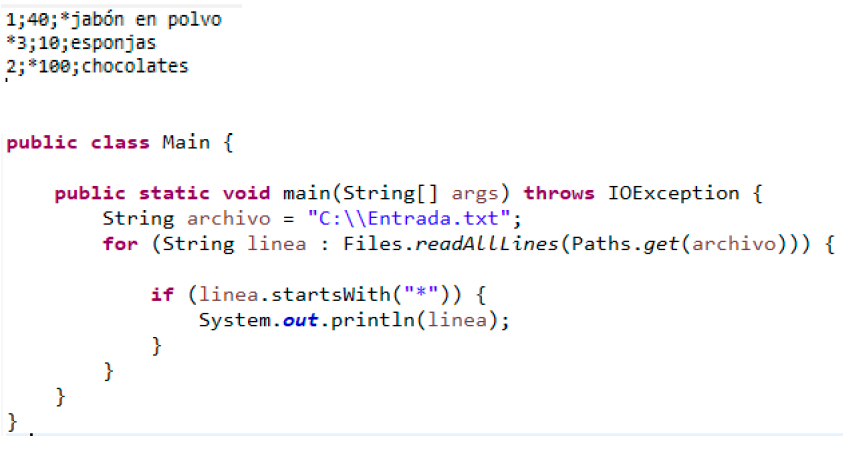
- Es un metodo

2. LocalTime.now() , de que Clase, Interfaz o Librería proviene.

Proviene de la librería time, debemos importarla a nuestra clase de la siguiente forma, import java.time.LocalTime;

**Java – Archivos, bucles de control, Métodos**

1. Dados el archivo y el código:



Explicar narrativamente lo que hace el código e indicar cuál sería la salida por pantalla.

El siguiente código realiza la lectura del archivo “entrada.txt”, lee cada línea del archivo y si la línea empieza con un asterisco la imprime por pantalla.

2. Identificar todos los Métodos usados y que a Clase pertenecen, describa brevemente.

El primer método que se utiliza, es el que pertenece al objeto “Files”, el cual se llama “readAllLines”, el cual permite leer las líneas del archivo .txt

El segundo método que se utiliza es el del objeto Paths, el metodo “get”, el cual prepara el archivo para que luego sea leído.

Por ultimo se utiliza el método “startsWith” de la clase String, el cual va a permitir imprimir la línea que comience con el carácter que se le coloca por argumento, en este caso es el “\*”;

3. Si usted está en Windows, y ejecuta un programa de Java desde C:\\Users\\SuNombre\\ y quiere abrir el archivo C:\\Users\\SuNombre\\Documentos\\unTexto.txt.

Marque la opción correcta:

a. Para abrir el archivo puede usar en el String la ruta “unText.txt”

b. Para abrir el archivo puede usar en el String la ruta “C:\\Users\\SuNombre\\Documentos\\unTexto.txt”

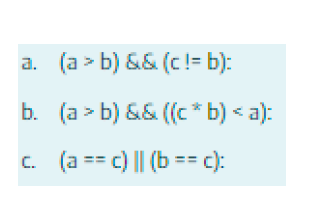
c. Para abrir el archivo puede usar en el String la ruta “Documentos\\unText.txt”

d . b y c son correctas

e. todas son incorrectas

**Java – Operadores lógicos y Bucles**

1. El dados el operador && (AND o “Y”) y el || (OR o “O”), y las variables a = 17, b = 4 y c = 4. Indique el resultado de las siguientes operaciones.



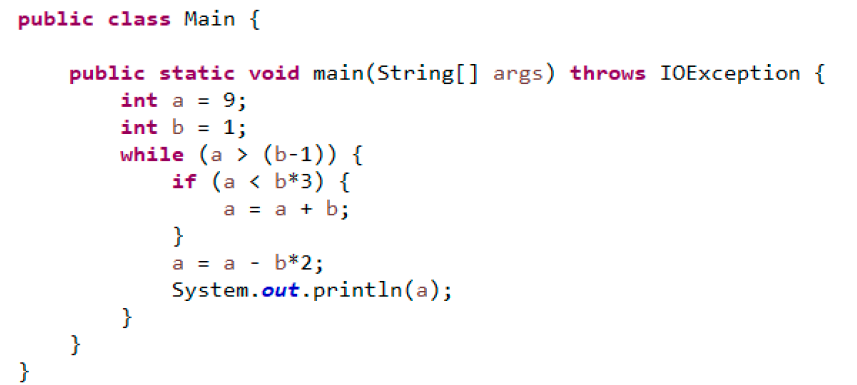
Respuestas:

a. falso

b. verdadero

c. verdadero

2. Dado el siguiente código que imprime por pantalla?



Respuesta:

Imprime los siguientes valores para a:

7,5,3,1,0

**Java – Arrays**

1. Si tenemos el siguiente array: int numeros[] = new int[]{1,37,16};

¿Qué valor es numeros [2]?

En la posicion [2] del array numeros, encontramos al valor “16”.

**Junit – Unit Test**

1. Liste los métodos básicos de un Test unitario (assertEqual, assertTrue, etc.), explique brevemente.

El método **assertEquals()** compara si dos objetos son iguales, utiliza el método equals().

Los métodos **assertTrue()** y **assertFalse()** simplemente validan un resultado si es verdadero o falso.

2. Liste los identificadores básicos de Test unitario (@Test, @Before, etc.), explique brevemente.

@before: para indicar que este método correrá antes de todos los @Test métodos.

@Test: Indicamos a Junit que se trata de un método de Test.

@BeforeEach la anotación se usa para indicar que el método anotado debe ejecutarse antes de cada invocación del método:

* @Test ,
* @RepeatedTest ,
* @ParameterizedTest
* @TestFactory en la clase actual.

@after: se ejecuta al finalizar cualquier test de la clase

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

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

b. El test que tiene la sentencia Assertions.assertEquals (false, hogar. tieneHambre) pasa satisfactoriamente

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

d. El test que tiene la sentencia Assertions.assertFalse (hogar. tieneHambre) no pasa satisfactoriamente.

Responda: Las respuestas correctas son las siguientes: a y d

**Java – POO**

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

No, ya que, los métodos estáticos no pueden hacer uso de la palabra reservada “this”.

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

No, en java no se puede hacer uso de la “herencia multiple”, por eso, para solucionar dicho problema se utilizan las interfaces

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

class CarritoCompra {

private List<ItemCarrito> items;

public CarritoCompra() {}

public void agregarItem(ItemCarrito ic) {

this.items.add(ic);

}

} ……..

CarritoCompra carrito1 = new CarritoCompra();

a. El carrito de compra no puede ser instanciado sin parámetros

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

c. carrito1.agregarItem (unItem) agregara un item a la colección items

d. carrito1.agregarItem(unItem) arrojará un error

Respuesta:

La opción correcta es la “d”, ya que, “items” no esta inicializado, solo se lo declaro.

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

Es un método estatico del tipo void, es el programa principal, que permite ejecutar nuestras aplicaciones.

5. Describa los modificadores de acceso (público, privado, protegido)

Con el modificador de acceso public: podemos acceder sin ninguna restricción, desde otras subclases, en el mismo paquete y hasta inclusive de otro paquete.

Con el modificador de acceso private: solo vamos a tener acceso dentro de la misma clase.

Con el modificador de acceso protected:dentro de la misma clase-subclase mismo paquete y no subclase en el mismo paquete,subclase en diferente paquete solo “static”.

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

En mi caso, uso el ide netbeans, lo hago de la siguiente forma: clic derecho dentro de la clase en cuestión, opción “insert code”-opcion getter o en su defecto – opción setter. Estos métodos nos van a permitir tener acceso a los atributos de cada objeto, ya que , si esta bien diseñada la clase, los atributos de la misma, se van a encontrar encapsulados, con estos métodos, voy a poder tener acceso a los mismos a partir de cualquier instancia de la clase en cuestion.

7. ¿Cómo instancio una Clase?

Carrito carrito= new Carrito();

8. ¿Cómo heredo de una Clase Abstracta?

Utilizando la palabra reservada “extends”

9. ¿Cómo heredo una interfaz?

Utilizando la palabra reservada “implements”.

10. ¿Cómo utilizo los métodos de una clase?

A partir de la instancia creada y utilizando luego el punto, ejemplo: la clase carrito, tiene el método realizar\_Compra(), utilizamos la instancia generada en el punto 7, y hacemos uso del método de la siguiente forma, **carrito.realizar\_Compra();**

11. ¿Cómo utilizo los métodos de una clase abstracta?

## Utilizando la palabra reservada @override, para sobrescribir el método y redefinir la funcionalidad/comportamiento del mismo.

12. ¿Cómo utilizo los métodos de una Interfaz?

Utilizando la palabra reservada @override, para sobrescribir el método y redefinir la funcionalidad/comportamiento del mismo.

**Eclipse**

1. Describa los pasos para crear un proyecto en Eclipse

En mi caso, que utilizo netbeans, lo hago de la siguiente forma:

File-new Project-opcion java with ant-java application-elijo el nombre y ubicacion para el proyecto,finish

2. Describa los pasos para crear una clase principal con el método Main(), dentro del proyecto.

Clic derecho sobre el proyecto-genero un nuevo paquete-dentro del paquete- new java main class,asigno el nombre,finish

3. Describa los pasos para crear paquetes

Primero creo un proyecto, luego clic derecho sobre el mismo- new package, asigno nombre, finish

4. Describa los pasos para crear una instancia de una clase C dentro de un paquete X, desde otra Clase A que está en otro paquete Z

Los pasos a seguir son los siguientes:

Primero debo importar el paquete y la clase en cuestión (en la clase donde quiero crear la instancia), dejo el ejemplo más abajo:

Import paquete X.clase C, una vez realizado el import correspondiente, podemos realizar la instancia de la clase C dentro de la clase A, lo hacemos de la siguiente forma, Clase C clase\_c= new Clase C ();