**Clase 9 – Actividad**

**Autor: Gil, Matias Ricardo**

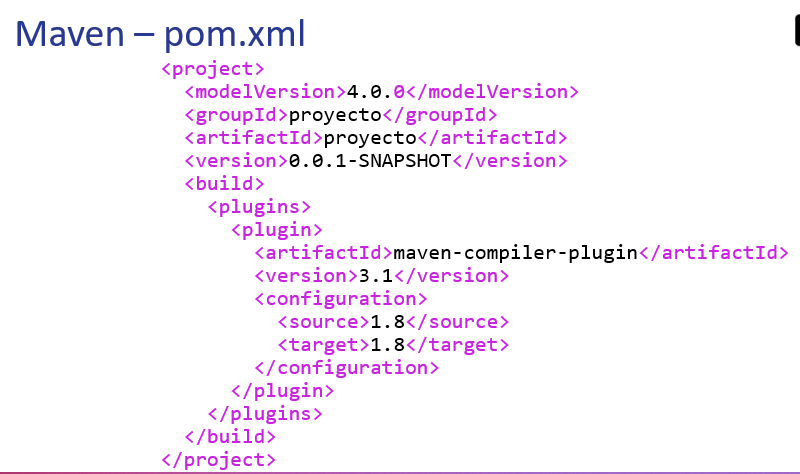
Maven

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

Ejecuta todos los pasos anteriores (clean, compile, test) y además junta el código compilado en un archivo .jar y deja la biblioteca disponible en el 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?

El archivo gestionado por Maven se llama pom.xml y tiene la siguiente estructura:



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

• mvn clean: Borra todos los .class y .jar generados.

• mvn compile: Compila el proyecto.

• mvn package: Genera el jar.

• mvn install: LLeva el jar a nuestro repositorio local de jars. Queda "visible" para otros proyectos maven en nuestro ordenador.

• mvn deploy: Lleva el jar a nuestro servidor de jars. Queda "visible" para otros proyectos maven en otros ordenadores. Este comando necesita que a maven se le haya indicado dónde está dicho servidor de jars.

• mvn javadoc:javadoc: Genera la documentación javadoc de nuestro proyecto.

• mvn site: Genera documentación html del proyecto. Por supuesto, necesitamos haber escrito en determinados ficheros el contenido de esa documentación.

• mvn eclipse:eclipse: que nos genera los ficheros típicos de un proyecto eclipse, de forma que podemos importar el proyecto directamente en nuestro entorno de desarrollo eclipse. (Ver Integración de Maven y Eclipse)

• mvn idea:idea: igual, pero para el entorno de desarrollo IntelliJ IDEA.

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 config --global user.name "Nombre Usuario"

• git config --global user.email "Email Usuario"

• git config --global credential.helper 'cache --timeout=3600': Recuerda usuario y contraseña temporalmente, en este caso una hora

• git config --global credential.helper cache: Recuerda usuario y contraseña durante un tiempo default de 15 min.

• git status : Hace un listado de los archivos nuevos o a insertar en un commit.

• git add <nombre de archivo> : Añade al stagin área (o comienza el seguimiento)

• git add . : Añade TODOS los archivos al stagin area (o comienza el seguimiento)

• git reset <archivo> : Saca de stagin area el archivo, poniendo -- . saca todos.

• git commit -m "Comentario de commit" : Realiza un commit. Guarda una copia del estado del archivo/s en ese momento y añade el código SHA a la cabecera del repositorio.

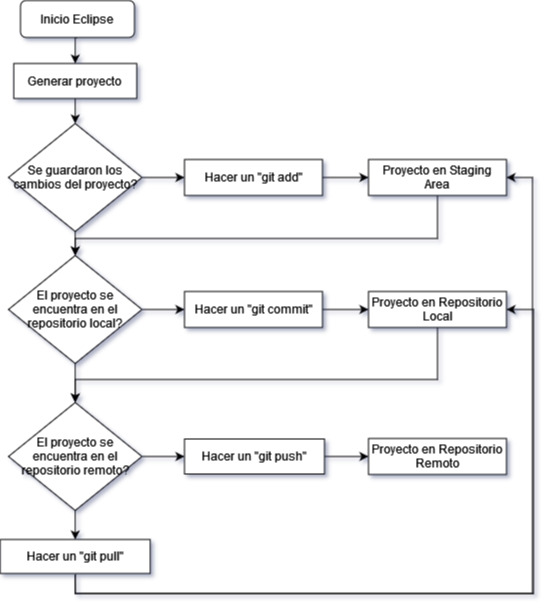
• git fetch : Descarga el historial del repo remoto.

• 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.

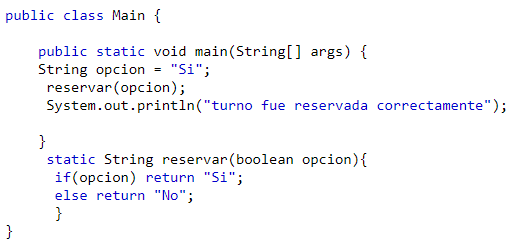
• git push <nombre\_de\_rama> : Sube los commits que tengamos en la rama local a la rama con ese nombre del repositorio remoto. Puede ser necesaria la palabra origin.

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



Excepciones:

Analice el código siguiente:



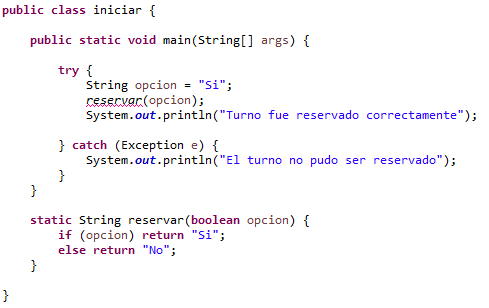
1. ¿Funciona o se produce una excepción?

SE PRODUCE UNA EXCEPCION.

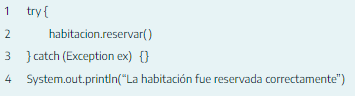
2. Explique narrativamente la mecánica en caso que funcione o Explique donde y porque se la excepción en caso que no funcione.

PORQUE RECIBE UN STRING EN VEZ DE UN BOOLEANO.

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



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



SE CORTA EL CODIGO EN LA 3° LINEA, PORQUE EN ESA LINEA SE PRODUCE UNA EXCEPCION.

Colecciones - clase HashMap

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

Map.put(): agregar un objeto HashMap.

Map.get(): leer un objeto HashMap.

Map.remove(): eliminar un objeto HashMap.

2. Responda:

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

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

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

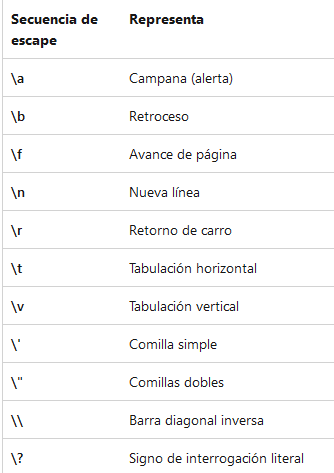
c. xx.get(“algo”) si la clave existe debe retornar un objeto tipo producto o null

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

Java – Literales, caracteres especiales, palabras reservadas

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

La barra invertida (\) es un carácter reservado. Las combinaciones de caracteres que constan de una barra diagonal inversa (\) seguidas de una letra o de una combinación de dígitos se denominan "secuencias de escape".



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

Se utiliza para representar un carácter de nueva línea, comillas simples u otros caracteres en una constante de caracteres, debe usar secuencias de escape.

Ejemplo:

System.out.println(“Seleccione una opción: \nOpcion1 \nOpcion2”);



Cuando se quiere representar una barra invertida (\) debe utilizarse doble barra invertida (\\). Por ejemplo:

String Path = “C:\ \Users\\Emi\\Carpeta Emi\\Documentos”;

Java – Métodos

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

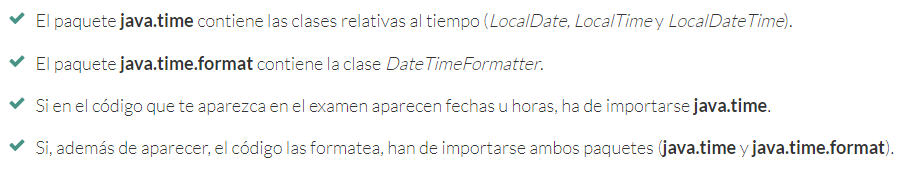
ES UN METODO STATIC SIN PARAMETROS.

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

Proviene de JAVA.TIME

Las clases relativas al tiempo en Java tienen su propio paquete

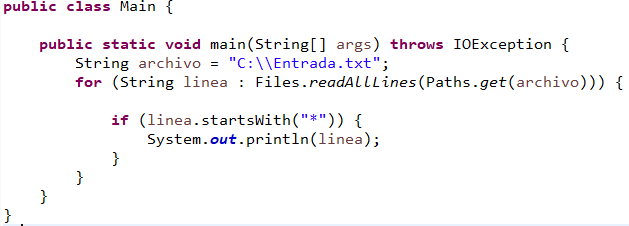
Esto es algo muy importante ya que, sin importar el paquete, el código no puede compilar.



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.

CREA UNA VARIABLE CON LA RUTA DEL ARCHIVO, RECORRE TODO LO QUE ESTA EN EL ARCHIVO, PREGUNTA SI LA LINEA COMIENZA CON ASTERISCO (\*), SI ES VERDADERO LO MUESTRA, SINO NO HACE NADA.

SE IMPRIMEN SOLAMENTE LA LINEAS QUE EMPIEZAN CON \*, O SEA, SOLO LA SEGUNDA LINEA.

Salida: 

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

ReadAllLines(): indica al sistema que debe leer todo lo escrito en el archivo txt.

Get(): toma como ruta el valor de la variable que se especifique.

StartsWith(): para indicar que solo tenga en cuenta las líneas que empiece con algún carácter en específico.

System.out.println(): sirve para imprimir en pantalla un mensaje.

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”

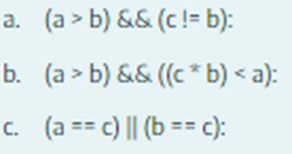
c. Para abrir el archivo puede usar en el String la ruta “Documentos\\unText.txt”

d . b y c son correctas

e. todas son incorrectas

Java – Operadores lógicos y Bucles

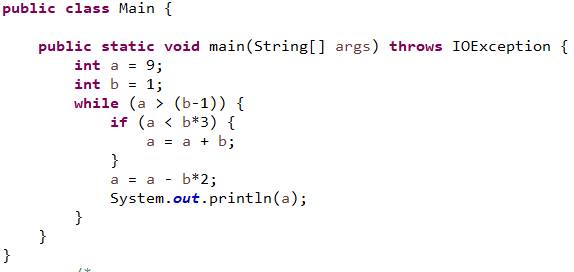
1. Dados el operador && (AND o “Y”) y el || (OR o “O”), y las variables a = 17, b = 4 y c = 4. Indique el resultado de las siguientes operaciones.

 FALSE

TRUE

TRUE

2. Dado el siguiente código que imprime por pantalla?



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]?

16

Junit – Unit Test

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

Las condiciones de aceptación del test se implementan con los asserts. Los más comunes son los siguientes:

• assertTrue/assertFalse (condición a testear): Comprueba que la condición es cierta o falsa.

• assertEquals/assertNotEquals (valor esperado, valor obtenido): Es importante el orden de los valores esperado y obtenido.

• assertNull/assertNotNull (object): Comprueba que el objeto obtenido es nulo o no.

• assertSame/assertNotSame(object1, object2): Comprueba si dos objetos son iguales o no.

• fail(): Fuerza que el test termine con fallo. Se puede indicar un mensaje.

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

• @Test: indica que el método que la contiene es un test: expected y timeout.

• @Before: ejecuta el método que la contiene justo antes de cada test. Por lo general, se usa para actualizar o restablecer el estado necesario para que los métodos de prueba se ejecuten correctamente.

• @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: ejecuta el método que la contiene justo después de cada test. Se puede utilizar para restablecer variables, eliminar archivos temporales, etc.

• @BeforeClass: ejecuta el método (estático) que la contiene justo antes del primer test.

• @AfterClass: ejecuta el método (estático) que la contiene justo después del último test.

• @Ignore: evita la ejecución del tests. No es muy recomendable su uso porque puede ocultar test fallidos. Si dudamos si el test debe estar o no, quizás borrarlo es la mejor de las decisiones.

3. Inicializar atributos en el método anotado con @BeforeEch de una clase de testeo causa que cada test unitario que utilice estos atributos sea atómico y aislado

VERDADERO

Java – POO

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

NO TIENE SENTIDO. UN METODO STATIC NO NECESITA SER INSTANCIADO.

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

UNA CLASE PUEDE HEREDAR SOLAMENTE DE UNA CLASE. HERENCIA SIMPLE.

UNA CLASE PUEDE IMPLEMENAR VARIAS INTERFASES (HERENCIA COMPUESTA)

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. FALSO

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

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

d. carrito1.agregarItem(unItem) arrojará un error. FALSO

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

El metodo Main() es Static void. Sirve para iniciar el programa.

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

• private: El campo o método sólo es visible dentro de la clase donde se define. DESDE EL PAQUETE USADO Y LA CLASE DONDE SE DEFINE.

• protected: El campo o método es visible en la clase en donde se define y en cualquiera de sus subclases. DESDE EL PAQUETE USADO

• public: El campo o método es visible en cualquier clase. DESDE CUALQUIER PAQUETE



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

En la barra de herramientas vamos a SOURCE/GENERATE GETTERS AND SETTERS.

Los Setters & Getters son métodos de acceso lo que indica que son siempre declarados públicos, y nos sirven para dos cosas:

• Setters: Del inglés Set, que significa establecer, pues nos sirve para asignar un valor inicial a un atributo, pero de forma explícita, además el Setter nunca retorna nada (Siempre es void), y solo nos permite dar acceso público a ciertos atributos que deseemos el usuario pueda modificar.

• Getters: Del inglés Get, que significa obtener, pues nos sirve para obtener (recuperar o acceder) el valor ya asignado a un atributo y utilizarlo para cierto método.

7. ¿Cómo instancio una Clase?

OBJETO nombre\_objeto = new CONSTRUCTOR();

8. ¿Cómo heredo de una Clase Abstracta?

Agregando en la clase que será hija la palabra EXTENDS seguida del nombre de la clase padre. Por ejemplo:

9. ¿Cómo heredo una interfaz?

Agregando en la clase que será hija la palabra IMPLEMENTS seguida del nombre de la interfaz.

10. ¿Cómo utilizo los métodos de una clase?

Nombre\_objeto.metodo\_clase(parametros);

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

NO SE PUEDE INSTANCIAR UNA CLASE ABSTRACTA POR LO TANTO LOS METODOS SE UTILIZAN DESDE LAS CLASES HIJAS.

12. ¿Cómo utilizo los métodos de una Interfaz?

Nombre\_objeto.metodo\_interfaz(parametros);

Eclipse

1. Describa los pasos para crear un proyecto en Eclipse

En la barra de herramientas vamos a ARCHIVO/NUEVO/PROYECTO JAVA

A continuación, detallamos el nombre del proyecto.

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

En la barra de herramientas vamos a ARCHIVO/NUEVO/CLASE

A continuación, detallamos el nombre de la clase y tildamos la opcion “Public static void main (String [] args)”.

3. Describa los pasos para crear paquetes

En la barra de herramientas vamos a ARCHIVO/NUEVO/PAQUETE

A continuación, detallamos el nombre del paquete.

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

