UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

Carrera:
INGENIERÍA ELECTRÓNICA Y AUTOMATIZACIÓN

Asignatura: PROGRAMACIÓN ORIENTADA OBJETOS

NRC
7492

Evidencias Primera Unidad

Apellidos y Nombres:
Manosalvas Clavijo Gabriel Alexander

ING. RUBEN ARROYO
Docente TC de DCCO

Sangolquí - Ecuador, febrero del 2022

Primera Unidad

TRABAJOS AUTÓNOMOS

UNIDAD 1

1. Laboratorio 1.- Excepciones.
2. TAA1.- Elaborar el resumen sobre Git y GitHub.
3. TAA2.- Transición de paradigmas.
4. TAA3.- Mapa conceptual de entornos de desarrollo de java.
5. TAA4.- Resumen de conceptos de POO, Objeto, atributos y métodos, clases.
6. TAA5.- Resumen de casos de uso y diagrama de clases.
7. TAA6.- Buenas prácticas de Programación.
8. TAG.- Trabajo de Autónomo Grupal 1 (Cuestionario)
9. Foro.- Encapsulamiento
10. Prueba Parcial
11. Examen Conjunto

Laboratorio 1

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO 1

OBJETIVO
Realizar un resumen de lo que es una Excepción en java

CARRERA: Ingeniería Electrónica y control	**GUÍA** No. 01	**TIEMPO ESTIMADO:** 1h y 30 min.
ASIGNATURA: Programación Orientada a Objetos POO **NRC:** 7492	**FECHA DE ELABORACION:** 9/11/2021 **SEMESTRE**: Noviembre – marzo 2022	
TÍTULO: Excepciones en Java	**DOCENTE:** Ing. Rubén Arroyo. Mgs	

INSTRUCCIONES

i) Realizar un resumen de Excepciones en java utilizando el formato entregado por el docente
ii) Estudiar y establecer lo que es una excepción en java

ACTIVIDADES

1. **Ubicación de recursos**
   i) Individual
   ii) Usar los recursos necesarios (libros, internet)

2. **Planteamiento del problema**
   a) Qué es una excepción
   b) Ejemplo
   c) Concepto
   d) conclusiones
   e) recomendaciones

3. **Entregable (s)**
   i) Archivo en formato especificado por el docente
   ii) También se deberá entregar un marco conceptual o antecedentes y las fuentes e información que contiene la información.

DOCENTE RESPONSABLE Ing. Rubén Arroyo. Mgs.	COORDINADOR DE ÁREA Ing. Silvia Arévalo Msc.

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO 1

UNIVERSIDAD DE LAS FUERZAS ARMADAS "ESPE"
CARRERA DE INGENIERÍA ELECTRONICA.

Nombre: Danilo Lucero Osorio
Gabriel Manosalvas Clavijo
Asignatura: Programación Orientada a Objetos
Fecha: 9/11/2021

No. Lista: 21
No. Lista: 22
Período: Pregrado
NRC: 7492

1. TEMA:

Excepciones en java.

1.1. SUBTEMAS:

- ¿Qué es una excepción?
- Ejemplo de excepción.
- Diagrama de las excepciones más comunes

2. OBJETIVOS:

2.1. OBJETIVO GENERAL:

Desarrollar el concepto de excepción en POO

2.2. OBJETIVOS ESPECÍFICOS:

- Establecer lo que es una excepción en Java.
- Incorporar un ejemplo de excepción.
- Entender mediante algunos tipos de excepciones, la forma de uso y forma de código en Java

3. DESARROLLO:

3.1 Definición de excepciones:

En Java los errores en tiempo de ejecución (cuando se está ejecutando el programa) se denominan excepciones, y esto ocurre cuando se produce un error en alguna de las instrucciones de nuestro programa, como por ejemplo cuando se hace una división entre cero, cuando un objeto es 'null' y no puede serlo, cuando no se abre correctamente un fichero, etc. Cuando se produce una excepción se muestra en la pantalla un mensaje de error y finaliza la ejecución del programa.

En Java (al igual que en otros lenguajes de programación), existen muchos tipos de excepciones y enumerar cada uno de ellos seria casi una labor infinita. En lo referente a las excepciones hay que decir que se aprenden a base experiencia, de encontrarte con ellas y de saber solucionarlas.

Cuando en Java se produce una excepción se crear un objeto de una determina clase (dependiendo del tipo de error que se haya producido), que mantendrá la información sobre el error producido y nos proporcionará los métodos necesarios para obtener dicha información. Estas clases tienen como clase padre la clase Throwable, por tanto, se mantiene una jerarquía en las excepciones.

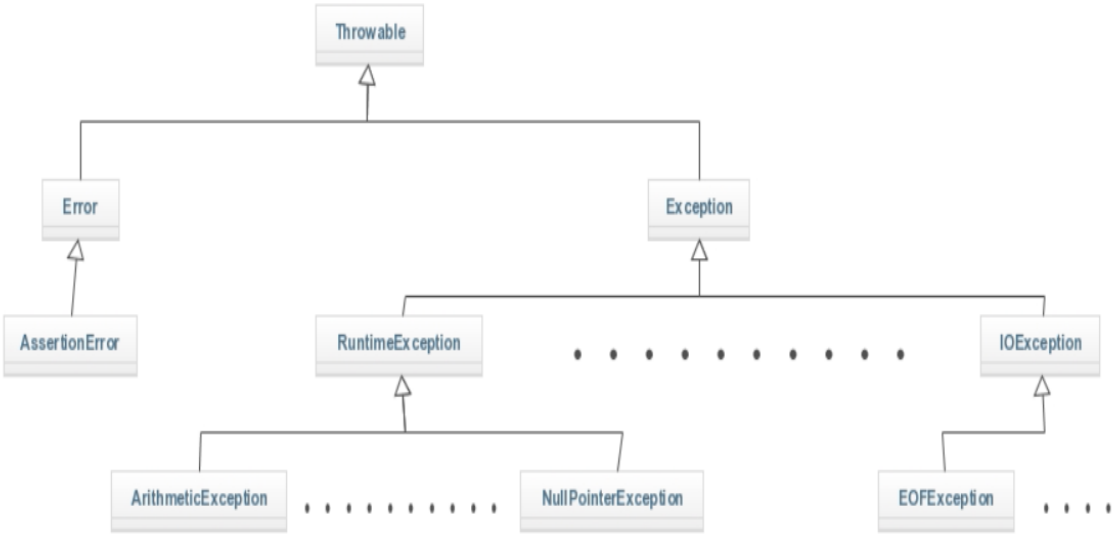
3.2 Tipos de Excepciones

Existen varios tipos fundamentales de excepciones:

- **Error:** Excepciones que indican problemas muy graves, que suelen ser no recuperables y no deben casi nunca ser capturadas.
- **Exception:** Excepciones no definitivas, pero que se detectan fuera del tiempo de ejecución.
- **RuntimeException:** Excepciones que se dan durante la ejecución del programa.

”**Melgar, V. (2016). Excepciones Java. Blog de IBM. Recuperado de https://www.ibm.com/docs/es/i/7.1?topic=driver-java-exceptions, el 5 de noviembre de 2018**”.

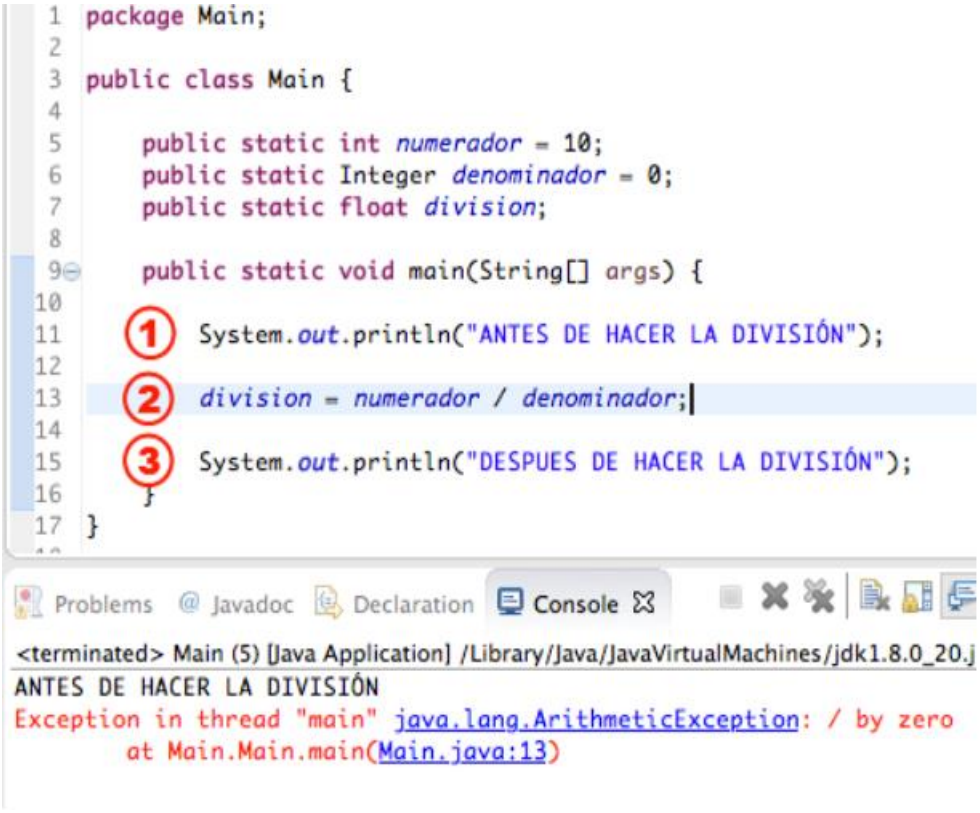
Algunos tipos mediante un gráfico:



”**Moya, R. (2019). Tipos de Excepciones Java. Lenguajes de programación. Recuperado de https: https://jarroba.com/excepciones-exception-en-java-con-ejemplos, el 5 de Enero de 2020**”.

Ejemplo:

A continuación vamos a mostrar un ejemplo de como al hacer una división entre cero, se produce una excepción. Veamos la siguiente imagen en el que podemos ver un fragmento de código y el resultado de la ejecución del código:



”**Moya, R. (2019). Tipos de Excepciones Java. Lenguajes de programación. Recuperado de https: https://jarroba.com/excepciones-exception-en-java-con-ejemplos, el 5 de Enero de 2020**”.

Como vemos en el programa de ejemplo tenemos 3 instrucciones. La primera debe de imprimir por pantalla el mensaje "ANTES DE HACER LA DIVISIÓN", la segunda debe de hacer la división y la última debe de imprimir por pantalla el mensaje "DESPUES DE HACER LA DIVISIÓN". La primera instrucción la ejecuta perfectamente, pero al llegar a la segunda se produce una "ArithmeticException" (excepción de la clase ArithmeticException) y se detiene la ejecución del programa ya que estamos dividiendo un número entre '0'.

Por suerte Java nos permite hacer un control de las excepciones para que programa no se pare inesperadamente y aunque se produzca una excepción, nuestro programa siga su ejecución. Para ello tenemos la estructura "try – catch – finally":

```
try {

        // Instrucciones cuando no hay una excepción

} catch (TypeException ex) {

        // Instrucciones cuando se produce una excepcion

} finally {

        // Instruciones que se ejecutan, tanto si hay como sino hay excepciones

}
```

Respecto a la estructura "try – catch – finally", se ha de decir que primero se ejecuta el bloque "try", si se produce una excepción se ejecuta el bloque "catch" y por último el bloque "finally". En esta estructura se puede omitir el bloque "catch" o el bloque "finally", pero no ambos.

Sabiendo esta estructura, podemos reescribir nuestro programa para que se ejecuten las tres instrucciones aunque se produzca una excepción. Previamente debemos de saber cual va a ser la clase de la excepción que puede aparecer que seria la "ArithmeticException" para definirla en la parte del "catch". Nuestro programa quedaría de la siguiente forma y se ejecutaría sin problema obteniendo también la información de la excepción:

```
1  package Main;
2
3  public class Main {
4
5      public static int numerador = 10;
6      public static Integer denominador = 0;
7      public static float division;
8
9      public static void main(String[] args) {
10         System.out.println("ANTES DE HACER LA DIVISIÓN");
11         try {
12             division = numerador / denominador;
13         } catch (ArithmeticException ex) {
14             division = 0; // Si hay una excepción doy valor '0' al atributo 'division'
15             System.out.println("Error: "+ex.getMessage());
16         } finally {
17             System.out.println("División: "+division);
18             System.out.println("DESPUES DE HACER LA DIVISIÓN");
19         }
20     }
21 }
```

```
Problems  Javadoc  Declaration  Console
<terminated> Main (5) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/Contents/Home/bin/java (1
ANTES DE HACER LA DIVISIÓN
Error: / by zero
División: 0.0
DESPUES DE HACER LA DIVISIÓN
```

”**Moya, R. (2019). Tipos de Excepciones Java. Lenguajes de programación. Recuperado de https: https://jarroba.com/excepciones-exception-en-java-con-ejemplos, el 5 de Enero de 2020**”.

Como vemos capturamos la excepción en un objeto "ex" de la clase "ArithmeticException" y podemos obtener el mensaje de error que nos da la excepción. Vemos también que el programa termina su ejecución aunque se haya producido una excepción.
Dentro de una misma estructura podemos definir todas las excepciones que queramos. En el caso anterior hemos definido solo la excepción "ArithmeticException"; pero por ejemplo, podemos definir también la excepción "NullPointerException", por si nos viene un valor a 'null' al hacer la división:

```java
package Main;

public class Main {

    public static int numerador = 10;
    public static Integer denominador = null;
    public static float division;

    public static void main(String[] args) {
        System.out.println("ANTES DE HACER LA DIVISIÓN");
        try {
            division = numerador / denominador;
        } catch (ArithmeticException ex) {
            division = 0; // Si hay una excepción doy valor '0' al atributo 'division'
            System.out.println("Error: "+ex.getMessage());
        } catch (NullPointerException ex) {
            division = 1; // Si si la excepción es de un null doy valor '1' al atributo 'division'
            System.out.println("Error: "+ex.getMessage());
        } finally {
            System.out.println("División: "+division);
            System.out.println("DESPUES DE HACER LA DIVISIÓN");
        }
    }
}
```

Problems @ Javadoc Declaration Console

```
<terminated> Main (5) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_20.jdk/Contents/Home/bin/java (12/10/2014 16:
ANTES DE HACER LA DIVISIÓN
Error: null
División: 1.0
DESPUES DE HACER LA DIVISIÓN
```

4. CONCLUSIONES:

- Las excepciones son un mecanismo para capturar y generar errores en un programa en tiempo de ejecución
- Para lanzar manualmente una excepción en java se usa la palabra clave throw.
- Cuando se está programando se pueden presentar diferentes tipos de excepciones y uno de ellos como ejemplo puede ser cuando se escribe de forma incorrecta algún código esté a su vez genera un tipo de excepción.

5. RECOMENDACIONES:

- En la colección de clases de la API de Java se incluyen muchas excepciones
- En la documentación Javadoc como en el resto de clases incluyen una descripción con la condición de error que indican
- Investigar más acerca de este tema en internet, puesto que podemos encontrar información muy valiosa que nos puede ayudar en este gran mundo de la programación.

6. BIBLIOGRAFIA

Blanco, Y. (2021). *Introduccion A La Programacion Orientada A Objetos*. Andavira.

Jackson, T. (2021). *Introducción del lenguaje Java*.

Budd, Timothy. (1994), Introducción *a la programación orientada a objetos, (1994), Addison-Wesley.*

OBSERVACIONES

1. PONER SUBTEMAS EN EL DESARROLLO QUE SE MUESTREN EN EL PANEL DE NAVEGACIÓN
2. REVISAR COMO PONER LOS OBJETIVOS
3. PONER CITAS,
4. NO ENCUENTRO EL RESUMEN
5. USAR LAS CITAS PUESTAS EN LAS REFERENCIAS
6. USAR NORMAS APA EN LAS FIGURAS

NOTA: 10/20

Trabajos Autónomos Individuales

UNIVERSIDAD DE LAS FUERZAS ARMADAS
ESPE

GUÍA DEL TRABAJO AUTÓNOMO DOCENTE

ASIGNATURA:

PROGRAMACIÓN ORIENTADA A OBJETOS (POO)

ING. RUBEN ARROYO

DOCENTE TC DE DCCO

Sangolquí- Ecuador, OCTUBRE de 2021

ÍNDICE

Contenido

UNIVERSIDAD DELAS FUERZAS ARMADAS - ESPE
GUÍA DE APRENDIZAJE AUTÓNOMO DE LA ASIGNATURA DE: INTRODUCCIÓN A LA PROGRAMACIÓN

A. INFORMACIÓN GENERAL:

- DEPARTAMENTO: Ciencias de la Computación
- ÁREA DE CONOCIMIENTO: Programación
- CARRERA: Electrónica y Automatización
- NIVEL: Segundo
- PERIODO ACADÉMICO: SII 202151
- CARGA HORARIA POR COMPONENTES DE APRENDIZAJE

CD	CP/E	CA	HS	HPAO
3	3	3	9	144
48	48	48	6	3

- ESTUDIANTE: Gabriel Alexander Manosalvas Clavijo
- DOCENTE: RUBÉN DARÍO ARROYO CHANGO
- CORREO ELECTRÓNICO INSTITUCIONAL DEL DOCENTE: rdarroyo@espe.edu.ec

“PRIMERA UNIDAD”

TRABAJOS AUTÓNOMOS

UNIDAD UNO

1. **TAA1.- Elaborar el resumen sobre Git y GitHub.**
2. **TAA2.- Transición de paradigma**
3. **TAA3.- Entorno de Desarrollo**
4. **TAA4.- Revisión de conceptos generales de la POO**
5. **TAA5.- UML: Diagramas de Clase**
6. **TAA6.- Buenas prácticas de programación**

B. EVIDENCIAS DEL TRABAJO AUTÓNOMO 1:

1. Tema: Software VCS: Git, GitHub

2. Objetivos:

1. Objetivo General:

Conocer lo que es un software VCS, Git, GitHub mediante su uso y sus aplicaciones.

2. Objetivos Específicos:

- Conocer el concepto de VCS, para utilizarlo en un ejemplo.
- Mostrar un ejemplo que se aplique al Git
- Mostrar un ejemplo que se aplique al GitHub

3. Desarrollo:

1. Definición de Software VCS: Git, GitHub

a) Software VCS:

Conocido como sistema de control de revisiones o de fuentes, es una herramienta de software que monitoriza y gestiona cambios en un sistema de archivos. Este ofrece herramientas de colaboración para compartir e integrar dichos de cambios en otros usurarios del VCS. Aplica como un repositorio que monitorea el sistema e archivos al alcance de archivos individuales de códigos fuentes, el cual se puede monitorear, eliminar y modificar las líneas de texto que contiene dicho archivo.

b) Git:

Git es una herramienta que realiza una función del control de versiones de código de forma distribuida de la que destacamos varias características:

- Es muy potente
- Fue diseñada por Linus Torvalds
- No depende de un repositorio digital
- Es software libre
- Tiene un sistema de trabajo con ramas que lo hace potente
- Está destinada a provocar proyectos divergentes

c) GitHub:

Es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador, y que fue comprada por Microsoft. Permite que los desarrolladores suban el código de sus aplicaciones y herramienta, y que como usuario se pueda colaborar en su desarrollo.

4. Conclusiones:

- Podemos concluir que el VCS nos ayuda a controlar las revisiones fuente de una herramienta de software.
- Tras el análisis, podemos deducir que el sistema de control Git es muy útil porque permite a los desarrolladores descargar software y subir la versión que han modificado.
- Tal y como comprobamos, la herramienta GitHub es una forma de colaborar con otros desarrolladores al momento de almacenar códigos y que todos los colaboradores puedan trabajar en dicho código'.

5. Recomendaciones:

- Extender los estudios y utilizar las plataformas VCS, Git y GitHub para un mejor entendimiento.
- Analizar con mayor detenimiento las plataformas para su correcta utilización.

6. Bibliografía:

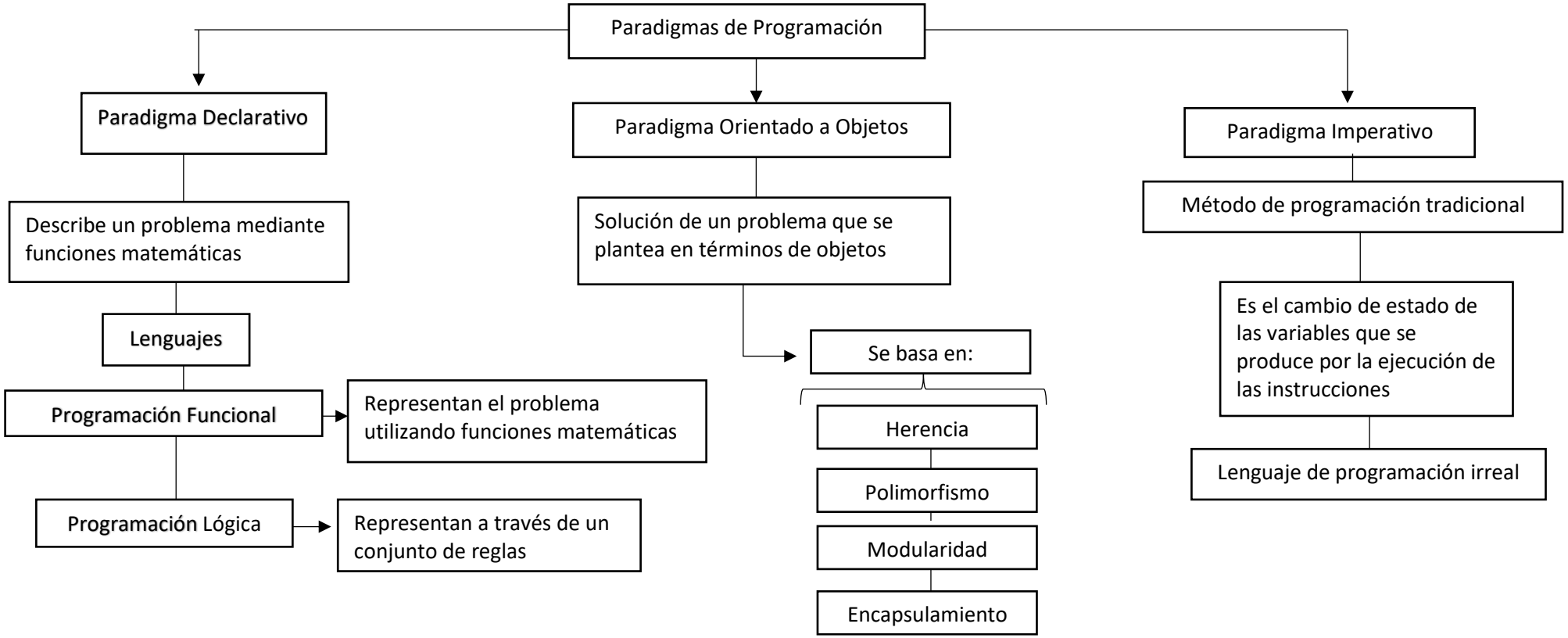
- Rubio, J. C. (2020, 30 julio). *Qué es GIT y para qué sirve*. OpenWebinars.net. https://openwebinars.net/blog/que-es-git-y-para-que-sirve/

- Carmenate, J. G. (2020, 15 diciembre). *Introducción a Git y GitHub - Crea 🐱 Organiza y Colabora 🚀*. Programar fácil con Arduino. https://programarfacil.com/blog/arduino-blog/git-y-github/

- Fernández, Y. (2019, 30 octubre). *Qué es Github y qué es lo que le ofrece a los desarrolladores*. Xataka. https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores

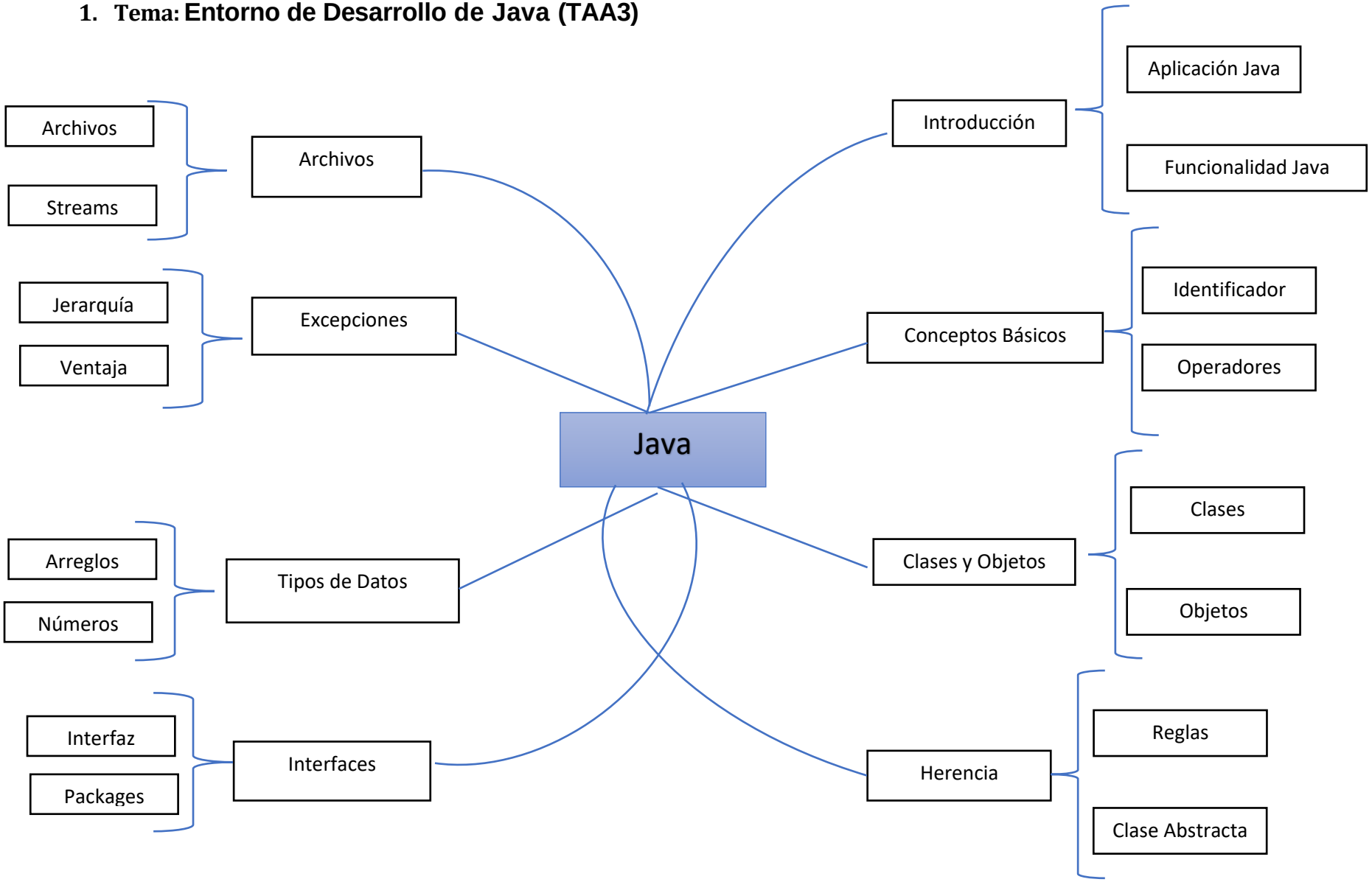
C. EVIDENCIAS DEL TRABAJO AUTÓNOMO 2:

1. Tema: Transición de paradigma (TAA2)



D. EVIDENCIAS DEL TRABAJO AUTÓNOMO 3:

1. Tema: Entorno de Desarrollo de Java (TAA3)



E. EVIDENCIAS DEL TRABAJO AUTÓNOMO 4:

1. Tema: Revisión de conceptos generales de la POO (TAA4)

2. Objetivos:

2.1 Objetivo General:

Entender los conceptos generales de la Programación Orientada a Objetos

2.2 Objetivos Específicos:

- Mostrar los conceptos generales de POO
- Reconocer los conceptos generales de POO

3. Desarrollo:

3.1 Definición de conceptos generales de POO

a) Conceptos generales

Es un paradigma de programación que usa objetos y sus interacciones, para desear aplicaciones y programas informáticos. Está basado en varias técnicas, incluyendo herencia, atracción, polimorfismo y encapsulamiento. Su uso se popularizo a principios de la década de 1990.

Entendamos que los conceptos generales son:

Clase: definiciones de las propiedades y comportamiento de un tipo de objeto en concreto.

Herencia: Es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C.

Objeto: Entidad provista de un conjunto de propiedades o atributos de comportamiento o funcionalidad.

Método: Algoritmo asociado a un objeto.

Evento: Es un suceso en el sistema.

Propiedades: Contenedor de un tipo de datos asociados a un objeto.

Estado interno: Es un variable que se declara privada.

4. Conclusiones:

- Comprobamos que los conceptos básicos son necesarios para el entendimiento de la programación.
- Identificamos los componentes de los objetos como lo son los atributos, identidades, relaciones y métodos.

5. Recomendaciones:

- Profundizar y repasar los conceptos básicos de la POO.

6. Bibliografía:

- EcuRed. (s. f.). *Programación Orientada a Objetos - EcuRed*. https://www.ecured.cu/Programaci%C3%B3n_Orientada_a_Objetos#Conceptos_fundamentales

F. EVIDENCIAS DEL TRABAJO AUTÓNOMO 5:

1. Tema: UML Diagramas de Clases (TAA5)

2. Objetivos:

2.1 Objetivo General:

Entender el uso de los Diagramas de Clases, utilizando diversos mecanismos de abstracción

2.2 Objetivos Específicos:

- Entender la representación de los aspectos de los diagramas de clases
- Reconocer los mecanismos de diagramas de clases

3. Desarrollo:

3.1Definición de Diagramas de Clases

a) Conceptos generales

El diagrama de clase de objetos y sus asociaciones. En este diagrama se representa la estructura y el comportamiento de cada uno de los objetos del sistema y sus relaciones con los demás objetos, pero no muestra información temporal.

Las clases que describen un conjunto de objetos con propiedades y comportamiento común, que puede ser:

- Los atributos de una clase representan los datos asociados a los objetos instanciados por esa clase.
- Las operaciones o métodos representan las funciones o procesos de los objetos de una clase, caracterizando a dichos objetos.

4. Conclusiones:

- Determinamos que los atributos son una clase de datos asociados.
- Identificamos las operaciones que representan las funciones de los objetos.

5. Recomendaciones:

- Entender todo el conjunto de relaciones y clases para entender bien los diagramas de clases.

6. Bibliografía:

- *Diagrama de Clases*. (2016, 4 diciembre). manuel.cillero.es. https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/diagrama-de-clases/

G. EVIDENCIAS DEL TRABAJO AUTÓNOMO 6:

1. **Tema: Buenas prácticas de programación (TAA6)**
2. **Objetivos:**

2.1 Objetivo General:

Determinar el uso correcto al momento de programar mediante las buenas prácticas de programación

2.2 Objetivos Específicos:

- Entender las buenas prácticas de programación
- Reconocer los mecanismos para realizar buenas prácticas de programación

3. **Desarrollo:**

3.1 Definición de buenas prácticas de programación

a) Conceptos generales

Debemos tomar en cuenta varios aspectos como, por ejemplo, limitar siempre el alcance de las variables locales, pero algunas veces pueden insertar durante el código. Minimizar el código sea más legible para que tengan menos errores y sea más mantenible.

Debemos procurar inicializar una variable desde su declaración, si eso no es posible asignar un valor nulo y se inicializa con los objetos vacíos cuando haga falta, en java es una de las operaciones más costosas en términos de uso de memoria e impacto en la creación de objetos, esto es evitable inicializando los objetos solo cuando sean requeridos los códigos.

4. **Conclusiones:**

- Identificamos las formas más efectivas para realizar el código.
- Determinamos como inicializar de una manera más fácil las variables para una buena práctica de programación.

5. **Recomendaciones:**

- Comprender de mejor manera el uso correcto para realizar buenas practicas de programación.

6. **Bibliografía:**

- (s. f.-a). *Buenas Prácticas en la programación orientada a objetos - #1*. Steemit. https://steemit.com/java/@dfuenmayor/buenas-practicas-en-la-programacion-

T.A.G 1

CUESTIONARIO

1.1. UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

GUÍA DEL TRABAJO AUTÓNOMO DOCENTE

ASIGNATURA:

PROGRAMACIÓN ORIENTADA A OBJETOS (POO)

ING. RUBEN ARROYO

DOCENTE TC DE DCCO

Sangolquí- Ecuador, noviembre de 2021

Preguntas

¿Qué significa las siglas VCS?

A. Versión de control del Sistema
B. Sistema de control de versiones
C. Software de control de versiones
D. Ver controles del Sistema

Respuesta:B

¿Qué es un paradigma de programación

Es un tipo de programación de software

Es un código de programación de software

Es un lenguaje de programación software

Es una regla de programación de software

Respuesta:C

¿Para qué sirve el VCS?

Para eliminar errores en el código del software

Para corregir errores en el código del software

Para crear códigos en el software

Para gestionar los cambios en el código de software

Respuesta:D

¿Cuál de los siguientes enunciado no pertenece a los tipos de paradigmas?

Imperativo

Funcional

Lógico

Exacto

Respuesta:D

¿Qué es un entorno de desarrollo o IDE?

Es un conjunto de procedimientos y herramientas que se utilizan para desarrollar un programa.

Es un conjunto de paradigmas y herramientas que se utilizan para desarrollar un programa

Es un conjunto de códigos y herramientas que se utilizan para desarrollar una clase

Es un conjunto de programas que se utilizan para desollar una clase.

Respuesta:A

¿Qué es la programación orientada a objetos?

Es un módulo de programación

Es un código de programación

Es un paradigma de programación

Es una clase de programación

Respuesta:C

¿Que es un diagrama de clase?

Conjunto de métodos y atributos

Conjunto de paradigmas y códigos

Conjunto de códigos y atributos

Conjunto de métodos y códigos

Respuesta:A

¿Qué es un objeto?

Es una unidad dentro de un programa que tiene un estado, y un comportamiento.

Es un conjunto de unidades dentro de un programa que tiene un estado y un comportamiento

Es un código que tiene un programa q tiene estado y comportamiento

Es un conjunto de unidad fuera de un programa que tiene un estado y un comportamiento

Respuesta: A

9. ¿Qué es un atributo?

Semejanzas de uno o varios objetos que compartes distintos

Características grupales de uno varios conjuntos de objetos que determinan su apariencia

Características individuales que diferencian un objeto de otro y determinan su apariencia

Diferencias de uno o varios objetos que comparten los mismos rasgos.

Respuesta: C

10. ¿Qué es un método?

Un bloque de código que tiene definido en su interior un conjunto de instrucciones

Un conjunto de códigos que tiene en su exterior un conjunto de instrucciones

Un bloque de atributos definidos en el interior de un conjunto de instrucciones

Un conjunto de clases que tiene definido en su interior un conjunto de instrucciones

Respuesta: A

¿Qué es el control de versonamiento?

Formas de clasificar lenguajes de programación.

Aplicaciones que permiten al programador implementar abstracciones del mundo real.

Un sistema de control de versiones es una herramienta capaz de registrar todos los cambios que se realizan en uno o más proyectos.

Posibilidades de manejo de errores para sus programas.

Respuesta: C

¿Qué es una excepción?

Copiar un directorio de su equipo local el archivo que iba a ser modificado indicando la fecha de modificación.

Un evento que se produce cuando se ejecuta el programa de forma que interrumpe el flujo normal de instrucciones.

Estilo de programación en un software.

Desarrollo de software por medio de técnicas simples que facilitan la escritura.

Respuesta: B

¿Qué es un encapsulamiento'

El ocultar el estado de un objeto de manera que sólo se puede cambiar mediante las operaciones definidas para ese objeto.

Una secuencia de acciones, alternativas o repetitivas.

Una agrupación de datos y de funciones que operan sobre esos datos.

Una unidad dentro de un programa informático que tiene un estado y comportamiento

Respuesta: A

¿Qué es una clase?

Agrupación de datos y de funciones que operan sobre esos datos.

Una herramienta capaz de registrar los cambios que se realizan en uno o más proyectos.

Un evento que se produce cuando se ejecuta el programa de forma que interrumpe el flujo normal de instrucciones.

Aplicaciones que permiten al programador implementar las abstracciones del mundo real.

Respuesta: A

¿Qué es un código limpio?

Un programa informático que tiene un estado y un comportamiento.

Estilo de programación en un software.

Formas de clasificar lenguajes de programación.

Técnicas simples que facilitan la escritura y lectura de un código volviéndolo más fácil de entender.

Respuesta: D

¿Qué es una lectura de datos por consola?

Copiar un directorio de su equipo local el archivo que iba a ser modificado indicando la fecha de modificación.

Desarrollo de software por medio de técnicas simples que facilitan la escritura.

Se puede usar para leer una entrada similar a la contraseña sin hacer eco de los caracteres ingresados por el usuario.

El ocultar el estado de un objeto de manera que sólo se puede cambiar mediante las operaciones definidas para ese objeto.

Respuesta: C

¿Qué es un paradigma de programación?

Forma de clasificar los lenguajes de programación en función de sus características.

Una secuencia de acciones, alternativas o repetitivas.

Un evento que se produce cuando se ejecuta el programa de forma que interrumpe el flujo normal de instrucciones.

Posibilidades de manejo de errores para sus programas.

Respuesta: A

¿Cuál es la estructura de un programa?

Copiar un directorio de su equipo local el archivo que iba a ser modificado indicando la fecha de modificación

Declaración de tipos, variables, subprogramas, cuerpo del programa.

Estilo de programación en un software.

Formas de clasificar lenguajes de programación.

Respuesta: B

¿Qué es persistencia de datos?

El ocultar el estado de un objeto de manera que sólo se puede cambiar mediante las operaciones definidas para ese objeto.

Propiedad de los datos para que estos sobrevivan de alguna manera.

Estilo de programación en un software.

Formas de clasificar lenguajes de programación.

Respuesta: B

¿Qué es un constructor?

Es una subrutina cuya misión es inicializar un objeto de una clase.

El ocultar el estado de un objeto de manera que sólo se puede cambiar mediante las operaciones definidas para ese objeto.

Desarrollo de software por medio de técnicas simples que facilitan la escritura.

Un evento que se produce cuando se ejecuta el programa de forma que interrumpe el flujo normal de instrucciones.

Respuesta: A

21. ¿Como se denomina al ocultamiento del estado, en Poo?

Encapsulamiento

Protegido

Cerrado

Respuesta: A

22. ¿Cómo este asilado cada objeto?

Interior

Exterior

Ninguna

Respuesta: B

23. ¿La clase tiene que tener obligatoriamente el mismo?

Nombre del constructor

Argumento

Ninguna

Respuesta: A

24. ¿Un constructor está diseñado específicamente para?

Crear

Cerrar

Inicializa

Respuesta: C

25. ¿Qué son los getter y setter en programación?

Métodos

Constructores habituales

Código

Respuesta: B

26. ¿Qué permite realizar el getter y setter?

Encapsular

Declarar

Inicializar

Respuesta: A

27. ¿Qué almacena un array en Poo?

Datos

Referencias del objeto

Ninguna

Respuesta: B

28. ¿Para qué usar arreglo?

Crear datos

Inicializar datos

Lectura de datos

Respuesta: C

29. ¿Qué representa una clase?

Abstracción

Constructor

Selector

Respuesta: B

30. ¿Qué representa el objeto de una clase?

La relación entre clase y objetos

Acoplamiento

Uso

Respuesta: A

Foro

Gabriel Alexander Manosalvas Clavijo - 18/11/2021 08:38

Encapsulamiento

Luego de que se crea una clase es posible que se utilice muchas veces, pero en determinado punto la clase puede comenzar a comportarse de una forma no deseada debido a que los valores de las variables.

¿Qué es Encapsulamiento de datos?

Es el proceso que consiste en organizar los datos y operaciones (métodos) de una clase que constituyen su estructura y su comportamiento con el fin de evitar el acceso a datos por cualquier otro medio que no sea el especificado y por esto el encapsulamiento de datos asegura la integridad de los datos que contiene el objeto.

¿Cómo se Encapsulan los datos?

Público (Public): Todos pueden acceder a los datos o métodos de una clase que se definen con este nivel, este es el nivel más bajo, esto es lo que tu quieres que la parte externa vea.

Protegido (Protected): Podemos decir que estás no son de acceso público, solamente son accesibles dentro de su clase y por subclases.

Privado (Private): En este nivel se puede declarar miembros accesibles sólo para la propia clase.

Bibliografía:

https://styde.net/. (s. f.). *Encapsulamiento en la programación orientada a objetos*. Styde.net. https://styde.net/encapsulamiento-en-la-programacion-orientada-a-objetos/

Prueba Parcial

```
/*              Universidad de las Fuerzas Armadas ESPE
          Carrera de Ingeniería en Electrónica y Automatización
Nombre: Gabriel Alexander Manosalvas Clavijo       No. Lista: 22
Asignatura:  Programación Orientada a Objetos      Período: PREGRADO S-II OCT21-MAR22
Fecha: 07/02/2022                                                    NRC: 7492

Tema: Prueba Parcial 1
package Prueba1;

import java.io.IOException;
import java.util.InputMismatchException;
import java.util.Scanner;

/**
 *
 * @author Gabriel
 */
public class Prueba1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IOException {
        //Menu
        Scanner sn = new Scanner(System.in);
        //Variables
        boolean salir = false;
        int opcion;


        while(!salir){
        //Menu
            System.out.println("C.- Crear un objeto");
            System.out.println("M.- Mostrar el Objeto");
            System.out.println("L.- Limpiar o vacirar el Objeto");
            System.out.println("S.- Salir");
            try{
            System.out.println("Ingrese una opcion: ");
            opcion = sn.nextInt();

            switch(opcion){
                case 1:
                    System.out.println("Ingrese un objeto:");
                    break;
                case 2:
                    System.out.println("Mostrar el objeto:");
                    break;
                case 3:
                    System.out.println("Inicializar los valores");
                    break;
                case 4:
```

```
                salir= true;
                break;

            }
        }
        catch(InputMismatchException e){
                System.out.println("Debes ingresare una letra");
                sn.next();

        }
      }

   }
}
package Prueba1;

import java.io.IOException;
import java.util.InputMismatchException;
import java.util.Scanner;

/**
 *
 * @author Gabriel
 */
public class Prueba1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IOException {
        //Menu
        Scanner sn = new Scanner(System.in);
        //Variables
        boolean salir = false;
        int opcion;


        while(!salir){
        //Menu
            System.out.println("C.- Crear un objeto");
            System.out.println("M.- Mostrar el Objeto");
            System.out.println("L.- Limpiar o vacirar el Objeto");
            System.out.println("S.- Salir");
            try{
            System.out.println("Ingrese una opcion: ");
            opcion = sn.nextInt();

            switch(opcion){
                case 1:
                    System.out.println("Ingrese un objeto:");
                    break;
                case 2:
                    System.out.println("Mostrar el objeto:");
                    break;
                case 3:
```

```
            System.out.println("Inicializar los valores");
            break;
        case 4:
            salir= true;
            break;

        }
    }
    catch(InputMismatchException e){
            System.out.println("Debes ingresare una letra");
            sn.next();

    }
}
```

Prueba Conjunta

Contenido > Unidad > Evaluación

Primer parcial

Evaluación 1ra Unidad

Evaluación 1ra Unidad
Puntaje total: 420.00
Puntaje de aprobación: 294.00
Incorrectas restan: No
Abierta: desde 30/11/2021 07:45 hasta 30/11/2021 08:30

Realización

Fecha: 30-nov-2021 07:45:10
Tiempo realización: 00:44:33
Cantidad de veces realizada: 1
Cantidad de respuestas correctas: 14 / 21

✖ No aprobada - 280.00

¿Juanito desea viajar de Miami a Guayaquil, para lo cual toma el vuelo de las 7h00, el cual lo realiza en un avión Aircraft Avianca Boeing 727-21, cuyo Registro es: HK-1803, indicar, que sería Guayaquil?

- (●) Objeto
- ( ) Atributo
- ( ) Método
- ( ) Clase

¿Juanito desea viajar de Miami a Guayaquil, para lo cual toma el vuelo de las 7h00, el cual lo realiza en un avión Aircraft Avianca Boeing 727-21, cuyo Registro es: HK-1803, indicar, que sería Juanito?

- ( ) Objeto
- ( ) Atributo
- ( ) Método
- (●) Clase

¿En un diagrama de casos de usos cual es el elemento que representa tanto a una persona como un sistema?

- (●) Actor
- ( ) Sistema
- ( ) Objeto
- ( ) Clase

¿En qué consiste el Código Limpio?

- ( ) Abrir una ventana en blanco en netbeans para iniciar un programa.
- ( ) Realizar la tabulación de atributos y métodos
- (●) Aplicar técnicas simples que facilitan la escritura y lectura de un código
- ( ) Digitar comentarios sobre los códigos escritos.

¿Cuáles son las partes o bloques que componen un programa?

- (●) Entrada y Salida
- ( ) Instrucciones y Declaraciones
- ( ) Principal y Secundario
- ( ) Atributos

¿Cuántos tipos de datos primitivos existen en Java?

- ( ) 5
- (●) 8
- ( ) 10
- ( ) 9

¿Cuáles son las maneras de lectura de datos en Java?

- (●) Buffered Reader, Scanner, Console
- ( ) Buffered Writer, Scanner, ConsoleReader, Printline, NextLine
- ( ) Scanner, System.in, Buffer Reader
- ( ) Buffered Reader, Input Stream Reader, printf

¿Que indica una excepción en un programa?

- ( ) El estado del programa
- (●) Un error en el programa.
- ( ) Los datos del programa
- ( ) La ejecución del programa.

¿Cuál es el beneficio principal de un encapsulamiento?

- ( ) Modificar los datos de una clase de manera más rápida
- ( ) Permite resolver los errores que se producen en el desarrollo de un programa
- (●) Los datos de una clase no pueden ser modificados por alguien externo
- ( ) Mantiene datos variados que son usados en el desarrollo de todo un programa

¿Qué tipo de almacenamiento permiten los arreglos?

- ( ) Dinámico
- ( ) Estático
- (●) Variable
- ( ) Modificable

¿Qué tipo de almacenamiento permiten las colecciones?

- (●) Dinámico
- ( ) Estático
- ( ) Variable

- Modificable

¿Cuáles son los tipos de colecciones?

- Public y Private
- Heredado y Protegido
- Desordenado y Variable
- List y Set

¿Cuántos elementos tengo en la siguiente declaración de un vector: float vector[]= {(float)(3.5),4,9,1,444,20,25};?

- 6
- 7
- 8
- 9

¿Qué instrucción o sentencia es la adecuada para ordenar ascendentemente un vector?

- Arrays.sort(vector);
- Array.sort(vector);
- vector.sort();
- sort.arrays (vector);

¿Qué se debería cambiar en el Método de Ordenación de la Burbuja, para hacerlo más eficiente?

- Contador i
- Contador j
- El contador i y j
- La variable temporal

¿Cómo se debería declarar un vector para registrar las notas de un estudiante?

- ArrayList<Float> notas= new Notas<Float>();
- ArrayList<Float> notas= new ArrayList< >();
- ArrayList<int> notas= new ArrayList<int>();
- ArrayList<float> notas= new ArrayList<float>();

¿Qué método se debería utilizar para ver el tamaño de un vector dinámico tipo ArrayList?

- size()
- length()
- toLowerCase()
- toUpperCase()

¿Qué se debería utilizar para obtener el valor de un número real almacenado en un vector tipo ArrayList?

- vector.get(i);
- vector.getNumero(i);
- vector.getNumero();
- vector.getNumero[i];

¿Qué método se debería utilizar para eliminar un nodo del vector tipo ArrayList?

- delete;
- erase;
- cls;
- remove;

¿Qué está mal en la siguiente clase:
public class e01Estructura {
 int numero=5;

 public static void main(String[] args) {
 System.out.println("El numero es:"+numero);
 }
}?

- Atributo: int numero=5;
- El mensaje: "El numero es:"
- La línea: System.out.println("El numero es:"+numero);
- Ninguno

¿Cuál es el valor que se muestra por pantalla
int x=10;
int y=0;
while (y<x) {
 y += x;
}
System.out.println(y);
?

- 8
- 9
- 10
- 11

Anterior Siguiente