INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

UNIDAD DE APRENDIZAJE: SISTEMAS DISTRIBUIDOS

TAREA 1: FUNDAMENTOS DE JAVA

NOMBRE: FARRERA MENDEZ EMMANUEL SINAI

PROFESOR: CORONILLA CONTRERAS UKRANIO

GRUPO: 7CM2

FECHA DE ENTREGA: 18 DE FEBRERO DEL 2024

Tarea 1: Fundamentos de java.

Elabore un documento en Word que contenga sus propios apuntes sobre la programación en lenguaje java para ser usados como material de consulta rápida durante el curso, el cual podrá ir ampliando durante el curso. Responda lo siguiente:

1.- ¿Qué es JDK, JRE y JVM?

JDK por sus siglas en ingles “Java Development Kit” es un software para los desarrolladores de java. Incluye el interprete, clases java y herramientas de desarrollo Java (JDT): compilador, depurador, desensamblador, visor de applets, generador de archivos apéndice y generador de documentación.

Este permite escribir aplicaciones que se desarrollan una sola vez y se ejecutan en cualquier lugar de cualquier máquina virtual Java.

JRE por sus siglas en ingles “Java Runtime Envitoment” es un conjunto de software que permite ejecutar aplicaciones de java. Proporciona todo lo necesario para que los programas funcionen en diferentes sistemas operativos.

Básicamente es como tener el reproductor de música para poder escuchar las canciones. Y normalmente ya viene incluido en el JDK pero si lo desea se puede instalar de forma separada.

JVM por sus siglas en ingles “Java Virtual Machine” es un componente fundamental de la plataforma Java. Es un programa que se encarga de ejecutar bytecode Java, que es el código que se genera cuando se compila un programa Java. Y de la misma manera que el JRE ya viene incluido en el JDK pero si lo desea se puede instalar de forma separada del igual manera.

2.- ¿Cuáles son los tipos de datos primitivos, cuenta memoria ocupa cada uno y cuáles son los rangos?

Tipo	Memoria	Rango
byte	1	-128 a 127
Short	2	-32768 a 32767
Int	4	-2147483648 a 2147483647
Long	8	-9223372036854775808 a 9223372036854775807
Float	4	3.4028235E38 a 1.4023982E-45
Double	8	1.7976931348623157E308 a 4.9406564584124654E-324
Boolean	1	True o false
char	2	0 a 65535

3.- ¿Qué es el casteo (casting) y para que se utiliza?

El casteo, también conocido como conversión de tipos, es una operación que permite convertir un valor de un tipo de dato a otro. En Java, existen dos tipos de casteo:

1. Casteo automático (widening): Es la conversión implícita que realiza Java de un tipo de dato a otro de mayor tamaño. Por ejemplo, un valor byte se puede convertir automáticamente a un valor int.

2. Casteo manual (narrowing): Es la conversión explícita que realiza el programador de un tipo de dato a otro de menor tamaño. Esta conversión puede ser peligrosa si no se realiza correctamente, ya que puede provocar la pérdida de información.

4.- ¿Qué es una clase y un objeto? (ejemplifica con un código breve y funcional)

En java una clase es una plantilla que define estructura y el comportamiento de un objeto. Especifica los datos y el código que operara en esos datos.

Un objeto en Java es una entidad que encapsula datos y comportamiento. Es como una plantilla que define las características y las acciones que puede realizar un tipo particular de dato.

```java
public class Coche {

    // Atributos
    private String marca;
    private String modelo;
    private String color;
    private int añoFabricacion;

    // Constructor
    public Coche(String marca, String modelo, String color, int añoFabricacion) {
        this.marca = marca;
        this.modelo = modelo;
        this.color = color;
        this.añoFabricacion = añoFabricacion;
    }

    // Métodos
    public void arrancar() {
        System.out.println("El coche " + marca + " " + modelo + " ha
arrancado.");
    }

    public void acelerar() {
        System.out.println("El coche " + marca + " " + modelo + " está
```

```
acelerando.");
    }

    public void frenar() {
        System.out.println("El coche " + marca + " " + modelo + " está
frenando.");
    }

    public void girar(String direccion) {
        System.out.println("El coche " + marca + " " + modelo + " está girando
hacia " + direccion);
    }

    // Método para obtener la marca
    public String getMarca() {
        return marca;
    }

    // Método para obtener el modelo
    public String getModelo() {
        return modelo;
    }

    // Método para obtener el color
    public String getColor() {
        return color;
    }

    // Método para obtener el año de fabricación
    public int getAñoFabricacion() {
        return añoFabricacion;
    }

}

public class Main {

    public static void main(String[] args) {

        // Crear un objeto de la clase Coche
        Coche coche1 = new Coche("Toyota", "Corolla", "Rojo", 2023);

        // Invocar métodos del objeto
        coche1.arrancar();
        coche1.acelerar();
        coche1.girar("izquierda");
        coche1.frenar();

        // Obtener información del objeto
        System.out.println("Marca: " + coche1.getMarca());
        System.out.println("Modelo: " + coche1.getModelo());
        System.out.println("Color: " + coche1.getColor());
```

```
        System.out.println("Año de fabricación: " + coche1.getAñoFabricacion());
    }
}
```

Explicación:

La clase Coche define los atributos y métodos de un coche.
El constructor de la clase Coche inicializa los atributos del objeto.
Los métodos de la clase Coche definen las acciones que el objeto puede realizar.
La clase Main crea un objeto de la clase Coche y llama a sus métodos.
El código también muestra cómo obtener información del objeto.

5.- ¿Qué son las clases wrapper, para que se usan y como se hacen las conversiones de datos primitivos a objetos mediante clases wrapper?

Las clases wrapper, también conocidas como envoltorios, son clases de encapsulan los tipos de dato primitivos de java.

Sus principales funciones son:

- Convertir tipos de datos primitivos a objetos que es útil cuando usamos tipos de datos primitivos en colecciones como un ArrayList, que solo puede almacenar objetos.
- Proporcionar métodos para trabajar con tipos de dato primitivos.
- Mejorar la legibilidad del código.

Ejemplo:

```
// Convertir un Integer a un int
int numeroInt = numero.intValue();

// Convertir un Double a un double
double decimalDouble = decimal.doubleValue();

// Convertir un Boolean a un boolean
boolean verdaderoBoolean = verdadero.booleanValue();
```

Clases Wrapper disponibles:

- Byte
- Short
- Integer

- Long
- Float
- Double
- Boolean
- Character

6.- ¿Cuál es la diferencia al almacenar en la memoria una variable local y una variable de tipo objeto? ¿Para qué sirve el garbage colector?

Variables locales:

- Se almacenan en la pila (stack) de memoria.
- Solo son accesibles dentro del bloque de código donde se declaran.
- Se eliminan automáticamente de la memoria cuando termina el bloque de código.

Variables de tipo objeto:

- Se almacenan en el heap de memoria.
- Son accesibles desde cualquier lugar del programa.
- No se eliminan automáticamente de la memoria. Se eliminan mediante el garbage collector.

El garbage collector es un proceso automático que se encarga de liberar la memoria que ya no está siendo utilizada por el programa. El garbage collector identifica los objetos que ya no son accesibles y los elimina de la memoria.

7.- ¿Qué son los arreglos y como se utilizan?

Un arreglo es una estructura de datos utilizada para almacenar datos del mismo tipo. Los arreglos almacenan sus elementos en ubicaciones de memoria contiguas.

En Java, los arreglos son objetos. Todos los métodos se pueden ser invocados en un arreglo. Podemos almacenar un número fijo de elementos en un arreglo.

8.- ¿Cómo se le pueden pasar argumentos al programa mediante el arreglo String[] args?

El método main de una clase java puede recibir argumentos desde la línea de comandos mediante el arreglo String[] args.

```java
public class Main {

    public static void main(String[] args) {
        // Imprimir los argumentos
        for (String arg : args) {
            System.out.println(arg);
        }
    }
}
```

Para ejecutar el programa escribimos en la línea de comando java main argumento1 argumento2

Y tendríamos en este caso de una forma muy simple la salida:
argumento1
argumento2

9.- ¿Qué es un package, para que se usan y como se importan?

Los paquetes en Java (packages) son la forma en la que Java nos permite agrupar de alguna manera lógica los componentes de nuestra aplicación que estén relacionados entre sí.

Los paquetes permiten poner en su interior casi cualquier cosa como: clases, interfaces, archivos de texto, entre otros.

Para importar un paquete en Java, se utiliza la palabra clave import seguida del nombre del paquete.

Tipos de importación:

- Importación simple: Se utiliza para importar una clase o interfaz específica.

import java.util.Scanner;
En este ejemplo, se importa la clase Scanner del paquete java.util.

- Importación con asterisco: Se utiliza para importar todas las clases e interfaces de un paquete.

import java.util.*;
En este ejemplo, se importan todas las clases e interfaces del paquete java.util.

10.- ¿Qué es la clase string y cuales con sus métodos más importantes?

La Clase String proporciona métodos para el tratamiento de las cadenas de caracteres: acceso a caracteres individuales, buscar y extraer una subcadena, copiar cadenas, convertir cadenas a mayúsculas o minúsculas, etc.

MÉTODO	**DESCRIPCIÓN**
length()	Devuelve la longitud de la cadena
indexOf(‘caracter’)	Devuelve la posición de la primera aparición de carácter
lastIndexOf(‘caracter’)	Devuelve la posición de la última aparición de carácter
charAt(n)	Devuelve el carácter que está en la posición n
substring(n1,n2)	Devuelve la subcadena comprendida entre las posiciones n1 y n2-1
toUpperCase()	Devuelve la cadena convertida a mayúsculas
toLowerCase()	Devuelve la cadena convertida a minúsculas
equals(«cad»)	Compara dos cadenas y devuelve true si son iguales
equalsIgnoreCase(«cad»)	Igual que equals pero sin considerar mayúsculas y minúsculas
compareTo(OtroString)	Devuelve 0 si las dos cadenas son iguales. <0 si la primera es alfabéticamente menor que la segunda ó >0 si la primera es alfabéticamente mayor que la segunda.
compareToIgnoreCase(OtroString)	Igual que compareTo pero sin considerar mayúsculas y minúsculas.
valueOf(N)	Método estático. Convierte el valor N a String. N puede ser de cualquier tipo.

11.- ¿Para qué sirve la palabra reservada this?

La palabra reservada this en Java se utiliza para:

1.- Referirse al objeto actual:

En una clase, this se utiliza para referirse al objeto actual en el que se está ejecutando el código. Esto permite acceder a los miembros (atributos y métodos) del objeto actual.

```java
public class Persona {

    private String nombre;

    public Persona(String nombre) {
        this.nombre = nombre;
    }

    public void saludar() {
        System.out.println("Hola, mi nombre es " + this.nombre);
    }
}
```

En este ejemplo, dentro del método saludar, this.nombre se utiliza para acceder al nombre del objeto actual.

2.- Invocar a otro constructor desde un constructor:

Se puede usar this para invocar a otro constructor desde un constructor dentro de la misma clase.

```java
public class Persona {

    private String nombre;
    private int edad;

    public Persona(String nombre) {
        this(nombre, 0);
    }

    public Persona(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

}
```

En este ejemplo, el constructor que recibe solo el nombre invoca al constructor que recibe nombre y edad.

3.- Evitar ambigüedades:

En caso de que exista un nombre de variable local que coincida con el nombre de un atributo de la clase, se puede usar this para acceder al atributo.

```
public class Persona {

    private String nombre;

    public void saludar(String nombre) {
        System.out.println("Hola, mi nombre es " + this.nombre + " y el tuyo es "
+ nombre);
    }
}
```

En este ejemplo, this.nombre se usa para acceder al atributo nombre de la clase, ya que existe una variable local con el mismo nombre.

12.- ¿Qué es la herencia? (ejemplificar con un código breve y funcional)

La herencia en Java es un mecanismo que permite a una clase (llamada clase hija o derivada) heredar las características (atributos y métodos) de otra clase (llamada clase padre o base).

Beneficios de la herencia:

- Reutilización de código: Permite reutilizar código ya escrito en la clase padre.
- Extensibilidad: Permite crear nuevas clases a partir de clases existentes.
- Mantenimiento: Facilita el mantenimiento del código.

```
public class Animal {

    private String nombre;

    public Animal(String nombre) {
        this.nombre = nombre;
    }

    public void comer() {
        System.out.println("El animal " + nombre + " está comiendo");
    }

    public void dormir() {
        System.out.println("El animal " + nombre + " está durmiendo");
    }
}
```

```java

public class Perro extends Animal {

    private String raza;

    public Perro(String nombre, String raza) {
        super(nombre); // Invoca al constructor de la clase padre
        this.raza = raza;
    }

    public void ladrar() {
        System.out.println("El perro " + nombre + " está ladrando");
    }
}

public class Main {

    public static void main(String[] args) {
        Perro perro = new Perro("Toby", "Labrador");

        perro.comer(); // Heredado de la clase Animal
        perro.dormir(); // Heredado de la clase Animal
        perro.ladrar(); // Propio de la clase Perro
    }
}
```

En este ejemplo, la clase Perro hereda de la clase Animal. Perro hereda los atributos nombre y los métodos comer() y dormir() de Animal. Además, Perro tiene un atributo propio raza y un método propio ladrar().

13.- ¿Qué es polimorfismo? (ejemplificar con un código breve y funcional)

El polimorfismo en Java es la capacidad que tienen los objetos de una clase en ofrecer respuesta distinta e independiente en función de los parámetros (diferentes implementaciones) utilizados durante su invocación.

Tipos de polimorfismo:

- Sobrecarga de métodos: Consiste en tener dos o más métodos con el mismo nombre pero con diferentes parámetros.

```java
public class Persona {

    public void saludar() {
        System.out.println("Hola!");
    }

```

```
    public void saludar(String nombre) {
        System.out.println("Hola " + nombre + "!");
    }
}

public class Main {

    public static void main(String[] args) {
        Persona persona = new Persona();

        persona.saludar(); // Imprime "Hola!"
        persona.saludar("Juan"); // Imprime "Hola Juan!"
    }
}
```

En este ejemplo, la clase Persona tiene dos métodos saludar(): uno sin parámetros y otro con un parámetro String.

- Sobreescritura de métodos: Consiste en redefinir un método de la clase padre en una clase hija.

```
public class Animal {

    public void comer() {
        System.out.println("El animal está comiendo");
    }
}

public class Perro extends Animal {

    @Override
    public void comer() {
        System.out.println("El perro está comiendo croquetas");
    }
}

public class Main {

    public static void main(String[] args) {
        Animal animal = new Perro();

        animal.comer(); // Imprime "El perro está comiendo croquetas"
    }
}
```

En este ejemplo, la clase Perro redefinió el método comer() de la clase Animal.

14.- ¿Qué es el @Override y para qué sirven los métodos toString() y equals()?

Un método anotado con @Override debe anular un método de una superclase. Si no lo hace, se producirá un error en tiempo de compilación. Se utiliza para asegurar que un método de superclase esté anulado y no simplemente sobrecargado. Esta es una anotación de marcador.

1. Método toString():

- Función: Convierte un objeto a una cadena de texto que lo representa.
- Utilidad: Se utiliza para mostrar información sobre un objeto en la consola, en archivos de texto o en interfaces gráficas.
- Implementación por defecto: La clase Object proporciona una implementación por defecto que devuelve la clase del objeto y su código hash.
- Sobreescritura: Se puede sobreescribir el método toString() en una clase para personalizar la representación textual del objeto.

```java
public class Persona {

    private String nombre;

    public Persona(String nombre) {
        this.nombre = nombre;
    }

    @Override
    public String toString() {
        return "Persona{" +
                "nombre='" + nombre + '\'' +
                '}';
    }
}

public class Main {

    public static void main(String[] args) {
        Persona persona = new Persona("Emmanuel");

        System.out.println(persona); // Imprime "Persona{nombre='Emmnauel'}"
    }
}
```

En este ejemplo, se sobreescribe el método toString() en la clase Persona para devolver una cadena que contiene el nombre de la persona.

2. Método equals():

- Función: Compara dos objetos para determinar si son iguales.
- Utilidad: Se utiliza para comparar objetos en colecciones, para verificar si dos objetos son el mismo objeto o para comparar dos objetos con el mismo contenido.
- Implementación por defecto: La clase Object proporciona una implementación por defecto que compara las referencias de los objetos.
- Sobreescritura: Se puede sobreescribir el método equals() en una clase para comparar los atributos de los objetos.

```java
public class Persona {

    private String nombre;

    public Persona(String nombre) {
        this.nombre = nombre;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }
        if (obj == null || getClass() != obj.getClass()) {
            return false;
        }
        Persona persona = (Persona) obj;
        return nombre.equals(persona.nombre);
    }
}

public class Main {

    public static void main(String[] args) {
        Persona persona1 = new Persona("Emmanuel");
        Persona persona2 = new Persona("Emmanuel");

        System.out.println(persona1 == persona2); // Imprime "false"
        System.out.println(persona1.equals(persona2)); // Imprime "true"
    }
}
```

En este ejemplo, se sobrescribe el método equals() en la clase Persona para comparar los nombres de las personas.

15.- ¿Qué es la sobrecarga del método? (ejemplificar con un código breve y funcional)

La sobrecarga de métodos es la creación de varios métodos con el mismo nombre pero con diferente lista de tipos de parámetros. Java utiliza el número y tipo de parámetros para seleccionar cuál definición de método ejecutar.

Java diferencia los métodos sobrecargados con base en el número y tipo de parámetros o argumentos que tiene el método y no por el tipo que devuelve.

También existe la sobrecarga de constructores: Cuando en una clase existen constructores múltiples, se dice que hay sobrecarga de constructores.

```
class Usuario4
{
      String nombre;
      int edad;
      String direccion;

      /* El constructor de la clase Usuario4 esta sobrecargado */
      Usuario4( )
      {
         nombre = null;
         edad = 0;
         direccion = null;
      }

      Usuario4(String nombre, int edad, String direccion)
      {
         this.nombre = nombre;
         this.edad = edad;
         this.direccion = direccion;
      }

      Usuario4(Usuario4 usr)
      {
         nombre = usr.getNombre();
         edad = usr.getEdad();
         direccion = usr.getDireccion();
      }

      void setNombre(String n)
      {
         nombre = n;
      }

      String getNombre()
      {
         return nombre;
      }
```

```
38
39      /* El metodo setEdad() está sobrecargado */
40      void setEdad(int e)
41      {
42         edad = e;
43      }
44
45      void setEdad(float e)
46      {
47         edad = (int)e;
48      }
49
50      int getEdad()
51      {
52         return edad;
53      }
54
55      void setDireccion(String d)
56      {
57         direccion = d;
58      }
59
60      String getDireccion()
61      {
62         return direccion;
63      }
64 }
```

16.- ¿Qué es el manejo de excepciones? (ejemplificar con un código breve y funcional)

El manejo de excepciones en Java es un mecanismo fundamental para controlar y gestionar errores en tiempo de ejecución. Permite que tu programa detecte, capture y gestione de forma robusta las excepciones que puedan surgir durante su ejecución, evitando que se bloquee o finalice inesperadamente.

Estas son eventos anómalos que ocurren durante la ejecución de un programa. Son objetos que derivan de la clase Exception y representan diferentes tipos de errores, como, por ejemplo:

- Errores de entrada/salida: por ejemplo, si se intenta leer un archivo que no existe.
- Errores de división por cero: si se intenta dividir un número por cero.
- Errores de formato: por ejemplo, si se intenta convertir una cadena a un número entero y la cadena no tiene un formato válido.
- Errores de lógica: por ejemplo, si se intenta acceder a un elemento de una lista que no existe.

```java
try {
    // Código que puede generar una excepción
} catch (ArithmeticException e) {
    // Manejar la excepción de división por cero
} catch (IOException e) {
    // Manejar la excepción de error de entrada/salida
} finally {
    // Liberar recursos
}
```

17.- ¿Para qué se utiliza static?

La palabra clave static en Java se utiliza para declarar miembros de clase que pertenecen a la clase en sí misma y no a una instancia individual de la clase. En otras palabras, los miembros static son compartidos por todas las instancias de la clase.

1. Variables:

- Declarar variables que no cambian, como constantes.
- Declarar variables que se comparten entre todas las instancias de la clase, como un contador global.

```java
public class MyClass {

    public static final int CONSTANTE = 10;
    private static int contador = 0;

    public void incrementarContador() {
        contador++;
    }

    public static int getContador() {
        return contador;
    }
}
```

2. Métodos:

- Declarar métodos que no necesitan acceder a variables de instancia, como métodos utilitarios.
- Declarar métodos que se pueden llamar sin crear una instancia de la clase, como métodos de fábrica.

```java
public class MyClass {

    public static int sumar(int a, int b) {
        return a + b;
    }

    public static String obtenerFechaActual() {
        return new Date().toString();
    }
}
```

Ventajas de usar static:

- Mejora la eficiencia: Solo se crea una copia de los miembros static en la memoria, lo que ahorra espacio y tiempo.
- Facilita el acceso: Se puede acceder a los miembros static sin necesidad de crear una instancia de la clase.
- Mejora la legibilidad del código: Se puede separar el código que depende del estado de una instancia del código que no depende del estado.

Desventajas de usar static:

- Dificulta la prueba de unidades: Los métodos static pueden ser difíciles de probar de forma unitaria, ya que no se pueden aislar del resto de la clase.
- Acopla el código: El uso excesivo de static puede acoplar el código, lo que dificulta su mantenimiento y reutilización.

Bibliografía.

IBM documentation. (2023, October 10). https://www.ibm.com/docs/es/i/7.3?topic=platform-java-development-kit

¿Qué es JRE (Java Runtime Environment)? | IBM. (n.d.). https://www.ibm.com/mx-es/topics/jre

Primitivo - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN. (2023, November 13). MDN Web Docs.
https://developer.mozilla.org/es/docs/Glossary/Primitive

De Roer, D. D., & De Roer, D. D. (2022, June 12). Casting en Java. Disco Duro de Roer -. https://www.discoduroderoer.es/casting-en-java/

González, D. B. (2023, December 4). Clases Wrapper (envoltorio) en Java. Profile Software Services. https://profile.es/blog/clases-wrapper-envoltorio-en-java/

¿Qué es el garbage collector en programación? (n.d.). EDteam - En Español Nadie Te Explica Mejor. https://ed.team/blog/que-es-el-garbage-collector-en-programacion

Acosta, E. V. (2023, January 30). Métodos arreglo de Java- Cómo imprimir arreglos en Java. freeCodeCamp.org.
https://www.freecodecamp.org/espanol/news/como-imprimir-arreglos-en-java/

González, J. D. M. (2021, June 20). Paquetes en Java.
https://www.programarya.com/Cursos/Java/Paquetes

11. Métodos de Clase String | Academia Códigos de Programación. (n.d.). https://codigosdeprogramacion.com/cursos/?lesson=11-metodos-de-clase-string

Ciberaula. (n.d.). 🏅🏅 Palabras Reservadas en Java.
https://www.ciberaula.com/cursos/java/palabras_reservadas_java.php

Mira, A. R. (2024, January 31). ¿Conoces las anotaciones en Java? Aquí te lo contamos. Tokio School. https://www.tokioschool.com/noticias/anotaciones-en-java/

Generación automática de los métodos toString y equals con Eclipse. (n.d.). https://www.tutorialesprogramacionya.com/javaya/detalleconcepto.php?punto=98&codigo=178&inicio=80

IV - Sobrecarga de metodos y de constructores. (n.d.). http://profesores.fi-b.unam.mx/carlos/java/java_basico4_6.html

Lesson: Exceptions (The JavaTM Tutorials > Essential Java Classes). (n.d.). https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html