Final Programación 2

¿Qué significa que la programación java sea multihilos?

-La capacidad de Java para ser "multihilos" se refiere a su capacidad para ejecutar múltiples hilos de ejecución simultáneamente en un programa. Un hilo es una secuencia de ejecución dentro de un proceso, y tener un entorno de programación multihilo significa que puedes tener varios hilos ejecutándose al mismo tiempo dentro de un programa Java.

¿Qué representa la referencia `this` y super en java?

En resumen, `this` se usa para referirse a la instancia actual de una clase, mientras que `super` se utiliza para acceder a los miembros de la clase padre desde una subclase.

¿Cuándo un lenguaje se dice orientado a objetos?

Significa que está diseñado principalmente en torno al concepto de "objetos". Significa que está diseñado principalmente en torno al concepto de "objetos".

Cuando incluye los cuatro conceptos fundamentales

-Clases y Objetos

Las clases son plantillas que definen las propiedades y comportamientos de los objetos. Los objetos son instancias de esas clases, que pueden contener datos (llamados atributos o propiedades) y métodos (funciones que operan en esos datos).

-Encapsulamiento

Es el concepto de ocultar el estado interno de un objeto y restringir el acceso a ciertas partes de un objeto. Esto se logra definiendo los atributos de un objeto como privados y proporcionando métodos públicos para acceder y modificar esos atributos.

-Herencia

Permite que una clase (llamada subclase o clase hija) herede propiedades y comportamientos de otra clase (llamada superclase o clase padre).

-Polimorfismo

o

¿Qué es el polimorfismo?

Es la capacidad de objetos de diferentes clases de responder al mismo mensaje o método. Permite que un método se comporte de manera diferente según el tipo de objeto al que se aplica.

¿En que orden debe escribirse las clausulas catch() del manejador de excepciones ?

-El orden recomendado es de las subclases hacia las superclases

comenzar con las excepciones mas especificas y luego las mas generales .

```
try{

int[] array = new int [5];

int valor = array[10];

}

catch (ArrayIndex.OutOfBoundsException ex)

{

System.out.println(” ”exception” atrapada”)

}

catch(Exception ex ){

System.out.println(”Exception general atrapada”);
```

```
}
```

¿Qué diferencia hay entre los calificadores final y static de un ejemplo de cada caso y recomiende donde los usaría?

`final`**:** Indica que una vez asignado un valor a una variable, este no puede ser cambiado.

final int numero = 10; // Una vez asignado, no se puede cambiar el valor de "numero"

En Java, el calificador `final` aplicado a clases se utiliza para indicar que una clase no puede ser heredada

`static`**:** Indica que un miembro (variable o método) pertenece a la clase en lugar de a las instancias individuales de esa clase.

Una variable estática pertenece a la clase en sí misma

Un método estático se asocia a la clase y no requiere una instancia de la clase para ser invocado.

```
class Ejemplo {
static int contador = 0; // Variable estática compartida entre todas las instancias de la clase
}

class Ejemplo {
static void metodoEstatico() {
// Código del método estático
}
}
```

¿Dónde se usarían?

`final`: Se utiliza para definir constantes o para garantizar que ciertos elementos no cambien, ya sea una variable, un método o una clase.

`static` **Se utiliza para crear miembros compartidos entre todas las instancias de una clase**

¿Qué diferencia existe entre las clausulas throw y throws del manejador de excepciones en java?

`throw` se utiliza para lanzar una excepción explícitamente dentro del código, mientras que `throws` se utiliza en la firma del método para indicar qué excepciones podría lanzar ese método, dejando la responsabilidad de manejar esas excepciones a quien llame al método.

Defina los siguientes tipos de variables soportados por java

Variables de instancia: Son variables asociadas a cada instancia u objeto creado a partir de una clase.

Variables de clase (o variables estáticas): Estas variables son compartidas por todas las instancias de una clase y se definen con la palabra clave `static`

Variables de referencia: Son variables que almacenan direcciones de memoria de objetos.

De un ejemplo de como se utiliza el constructor super() para Herencia .

```
class Vehiculo {
  int ruedas;
  public Vehiculo(int ruedas) {
      this.ruedas = ruedas;
      System.out.println("Se ha creado un vehículo con " + ruedas + " ruedas.");
```

```
  }
}
```

```
class Automovil extends Vehiculo {
String tipo;

  public Automovil(int ruedas, String tipo) {
      super(ruedas); // Llamada al constructor de la superclase Vehiculo
      this.tipo = tipo;
      System.out.println("Se ha creado un automóvil de tipo " + tipo);
  }
}
```

¿Qué son los Bytecodes?

Los bytecodes son un conjunto de instrucciones de bajo nivel y independientes de la plataforma en la que se ejecutan, diseñados para ser ejecutados por la máquina virtual de Java (JVM).

¿Qué diferencia existen entre los contenedores y arreglos?

En resumen, los arreglos son estructuras de datos estáticas con tamaño fijo y acceso rápido a elementos individuales, mientras que los contenedores son estructuras de datos dinámicas, más flexibles en tamaño y con métodos incorporados para manipular los elementos de manera más dinámica.

```
// Arreglo estático de tamaño fijo
int[] numeros = new int[5]; // Arreglo de enteros de tamaño 5

// Asignar valores al arreglo
numeros[0] = 10;
numeros[1] = 20;
numeros[2] = 30;
numeros[3] = 40;
numeros[4] = 50;

// Acceder a elementos del arreglo
System.out.println("El segundo elemento del arreglo es: " + numeros[1]); // Imprime: El segundo elemento del arreglo es: 20
```

```
import java.util.ArrayList;

// Crear un contenedor dinámico usando ArrayList
ArrayList<String> listaNombres = new ArrayList<>();

// Agregar elementos al contenedor
listaNombres.add("Juan");
listaNombres.add("María");
listaNombres.add("Carlos");

// Acceder a elementos del contenedor
System.out.println("El segundo nombre en la lista es: " + listaNombres.get(1)); // Imprime: El segundo nombre en la lista es: María

// Eliminar un elemento del contenedor
listaNombres.remove("María");
System.out.println("Lista después de eliminar a María: " + listaNombres); // Imprime: Lista después de eliminar a María: [Juan, Carlos]
```

¿Qué es un Hastable?

Hashtable es una **estructura de datos donde los datos se almacenan en un formato de matriz.**

-Características :

- **Pares clave-valor**
- **Eficiencia en la búsqueda**
- **No permite claves duplicadas**
- **No permite claves nulas**

- Como se recorre :

```
import java.util.Hashtable;

public class EjemploHashtable {
    public static void main(String[] args) {
        // Crear una Hashtable
        Hashtable<String, Integer> hashtable = new Hashtable<>();

        // Agregar elementos
        hashtable.put("A", 1);
        hashtable.put("B", 2);
        hashtable.put("C", 3);

        // Acceder a un elemento por clave
        int valor = hashtable.get("B");
        System.out.println("El valor asociado a la clave 'B' es: " + valor);

        // Eliminar un elemento por clave
        hashtable.remove("A");
        System.out.println("Hashtable después de eliminar 'A': " + hashtable);
    }
}
```

Cómo se recorre una Hashtable en Java:

```
// Recorrer la Hashtable
for (String clave : hashtable.keySet()) {
    int valor = hashtable.get(clave);
    System.out.println("Clave: " + clave + ", Valor: " + valor);
}
```