

## Programación Orientada a objetos

Actividad de consulta.

Nombre: Ronny Casco

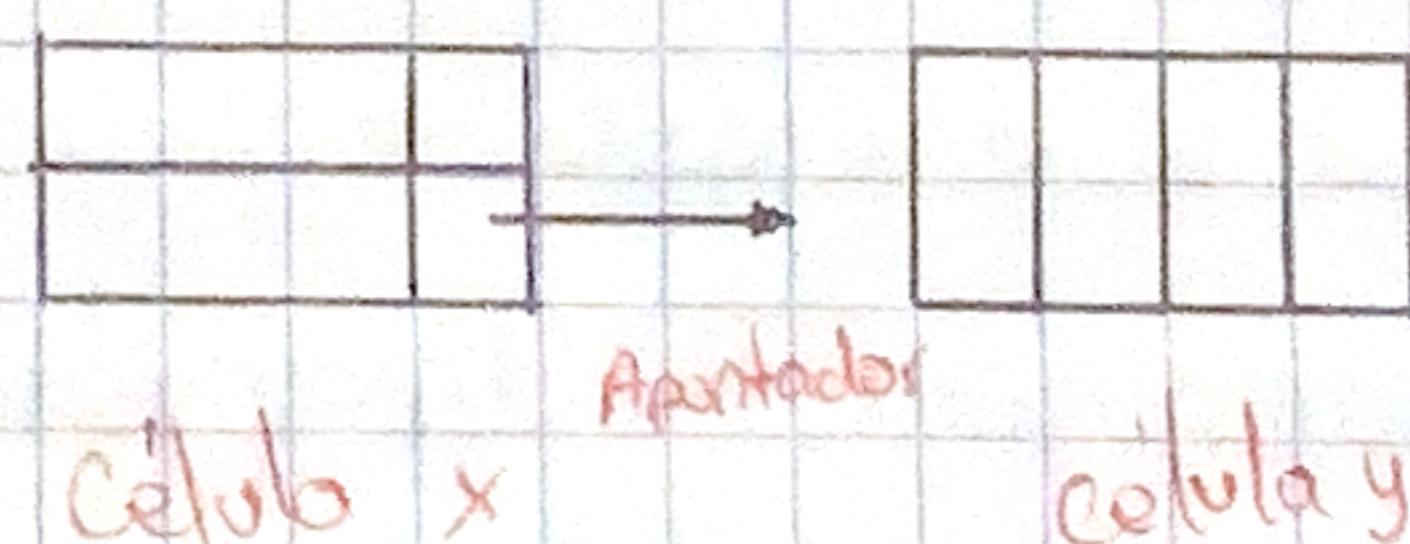
NRC: 1323

En un lenguaje de programación, el tipo de datos de una variable está determinado por el conjunto de valores que dicha variable puede tomar y el conjunto de operaciones que se pueden realizar con variables del mencionado tipo.

Una estructura de datos es una colección de variables (del mismo tipo o no), organizadas de alguna manera determinada. Se considera a la célula como la unidad básica de una estructura de datos. Además de la capacidad propia de un lenguaje de programación para agrupar las células de una estructura de datos, existe la posibilidad de crear estructuras relacionando e enlazando celdas usando apuntadores.

Un apuntador es un valor que dirige a una determinada célula, es decir que permite que

se pueda acceder a ella. Un apuntador puede implementarse como un cursor cuando las células son componentes de un arreglo, o como puntero si es que el lenguaje de programación cuenta con dicha facilidad.



## Tipo de Dato Abstracto (TDA)

Un (TDA) es dar el modelo y el conjunto de operaciones correspondientes, expresando con claridad y sin ambigüedad las características de cada una.

Por ejemplo, podría definirse el TDA de un número complejo de la siguiente forma:

- Modelo: por ordenado de números reales.
- Conjunto de operaciones: suma, diferencia, producto, modulo, argumento  
leer\_número, imprimir\_número
- Descripción precisa de cada uno de los operaciones.

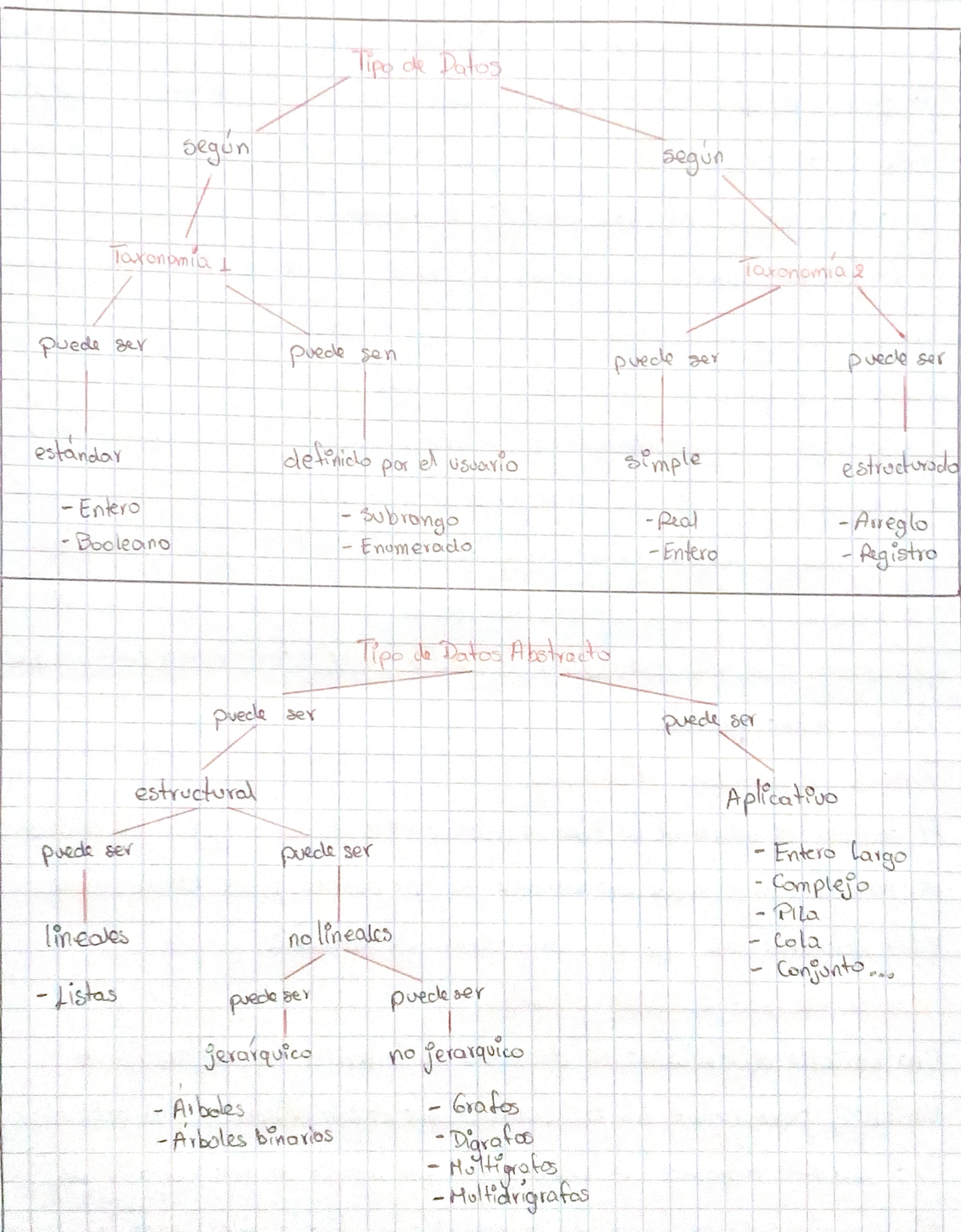
La definición de un TDA debe ser clara y precisa, ya que por un lado está el implementador del TDA quien tomará dicha descripción como base y la seguirá fielmente, y por otro lado está el usuario del TDA que lo usará teniendo en cuenta también lo que indica la definición. Obviamente, para que luego el programa funcione correctamente, ambas cosas deben concordar.

Las Definiciones de los TDA se hacen cuando se diseñan los algoritmos, mientras que cuando se implementan los algoritmos en un determinado lenguaje de programación es cuando también se implementan los TDA que se han definido previamente, si es que no se cuenta con su implementación cabe señalar que si el lenguaje de programación con el que se implementa el algoritmo cuenta entre sus tipos estándar con uno que concuerda perfectamente con la definición realizada para un determinado TDA, se podrá usar dicho tipo sin necesidad de realizar una implementación especial para el TDA en cuestión.



# ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS  
INNOVACIÓN PARA LA EXCELENCIA



## Estructuras Dinámicas (ED)

Existen estructuras de datos, como los arreglos o los registros, para los



cuales se conoce en tiempo de compilación el espacio que ocuparán durante la ejecución del procedimiento (o programa) donde se encuentran declaradas.

Además de conocerse con anticipación la cantidad de memoria que ocuparán durante la ejecución del procedimiento (o programa) al comenzarse a ejecutar el procedimiento donde se encuentran declaradas, se reserva el espacio de memoria correspondiente y se mantiene reservado mientras dura la ejecución de dicho procedimiento, independientemente de la necesidad de su uso.

Para mejorar su entendimiento se dice que una estructura es dinámica cuando el espacio de almacenamiento en memoria se va obteniendo durante la ejecución, a medida que se lo necesita. Por lo tanto la cantidad máxima de memoria que ocupará la estructura de datos durante la ejecución del procedimiento donde está declarada, no se conoce hasta el momento en que finaliza dicha ejecución.

### Punteros y variables dinámicas.

El conjunto de valores de un tipo de datos puntero es un conjunto de direcciones de memoria. Es decir que una variable de tipo puntero puede tomar como valor únicamente una de esas direcciones de memoria.

### Creación de una variable dinámica

Una variable dinámica no es declarada en la sección de declaración de Variables, como ocurre con las variables que no son dinámicas (variables estáticas).

Una variable dinámica es referenciada por medio de una variable de tipo puntero es decir que en la variable de tipo puntero está la dirección de memoria de la variable dinámica.



# ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS  
INNOVACIÓN PARA LA EXCELENCIA

## Típos de datos - Típos Primitivos

Consideramos que las variables son entidades elementales: un número, un carácter, un valor verdadero o falso... mientras que los objetos son entidades complejas que pueden estar formadas por la agrupación de muchas variables y métodos. Pero ambas cosas ocupan lo mismo: un espacio de memoria (que puede ser más o menos grande)

En los programas en java puede ser necesario tanto el uso de datos elementales como datos complejos. Por eso en java se utiliza el término "Tipos de datos" para englobar a cualquier cosa que ocupa un espacio de memoria y que puede ir tomando distintos valores o características durante la ejecución del programa. Es decir, en vez de hablar de tipos de variables o de tipos de objetos, hablaremos simplemente de tipos de datos.

1. Un objeto es una cosa distinta a un tipo primitivo, aunque "porten" la misma información. Tener siempre presente que los objetos en Java tienen un tipo de tratamiento y los tipos primitivos otro.

O sea en un momento dado contengan la misma información no significa en ningún caso que sean lo mismo.

2. ¿Para qué tener esa aparente duplicidad entre tipos primitivos y tipos envoltorio? Esto es una cuestión que atañe a la concepción del lenguaje de programación. Tener en cuenta una cosa: un tipo primitivo es un dato elemental y carece de métodos, mientras que un objeto es una entidad compleja y dispone de métodos.

3. Los nombres de tipos primitivos y envoltorio se parecen mucho. En realidad, excepto entre int e integer y char y Character, la diferencia se



l<sup>o</sup>mita a que en un caso la inicial es minúscula.

4. Una cadena de caracteres es un objeto. El tipo String en Java nos permite crear objetos que contienen objetos (texto, palabras, frases).

5. Hay distintos tipos primitivos enteros. ¿Cuál usar? Por norma general usaremos el tipo int. Para los casos en los que el entero pueda ser muy grande usaremos el tipo long.

6. ¿Cuántos tipos de la biblioteca estándar de Java hay? Hay muchos cientos, miles. Es imposible conocerlos todos.

7. ¿Un array es un objeto? Los arrays los consideramos objetos especiales, los únicos objetos en Java que carecen de métodos.

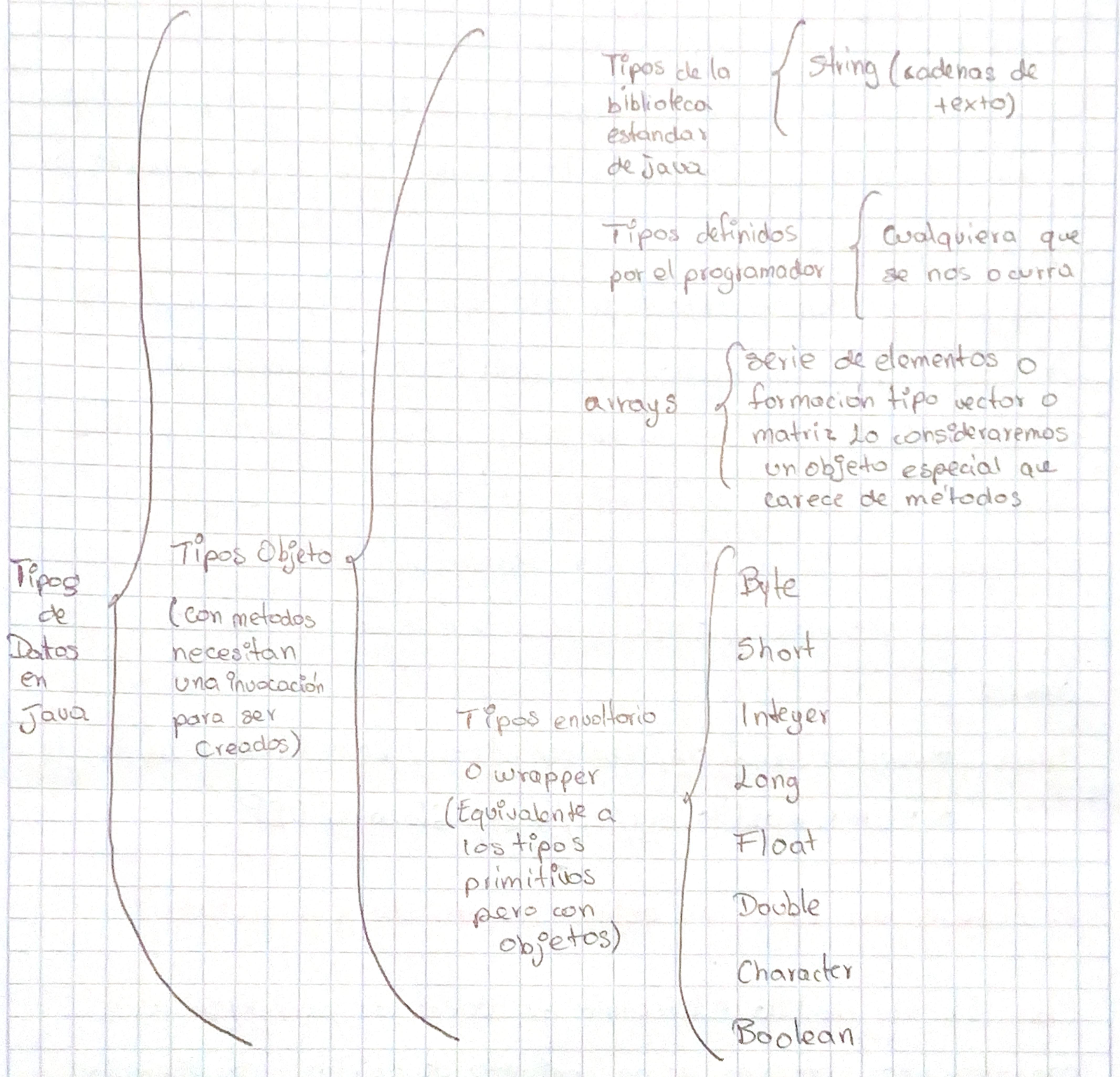
Nombre	Tipo	Ocupa	P.Aproximado
byte	Entero	1 byte	-128 a 127
short	Entero	2 bytes	-32768 a 32767
int	Entero	4 bytes	$2 \times 10^9$
long	Entero	8 bytes	Muy grande
float	Decimal simple	4 bytes	Muy grande
double	Decimal doble	8 bytes	-
char	Carácter simple	2 bytes	-
bodean	Valor true o false	1 byte	-

Tipos de  
Datos  
en  
Java

Tipos  
Primitivos  
sin métodos;  
no son objetos;  
no necesitan una  
invocación para  
crearlos

Fuentes: CS.uns.edu.ar/~ldm/data/eda/apuntes/o1\_t01-ed-t01.pdf

Aprenderaprogramar.com/index.php?option=com\_content&view=category&id=68&Itemid=188



Respondo las siguientes preguntas.

1. ¿Qué es el paradigma de programación orientada a objetos?

El paradigma de programación orientada a objetos, es un modelo de programación que organiza el software en torno a objetos, en lugar de funciones o lógicas. Cada objeto representa una entidad del mundo real con atributos (datos) y métodos (comportamientos). Este enfoque permite crear sistemas modulares, reutilizables y fáciles de mantener.

La POO se basa en cuatro principios clave:

1. **Encapsulación**: Los datos y métodos de un objeto están protegidos y solo son accesibles mediante interfaces específicas, lo que aumenta la seguridad y reduce errores.
2. **Abstracción**: Permite interactuar con objetos complejos mediante interfaces simplificadas ocultando los detalles de implementación.
3. **Herencia**: Facilitar la creación de nuevas clases a partir de otras existentes, reutilizando atributos y métodos.
4. **Polimorfismo**: Los objetos pueden adoptar múltiples formas, lo que permite tratar diferentes tipos de objetos de manera uniforme según el contexto.

Fuente: [www.luis.izqui.org/resources/ProgOrientadaObjetos.pdf](http://www.luis.izqui.org/resources/ProgOrientadaObjetos.pdf)  
[developer.mozilla.org/es/docs/Glossary/ODP](https://developer.mozilla.org/es/docs/Glossary/ODP)

2. ¿Qué es una clase, un objeto, un atributo y un método?

**Clase**: Es una plantilla o blueprint que define las características y comportamientos comunes de un conjunto de objetos. Por ejemplo, una clase "Teléfono Móvil" puede incluir atributos como "marca iphone" "modelo XR" y métodos como "llamar" o "ver videos" es una forma de agrupar datos y comportamientos relacionados en una unidad lógica.

**Objeto**: Es una instancia específica de una clase. Cada objeto tiene sus propios valores para los atributos definidos en la clase. Por ejemplo un objeto de la clase "smartphone" puede ser un smartphone



específico con la marca "iphone" y el modelo "XR"

**Atributo:** Son las variables que almacenan datos dentro de un objeto. Define las características del objeto. Por ejemplo, "color" y "almacenamiento" podrían ser atributos de un objeto de la clase "smartphone".

**Método:** Son funciones definidas dentro de una clase que describe los comportamientos de los objetos. Los métodos pueden acceder y modificar los atributos de un objeto. Por ejemplo un método "encendido" puede cambiar el estado del atributo "encendido de un smartphone".

**Fuente:** [diveintopython.org/es/learn/classes/object-instantiation](https://diveintopython.org/es/learn/classes/object-instantiation)

3. ¿Qué es un sistema de control de versionamiento y para qué sirve?

Un Sistema de control de versiones (VCS) es una herramienta que permite gestionar los cambios realizados en los archivos de un proyecto, rastreando el historial de modificaciones y facilitando la colaboración entre varios desarrolladores. Los VCS son esenciales en proyectos de software porque ayudan a mantener un registro claro de las contribuciones individuales, permiten revertir cambios si es necesario y aseguran la integridad del trabajo en equipo.

Los sistemas de control de versiones pueden ser locales, centralizados o distribuidos. Git es uno de los VCS distribuidos más populares, permitiendo a los desarrolladores trabajar de manera independiente en copias locales de un proyecto y luego sincronizar sus cambios con un repositorio remoto. Esto mejora la flexibilidad, la colaboración y la resolución de conflictos.

La naturaleza es el mejor maestro.

BADMINTON BASICO PARA INICIANTES



Fuente: free code camp.org/espanol/news/git-vs-github-what-is-version-control-and-how-does-it-work/

4. Realiza 3 UML de 2 clases hijo y un objeto padre

Clase Padre

Empleado

- Atributos: Nombre, Salario
- Métodos: Calcular Salario y Registrar Asistencia

Clase hijo

Jefe

- Atributos: Oficina
- Método: Supervisión

Clase hijo

Técnico

- Atributos: Especialidad
- Método: Mantenimiento

Clase Padre

Figuras Geométricas

- Atributos: color, posición
- Métodos: calcular Área, Calcular Perímetro

Clase hijo

Círculo

Atributo: Radio  
Método: Calcular Diámetro

Clase hijo

Rectángulo

Atributo: base, altura  
Método: calcular diagonal.

Clase Padre

Dispositivos Electrónicos

Atributos: marca, modelo, batería  
Métodos: encender, apagar

Clase hijo

Teléfono

Atributo: número SIM  
Método: hacer llamada

Clase hijo

Computadora

Atributo: Tipo de procesador  
Método: Ejecutar programas.