🟢 1️⃣ ¿Cuándo usar List?

Una **List** es:

👉 Colección ordenada
👉 Permite duplicados
👉 Acceso por índice

Ejemplo mental:
“Quiero una lista de tareas en orden”.

List<String> tareas = new ArrayList<>();

- 🔹 **ArrayList**
  - Rápido para acceder por índice
  - Malo para insertar en medio
- 🔹 **LinkedList**
  - Bueno para insertar/eliminar al principio o medio
  - Peor acceso por índice

📌 ¿Cuándo usar LinkedList?

Casi nunca.

Solo si:

- Vas a insertar/eliminar mucho en medio
- No necesitas acceso frecuente por índice

En el 90% de casos → usa ArrayList.

LinkedList está sobrevalorada en teoría y poco usada en la vida real.

🔵 2️⃣ ¿Cuándo usar Set?

Un **Set** es:

👉 No permite duplicados
👉 No tiene índice
👉 La igualdad la define equals() + hashCode()

Ejemplo:
“Quiero un conjunto único de usuarios”.

Set<String> usuarios = new HashSet<>();

- 🔹 **HashSet**

- No mantiene orden
- Muy rápido

🔹 **LinkedHashSet**

- Mantiene orden de inserción

🔹 **TreeSet**

- Ordena automáticamente

📌 Usa Set cuando:

- No quieres duplicados
- Te importa la unicidad

🟣 3️⃣ **¿Cuándo usar Map?**

Un **Map** es:

👉 Clave → Valor
👉 No hay claves duplicadas

Ejemplo:
“Quiero buscar un planeta por su nombre”.

Map<String, Planeta> sistemaSolar = new HashMap<>();

🔹 **HashMap**

- Muy rápido
- Sin orden

🔹 **LinkedHashMap**

- Mantiene orden de inserción

🔹 **TreeMap**

- Ordena por clave

📌 Usa Map cuando:

- Necesitas búsqueda rápida por clave
- Tienes relación clave → objeto

🔥 **Ahora lo serio: equals() y hashCode()**

Si usas:

- HashSet
- HashMap
- LinkedHashSet
- LinkedHashMap

Entonces equals y hashCode son obligatorios.

🔹 **equals()**

Define cuándo dos objetos son "iguales".

Ejemplo:

```
@Override
public boolean equals(Object o) {
   if (this == o) return true;
   if (o == null || getClass() != o.getClass()) return false;

   Persona persona = (Persona) o;
   return Objects.equals(nombre, persona.nombre);
}
```

🔹 **hashCode()**

Devuelve un número que representa al objeto.

```
@Override
public int hashCode() {
   return Objects.hash(nombre);
}
```

🧠 **Regla sagrada**

Si sobrescribes equals → debes sobrescribir hashCode.

Siempre.

🧠 **¿Por qué?**

Porque HashSet funciona así:

1️⃣ Calcula hashCode
2️⃣ Va al bucket correspondiente
3️⃣ Luego compara con equals

Si hashCode no coincide, ni siquiera mira equals.

🔥 **Error clásico**

Dos objetos parecen iguales:

new Persona("Ana")
new Persona("Ana")

Pero si no implementas hashCode correctamente:

set.add(persona1);
set.add(persona2);

Tendrás duplicados.

Y llorarás.

📌 **Resumen brutalmente claro**

Estructura	Permite duplicados	Mantiene orden	Acceso por clave
List	Sí	Sí	Índice
Set	No	Depende	No
Map	No claves	Depende	Sí