Apunte Actividad 4

Recursividad, Arrays y Listas

1. ¿Qué es la recursividad?

La **recursividad** es una técnica en la que una función se **llama a sí misma** para resolver un problema.
Se utiliza dividiendo el problema en **subproblemas más pequeños**, hasta llegar a un **caso base** que detiene la recursión.

Estructura típica de una función recursiva:

```
tipoRetorno funcion(parametros) {
  if (condicionBase) {
    return resultado; // Caso base
  } else {
    return funcion(modificacionDeParametros); // Llamada recursiva
  }
}
```

Dos partes clave:

- **Caso base:** condición que detiene la recursión.
- **Llamada recursiva:** el método se invoca nuevamente con un problema más pequeño.

2. Recursividad vs. Iteración

- **Iteración (bucles):** repite instrucciones mientras se cumpla una condición.
- **Recursividad:** se basa en llamadas repetidas a la misma función.

- *Iteración suele ser más eficiente en memoria.*
- *Recursividad puede dar soluciones más elegantes y fáciles de entender.*

3. Ejemplos de recursividad en Java

Factorial

```java
public int factorial(int n) {
    if (n == 0) { // caso base
        return 1;
    }
    return n * factorial(n - 1); // llamada recursiva
}
```

Serie de Fibonacci

```java
public int fibonacci(int n) {
    if (n == 0) {
        return 0;  // caso base
    }
    if (n == 1) {
        return 1;  // caso base
    }
    return fibonacci(n - 1) + fibonacci(n - 2);
}
```

Búsqueda Binaria (en un array ordenado)

```java
public int busquedaBinaria(int[] arr, int inicio, int fin, int valor) {
    if (inicio > fin) {
        return -1; // no encontrado
    }
    int medio = (inicio + fin) / 2;
    if (arr[medio] == valor) {
        return medio;
    } else if (valor < arr[medio]) {
        return busquedaBinaria(arr, inicio, medio - 1, valor);
    } else {
        return busquedaBinaria(arr, medio + 1, fin, valor);
    }
}
```

4. Arreglos (Arrays) en Java

Un **array** es una estructura de datos que almacena elementos del **mismo tipo** en posiciones consecutivas de memoria.

Declaración y uso

```java
int[] numeros = new int[5]; // array de 5 enteros
numeros[0] = 10; // asignación
System.out.println(numeros[0]); // acceso
```

Inicialización rápida

```java
String[] nombres = {"Ana", "Pedro", "Luis"};
```

- Los arrays tienen un tamaño **fijo** que se define al crearlos.

5. Listas en Java

Cuando se necesita almacenar datos de manera **dinámica**, se utilizan las **listas** de la librería java.util.

Ejemplo con ArrayList

```
import java.util.ArrayList;

ArrayList<String> lista = new ArrayList<>();

lista.add("Manzana");
lista.add("Banana");
lista.add("Naranja");

System.out.println(lista.get(1)); // imprime "Banana"
lista.remove("Banana"); // elimina elemento
```

Ventajas frente a los arrays:

- Pueden crecer y reducir su tamaño dinámicamente.
- Tienen métodos útiles como add(), remove(), size(), contains().

6. Cuándo usar Arrays y Listas

- **Array:** cuando se sabe de antemano el número fijo de elementos y se necesita alta eficiencia.
- **Lista (ArrayList, LinkedList):** cuando el número de elementos puede variar o se requiere flexibilidad.