Resumen/Esquema para Dominar los Puntos del Examen de 1º DAM

---

1. Uso de Librerías

Definición

Las librerías en Java son colecciones de clases y métodos predefinidos que facilitan tareas comunes, evitando la necesidad de escribir código desde cero.

Algunas librerías estándar importantes:

- `java.util` → Estructuras de datos (ArrayList, HashMap, etc.).
- `java.io` → Entrada y salida de datos (archivos, flujo de datos).
- `java.time` → Manejo de fechas y horas.
- `java.util.regex` → Expresiones regulares para validaciones de texto.

Cómo usarlas

1. Importar la librería necesaria con `import`.
2. Usar las clases y métodos provistos en el código.

Ejemplo (Lectura de entrada del usuario con `Scanner`):

```
import java.util.Scanner;

public class EjemploLibreria {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Ingresa tu nombre: ");
        String nombre = scanner.nextLine();
        System.out.println("Hola, " + nombre + "!");
        scanner.close();
    }
}
```

Consejos para el examen

✅ Familiarízate con las librerías estándar más utilizadas. ✅ Practica importar y usar clases como `ArrayList`, `Scanner`, o `LocalDate/Time`. ✅ Aprende a leer y escribir archivos con `java.io`.

---

2. Creación de Clases

Definición

Una **clase** en Java es un modelo o plantilla para crear objetos, definiendo atributos (variables) y métodos (comportamientos).

Elementos principales

- **Atributos** → Variables de la clase (públicas, privadas, etc.).
- **Métodos** → Funciones que operan sobre los atributos.
- **Constructores** → Métodos especiales para inicializar objetos.

Ejemplo (Definir una clase `Persona` con atributos y métodos):

```
public class Persona {
    private String nombre;
    private int edad;

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

    // Método para mostrar información
    public void mostrarInfo() {
        System.out.println("Nombre: " + nombre + ", Edad: " + edad);
    }
}
```

Consejos para el examen

✅ Practica definir clases con atributos privados y métodos públicos. ✅ Aprende a instanciar objetos y usar constructores correctamente. ✅ Entiende la diferencia entre clases públicas, abstractas e interfaces.

---

3. Herencia

Definición

La **herencia** permite que una clase (subclase) herede atributos y métodos de otra clase (superclase) usando `extends`.

Ventajas

- **Reutilización de código**.
- **Polimorfismo** (sobrescribir métodos para modificar comportamientos).

Ejemplo (Clase `Animal` y subclase `Perro`):

```
public class Animal {
    protected String nombre;

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

    public void hacerSonido() {
        System.out.println("Sonido genérico");
    }
}

public class Perro extends Animal {
    public Perro(String nombre) {
        super(nombre);
    }

    @Override
    public void hacerSonido() {
        System.out.println(nombre + " dice: ¡Guau!");
    }
}
```

Consejos para el examen

✅ Entiende el uso de `super` para llamar al constructor de la superclase. ✅ Practica el uso de `@Override` para sobrescribir métodos. ✅ Aprende sobre interfaces para implementar herencia múltiple en Java.

---

4. Igualdad y Comparación

Definición

- `==` → Compara referencias en memoria.
- `equals()` → Compara el contenido de los objetos (debe sobrescribirse en la clase).
- `compareTo()` → Usado en `Comparable` para ordenar objetos.

Ejemplo (Sobrescribir `equals()` y `compareTo()` en `Persona`):

```
public class Persona implements Comparable<Persona> {
    private String nombre;
    private int edad;

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

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

    @Override
    public int compareTo(Persona otra) {
        return this.nombre.compareTo(otra.nombre);
    }
}
```

Consejos para el examen

✅ Practica sobrescribir `equals()` y `hashCode()` juntos. ✅ Comprende la diferencia entre `==` y `equals()`. ✅ Aprende a usar `Comparable` y `Comparator` para ordenar objetos.

---

5. Estructuras de Datos y Ordenación

Estructuras más usadas

- **Arrays** (`int[]`, `String[]`).
- **Listas** (`ArrayList`, `LinkedList`).
- **Conjuntos** (`HashSet`, `TreeSet`).
- **Mapas** (`HashMap`, `TreeMap`).

Ejemplo (Ordenar una lista con `Collections.sort()`):

```
import java.util.ArrayList;
import java.util.Collections;
```

```
public class EjemploEstructuras {
    public static void main(String[] args) {
        ArrayList<Integer> numeros = new ArrayList<>();
        numeros.add(5);
        numeros.add(2);
        numeros.add(8);

        Collections.sort(numeros);
        System.out.println("Números ordenados: " + numeros);
    }
}
```

Consejos para el examen

✅ Practica usar `ArrayList`, `HashSet`, `HashMap`. ✅ Aprende a ordenar con `Collections.sort()` o `Arrays.sort()`. ✅ Entiende la diferencia entre estructuras ordenadas (`TreeSet`, `TreeMap`) y no ordenadas (`HashSet`, `HashMap`).

---

6. Expresiones Regulares (Regex)

Ejemplo (Validar email con `Pattern` y `Matcher`):

```
import java.util.regex.Pattern;
import java.util.regex.Matcher;

public class EjemploRegex {
    public static void main(String[] args) {
        String email = "usuario@dominio.com";
        String regex = "^[A-Za-z0-9+_.-]+@(.+)$";

        Pattern pattern = Pattern.compile(regex);
        Matcher matcher = pattern.matcher(email);

        if (matcher.matches()) {
            System.out.println("Email válido!");
        } else {
            System.out.println("Email no válido.");
        }
    }
}
```

Consejos para el examen

✅ Aprende patrones básicos (`[A-Za-z]`, `\d`). ✅ Practica validar formatos comunes (emails, números de teléfono, etc.).

¡Practica, revisa errores comunes y repasa estos conceptos clave para el examen!