1️⃣

Métodos y atributos

▼ Declaración y uso de métodos

Un método es un trozo de código que puede ser llamado o invocado por el programa principal o por otro método para realizar alguna tarea específica. El término método en Java es equivalente al de subprograma, rutina, subrutina, procedimiento o función en otros lenguajes de programación. El método es llamado por su nombre o identificador seguido por una secuencia de parámetros o argumentos (datos utilizados por el propio método para sus cálculos) entre paréntesis. Cuando el método finaliza sus operaciones, devuelve habitualmente un valor simple al programa que lo llama, que utiliza dicho valor de la forma que le convenga. El tipo de dato devuelto por la sentencia return debe coincidir con el tipo de dato declarado en la cabecera del método.

Sintaxis de declaración de un método:

```
[modificadores] TipoDeDato nombreMetodo (parametros formales) {
  declaraciones de variables locales;
  sentencia1;
  sentencia2;
  ...
  sentencian;
  //dentro de estas sentencias se incluye al menos un return
}
```

Seguidamente, se muestra un ejemplo de declaración y uso de un método

El método saludar en el ejemplo, requiere pasar una variable String, por lo que en su llamada y en su definición las variables saludo y cadena han de coincidir en el tipo. Los métodos que no

```
//Ejemplo seudocodigo
public class HolaMundo {
  public static void main(String[] args) {
    String saludo="hola";
    saludar(saludo);
  }
  public static void saludar(String cadena) {
    System.out.println(cadena);
```

devuelven ningún valor no llevan sentencia return y en el tipo de dato a devolver se especifica void (“vacío”).

```
  }
}
```

▼ Ejemplos de llamadas a métodos según sus parámetros:

Ejemplo1: El método no requiere paso de parámetros, por lo que en la llamada, entre paréntesis, no se especifica ninguna variable. Al no devolver ningún valor, el método se define especificando void (“vacio”) en la cabecera de la definición.

```
//Ejemplo 1 seudocodigo
public class HolaMundo {
  public static void main(String[] args) {
    //Llamada al metodo
    ejemplo1();
  }
  //Definicion del metodo
  public static void ejemplo1() {
    System.out.println("Hola mundo");
  }
}
```

```
//Ejemplo 2 seudocodigo
public class HolaMundo {
  public static void main(String[] args) {
    //Llamada al metodo
    String aux=ejemplo2();
    System.out.println(aux);
  }
  //Definicion del metodo
  public static String ejemplo2() {
    String saludo="Hola mundo";
    return saludo;
  }
}
```

Ejemplo2: Al igual que ejemplo1, no requiere enviar parámetros al método, por lo que en su llamada no se especifican variables dentro de los paréntesis. Sin embargo, en la definición se especifica que devuelve una variable tipo String. Esta variable se llama “saludo”, local al método ejemplo2, no es accesible fuera del propio método. Al devolver una variable tipo String, se debe especificar mediante la sentencia “return” la variable a

devolver. En el ejemplo el valor de saludo se almacena en aux, siempre que devolvamos algún valor se debe almacenar u operar con el.

Ejemplo3: Al método se le pasa una variable “saludo” de tipo String, por lo que en su definición se ha de especificar que recibe una variable del mismo tipo (en el ejemplo String cadena) dentro de los paréntesis. Como no devuelve ningún valor, se define como void al tipo de dato a devolver y no se añade la sentencia return.

```
//Ejemplo 3 seudocodigo
public class HolaMundo {
  public static void main(String[] args) {
    String saludo="Hola";
    //Llamada al metodo
    ejemplo3(saludo);
  }
  //Definicion del metodo
  public static void ejemplo3(String cadena) {
    System.out.println(cadena+"Mundo");
  }
}
```

```
//Ejemplo 4 seudocodigo
public class HolaMundo {
  public static void main(String[] args) {
    String saludo="Hola";
    //Llamada al metodo
    String aux=ejemplo4(saludo);
    System.out.println(aux);
  }
  //Definicion del metodo
  public static String ejemplo4(String cadena) {
    String variable="Mundo";
    return cadena+variable;
  }
}
```

Ejemplo4: Este método requiere pasar un String en su llamada como devolver un String, por ello, en su definición se especifica el tipo de dato a devolver String y entre paréntesis la variable String cadena que

admite en la llamada(“saludo”). Al devolver un String es necesario también que el método cuente con la sentencia return.

▼ Sobrecargar de métodos

Java permite asignar el mismo identificador a distintos métodos, cuya diferencia reside en el tipo o número de parámetros que utilicen. Esto resulta especialmente conveniente cuando se desea llevar a cabo la misma tarea en diferente número o tipos de variables. La sobrecarga (overloading) de los métodos puede resultar muy útil al efectuar llamadas a un método, ya que en lugar de tener que recordar identificadores de métodos distintos, basta con recordar uno solo. El compilador se encarga de averiguar cuál de los métodos que comparten identificador debe ejecutar.

```
//Ejemplo seudocodigo
public class HolaMundo {
  public static void main(String[] args) {
    String saludo="Hola";
    saludar();
    saludar(saludo);
  }
  public static void saludar() {
    System.out.println("Hola mundo");
  }
  public static void saludar(String cadena) {
    System.out.println(cadena+" mundo");
  }
}
```