2.2_1

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		boolean b = false;
		int i = (int) b;	No se puede convertir un booleano a intenger
	}

}
```

2.2_2

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		boolean condition1 = false;
		boolean condition2 = true;

		int valueCondition1 = booleanToInt(condition1);
		int valueCondition2 = booleanToInt(condition2);

		System.out.println(condition1 + " = " + valueCondition1);
		System.out.println(condition2 + " = " + valueCondition2);
	}

	public static int booleanToInt(boolean value) {
      // Convert true to 1 and false to 0.
		return value ? 1 : 0;
		/*
		if(value) {
			return 1;
		} else{
			return 0;
		}*/
 }

}
```

2.3_1

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		int a = 1, b = 1, c = 3;
		System.out.println(a == b); // En este caso, el resultado de la condiciones es TRUE, ya que A y B son
									// iguales

		boolean resultado = (a == c); // Variable de tipo boolean en la que almacenamos el resultado de comparar si A y C son iguales (==).

		System.out.println(resultado); // En este caso, el resultado de la condiciones es FALSE, ya que A y C no son iguales

		System.out.println((b != c)); // En este caso, el resultado de la condicion es TRUE, ya que B no es igual (=!) a C

		resultado = (1 < 0);
		System.out.println(resultado); // Compara si 1 es menor que 0, en este caso retorna false ya que 1 es más grande que 0

		resultado = (1 == a && 2 == 2); // Comparando dos condiciones
		System.out.println(resultado); // Compara si 1 es igual a 1 Y a su vez si 2 es igual a 2. Como ambas son TRUE, retorna TRUE

	}
}
```

2.5_1

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		// Lo que va entre parentesis es la condición. En este caso,
		// al ser true, el resultado de la condición retornará true
		// y por tanto, ejecutará el bloque englobado entre {}
		// en caso de ser false, no se ejecutará el bloque englobado entre {}
		if(true) {
			/*
			 * Instrucción 1
			 * Instrucción 2
			 *  ...
			 */
		}
	}
}
```

2.5_2

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		double nota = 3;
		System.out.println(nota < 5); // En este caso, el resultado de dicha
condición será false
		// Solamente ejecutará las instrucciones que están dentro del bloque {}
cuando la condición retorne true
		if(nota < 5) {
			System.out.println("Suspendido");
		}
	}

}
```

2.6_1

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		double nota = 5;
		System.out.println(nota < 5); // False
		if(nota < 5) {
			System.out.println("Suspendido");
		}

		System.out.println(nota >= 5); // True
		if(nota >= 5) {
			System.out.println("Aprobado");
		}
	}

}
```

2.6_2

```
package entradaSalida;

public class Main {
	public static void main(String[] args) {
		double nota = 6;
		if(nota < 5) {
			System.out.println("Suspendido");
		}
		if(nota >= 5) {
			System.out.print("Aprobado"); // Se ejecutará siempre que sea
aprobado
			// EJEMPLO DE IFS ANIDADOS
			if(nota >= 5 && nota < 6)   No hace falta poner paréntesis cuando solo hay una instrucción
				System.out.println(" con un suficiente (" + nota + ")");
			if(nota >= 6 && nota < 7)
				System.out.println(" con un bien (" + nota + ")");
			if(nota >= 7 && nota < 9)
				System.out.println(" con un notable (" + nota + ")");
			if(nota >= 9 && nota <10)
				System.out.println(" con un excelente (" + nota + ")");
			if(nota == 10)
				System.out.println(" con un matricula de honor (" + nota +
")");
		}
	}
}
```

2.7_1

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		double nota = 3;

		if(nota < 5) {
			System.out.println("Suspendido");
		}

		if(nota >= 5) {
			System.out.println("Aprobado");
		}
	}

}
```

2.7_2

```
package entradaSalida;

public class Main {

	public static void main(String[] args) throws InterruptedException {
		int iteraciones = 5000;
		double nota = 8;

		// DOBLE IF EJEMPLO
		long start = System.currentTimeMillis();	// Milisegundos inciales
		// Bucle que se ejecuta tantas veces como definamos en la variable
iterraciones
		for(int i=0; i<iteraciones; i++) {
				if(nota < 5) { // Cuando la nota es menor que 5
					Thread.sleep(1);
				}

				if (nota >= 5){ // Cuando la nota es menor que 5
					Thread.sleep(1);
				}
		}

		// IF ELSE EJEMPLO
		long finish = System.currentTimeMillis() - start;
		System.out.println("Tiempo total de ejecución de un doble IF es: " +
finish + "ms");
```

```
		start = System.currentTimeMillis();
		// Bucle que se ejecuta tantas veces como definamos en la variable iterraciones
		for(int i=0; i<iteraciones; i++) {
			if(nota < 5) { // Cuando la nota es menor que 5
				Thread.sleep(1);
			}
			else { // El resto de veces
				Thread.sleep(1);
			}
		}
		finish = System.currentTimeMillis() - start;
		System.out.println("Tiempo total de ejecución de un IF ELSE es: " + finish + "ms");

		// IF ELSE IF ELSE EJEMPLO
		start = System.currentTimeMillis();
		// Bucle que se ejecuta tantas veces como definamos en la variable iterraciones
		for(int i=0; i<iteraciones; i++) {
			if(nota < 5) { // Cuando la nota es menor que 5
				Thread.sleep(1);
			} else if (nota <= 5 && nota >= 10) { // Cuando la nota es mayor o igual que 5 y menor o igual que 10
				Thread.sleep(1);
			}
			else { // El resto de veces
				Thread.sleep(1);
			}
		}
		finish = System.currentTimeMillis() - start;
		System.out.println("Tiempo total de ejecución de un IF IFELSE ELSE es: " + finish + "ms");
	}

}
```

2.8_1

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		boolean interruptor = false;

		if (interruptor) {
			//Se ejecuta cuando la condición es true
			System.out.println("La luz está encendida");
		} else {
			//Se ejecuta cuando la condición es false
			System.out.println("La luz está pagada");
		}
	}
}
```

2.9_1 Ley del 70% 30%

```
package entradaSalida;

public class Main {

	public static void main(String[] args) throws InterruptedException {
		//Si el 90 % de los alumnos suspenden...
		double nota = 1;
		int veces = 5000;

		long start = System.currentTimeMillis();	// Milisegundos inciales
		for(int i = 0; i < veces; i++) {
			if (nota<5) {
				//Se ejecuta cuando la condición es true  Condición que más se va a realizar (70%)
				Thread.sleep(1);
			} else {
				//Se ejecuta cuando la condición es false
				Thread.sleep(1);
			}
		}

		long finish1 = System.currentTimeMillis() - start;
		System.out.println("Cuando se ejecuta el primer boque del if-else tarda "
+ finish1 + " ms.");

		long start2 = System.currentTimeMillis();	// Milisegundos inciales
		for(int i = 0; i < veces; i++) {
```

```
                        if (nota>=5) {
                                //Se ejecuta cuando la condición es true
                                Thread.sleep(1);
                        } else {
                                //Se ejecuta cuando la condición es false
                                Thread.sleep(1);
                        }
                }

                long finish2 = System.currentTimeMillis() - start2;
                System.out.println("Cuando se ejecuta el segundo boque del if-else tarda
" + finish2 + " ms.");

                System.out.println("\nPor tanto, la diferencia entre ejecutar el primer
bloque y el segundo " + veces + " veces es " + (finish2 - finish1) + " ms.");
        }
}
```

2.10_1

```
package entradaSalida;

public class Main {
        public static void main(String[] args) {
                double number = 55;

                if (number > 0) {
                        System.out.println("¡Es positivo!");
                } else if (number < 0) {
                        System.out.println("¡Es negativo!");
                } else {
                        System.out.println("¡Es cero, na de ná!");
                }
        }
}
```

2.10_2 Podemos poner 'else if' tantas veces como queramos

```
package entradaSalida;

public class Main {
	public static void main(String[] args) {
	  int cp = 8241; // CP de Manresa

	  if (cp >= 8000 && cp <= 8080) {  // 08000 - 08080 = Barcelona
	System.out.println("¡Código postal de Barcelona!");
    } else if ((cp >= 8171 && cp <= 8174) || (cp >= 8195 && cp <= 8198)) { //08171 -
08174, 08195 - 08198 = Sant Cugat del Vallés
	System.out.println("¡Código postal de Sant Cugat del Vallés!");
    } else if (cp >= 8240 && cp <= 8248) { //08240 - 08248 = Manresa
	  System.out.println("¡Código postal de Manresa!");
    } else {
	  System.out.println("No conozco este código postal :(");
    }
	  }
}
```

2.11_1 Estructura condicional operador ternario/condicional ("bidireccional" en principio)

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		int edad = 18;
		// Ejemplo de operador ternario :? directamente desde un
System.out.println();
	  System.out.println(
		edad >= 18 // Condicion
	  True ? "Eres mayor de edad." // Instrucción que se ejecuta cuando es true
False : "Eres menor de edad." //Instrucción que se ejecuta cuando es false
		);

		// Ejemplo de operador ternario :? almacenado en un boolean y
posteriormente imprimido desde un System.out.println();
	  boolean esMayordeEdadBoolean = (edad >= 18) ? true: false;
	  System.out.println(esMayordeEdadBoolean);

	  // Ejemplo de operador ternario :? almacenado sobre un String y
posteriormente imprimido desde un System.out.println();
	  String esMayordeEdadString = (edad >= 18) ? "Eres mayor de edad.": "Eres
menor de edad.";
	  System.out.println(esMayordeEdadString);
	}
}
```

2.12_1 Implementación de 2 condiciones concatenadas

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		int edad = 15;
		System.out.println(edad>65?"¡No puedes trabajar!":edad>18?"¡A
Trabajar!":"¡Al cole!");
	}
}
```

2.13_1

```
package entradaSalida;

public class Main {
	public static void main(String[] args) {
	  int day = 1; // Dia de la semana
	  String dayType = "";

	  switch (day)
    {
	  case 1:   No es necesario poner llaves
	    System.out.println("Lunes");
	    dayType = "laborable";
	    break;   Para  no continuar la ejecución
	  case 2:
	    System.out.println("Martes");
	    dayType = "laborable";
	    break;
	  case 3:
	    System.out.println("Miércoles");
	    dayType = "laborable";
	    break;
	  case 4:
	    System.out.println("Jueves");
	    dayType = "laborable";
	    break;
	  case 5:
	    System.out.println("Viernes");
	    dayType = "laborable";
	    break;
	  case 6:
	    System.out.println("Sábado");
	    dayType = "festivo";
```

```
            break;
        case 7:
            System.out.println("Domingo");
            dayType = "festivo";
            break;

        default:
            System.out.println("Error! Día invalido");
            break;
    }

            // Mensaje que imprime el num del día y si es laborable
    System.out.println(day +" es un "+ dayType);
        }
}
```

2_13_2

```
package entradaSalida;

public class Main {
          public static void main(String[] args) {
            int day = 3; // Dia de la semana
            String dayType = "";

            switch (day)
    {                   No podemos utilizar variables booleanas
                        Debe ser una variable o expresión entera
        case 1:
            System.out.println("Lunes");
            dayType = "laborable";
        case 2:
            System.out.println("Martes");
            dayType = "laborable";
        case 3:
            System.out.println("Miércoles");
            dayType = "laborable";
        case 4:
            System.out.println("Jueves");
            dayType = "laborable";
        case 5:
            System.out.println("Viernes");
            dayType = "laborable";
        case 6:
            System.out.println("Sábado");
            dayType = "festivo";
        case 7:
            System.out.println("Domingo");
            dayType = "festivo";
```

```
        default: System.out.println("Error! Día invalido");
    }

    System.out.println(day +" es un "+ dayType);
        }
}
```

2.14_1

```
package entradaSalida;

public class Main {

        public static void main(String[] args) {
                int interruptor = 1;

                switch (interruptor) {
                        case 1 -> System.out.println("a");
                        case 2 -> System.out.println("b");
                }
        }
}
```

2.14_2

```
package entradaSalida;

public class Main {
        public static void main(String[] args) {
          char op = '*'; // Dia de la semana

          switch (op)
    {
        case '+':
          System.out.println("Suma");
          //break;
        case '-':
          System.out.println("Resta");
          //break;
        default:
          System.out.println("Error!");
          // break;
        case '/':
          System.out.println("Divisón");
          // break;
    }
        }
}
```

2.15_1 Ejemplo de SWITCH con cascada de CASES

```
package entradaSalida;

public class Main {
        public static void main(String[] args) {
          int day = 1; // Dia de la semana
          String dayType = "";

          switch (day)
    {
                    // Ejemplo de case con multiples opciones
        case 1:
        case 2:
        case 3:
        case 4:
        case 5:
          dayType = "laborable";
          break;
        case 6:
        case 7:

          dayType = "festivo";
          break;

        default:
          System.out.println("Error! Día invalido");
          break;
    }

          System.out.println(dayType);
        }
}
```

Todos estos cases son IGUALES

2.16_1

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		//Solo compatible con la versión 14 de JDK en adelante
		int day = 1;

		switch (day) {                    Con la nueva versión de Java
			case 1, 2, 3, 4, 5:
				System.out.println("Laborable"); break;
			case 6, 7:
				System.out.println("Festivo"); break;
			default:
				System.err.println("¡No es un día de la semana!");
		}

		String dayType = "";
		day = 6;
		switch (day) {                    Otra forma
			case 1, 2, 3, 4, 5 ->     dayType = "Laborable";
			case 6, 7 ->                      dayType = "Festivo";
			default ->                                System.err.println("¡No es un
día de la semana!");
		}
		System.out.println(dayType);
	}
}
```

2.17_1 SWITCH dentro de SWITCH

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		int tema = 1;
		int uf = 1;

		switch (tema) {
			case 1:
				System.out.print("1");
			switch (uf) {
				case 1:
					System.out.println("." + uf + " - " + "Variables en
Java");
			}
		}
	}
}
```

2.19_1 Más práctico utilizar un SWITCH (tarda muchos menos milisegundos en ejecutarse) que un ELSEIF

```
package entradaSalida;

public class Main {

	public static void main(String[] args) throws InterruptedException {
		int num = 5;
		int iteraciones = 5000;

		long start = System.currentTimeMillis();

		for(int i =0; i<iteraciones; i++) {
			if(num==1) {
				Thread.sleep(1);
			}else if(num==2) {
				Thread.sleep(1);
			}else if(num==3) {
				Thread.sleep(1);
			}else if(num==4) {
				Thread.sleep(1);
			}else if(num==5) {
				Thread.sleep(1);
			}
		}

		long finish = System.currentTimeMillis();
```

```
		System.out.println("Milisegundos con IF-ELSEIF: " + (start - finish));

		start = System.currentTimeMillis();
		for(int i =0; i<iteraciones; i++) {
			switch (num) {
				case 1:
					Thread.sleep(1);
					break;
				case 2:
					Thread.sleep(1);
					break;
				case 3:
					Thread.sleep(1);
					break;
				case 4:
					Thread.sleep(1);
					break;
				case 5:
					Thread.sleep(1);
					break;
			}
		}

		finish = System.currentTimeMillis();
		System.out.println("Milisegundos con SWITCH: " + (start - finish));
	}
}
```

3.2_1 Estructuras repetitivas/iterativas

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		int iteraciones = 100;
		for(int i = 0; i < iteraciones; i ++) {
			System.out.println(i +". I will not yell \"fire\" in a crowded
classroom");
		}
	}
}
```

3.3_1 Estructura de una condición de un FOR

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
	  for(int i=2; i<=10; i +=2 ) {
		System.out.println("Iteracción " + i);
	  }
	}
}
```

3.4_1

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {

		String[] miArray = {"Hola", "¿Qué tal?"};

	  for(int i=0; i < miArray.length; i++) {
		System.out.println("Palabra: " +  miArray[i] + "\n");
		for(int j = 0; j < miArray[i].length(); j++) {
			System.out.println("\t" + miArray[i].charAt(j));
		}
	  }
	}
}
```

3.5_1 Estructura de repetición ForEach (para cada), disponible a partir de la versión 5 de Java

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {

		String[] array = {"Hola", "¿Qué tal?"};

		for (String string : array) {
			System.out.println(string);
		}
	}
}
```

Hola
¿Qué tal?

3.5_2

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {

		String[] array = {"Hola", "¿Qué tal?"};

		for (String string : array) {
			System.out.println("\n" + string + "\n");
			char[] letras = string.toCharArray();
			for (char letra : letras) {
				System.out.println(letra);
			}
		}
	}
}
```

Igual que el ejemplo de arriba pero más fácil

3.6_1 Estructura de control WHILE

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {

		boolean interruptor = true;

		while (interruptor) {
			System.out.println("No hay luz");
		}
	}
}
```

3.6_2

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		boolean interruptor = true;
		int i = 1;

		while (interruptor) {
			System.out.println(i + ".I will not yell \"fire\" in a crowded
classroom");
			if(i==5) {
				System.err.println("Fin del programa");
				interruptor = false;
			}
			i++;
		}
	}
}
```

3.6_3

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {

		int i = 1;

		while (i<=5) {
			System.out.println(i + ".I will not yell \"fire\" in a crowded
classroom");
			i++;
		}
	}
}
```

3.7_1 Estructura de control DO WHILE

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		boolean interruptor = false;

		do {
			System.out.println("Cuando la condición es false se ejecuta 1 sola
vez");
		} while (interruptor);
	}
}
```

3.7_2

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		boolean interruptor = true;
		long time = System.currentTimeMillis();
		long time1s = time + 1000;

		int i = 0;
		System.out.println(time);
		do {
			if(time != time1s) {
```

```
                                System.out.println(i);
                                time = System.currentTimeMillis();
                        }
                        i++;
                } while (interruptor);
        }
}
```

4.2_1 Instrucciones de salto: break y continue

```
package entradaSalida;

public class Main {

        public static void main(String[] args) {
                // Bucle sin break
                System.out.println("Ejemplo de bucle FOR normal:");

                for (int i = 1; i < 6; i++) {
                        System.out.println("Instrucción " + i); //Instrucción 1, 2, 3, 4
                }

                System.out.println("----------------------------------------");
                System.out.println("Ejemplo de bucle FOR con CONTINUE:");

                // Bucle con continue en la 3 iteración
                for (int i = 1; i < 6; i++) {
                        /* Cuando inicia la ejecución del bucle (estructura repetitiva) for
                         * dentro del bloque de instrucciones que vamos a repetir
                         * analizamos con (una estructura condicional) if
                         * si la variable i tiene un valor de 3
                         * y si se cumple dicho caso, entre en el bloque de if
                         * ejecutando (la instruccion de salto) CONTINUE
                         * del bucle ya que  con la palabra reservada CONTINUE
                         * evitamos que se ejecuten el resto de las líneas de dicha
iteracción
                         * pero no las del resto de iterraciones del bucle.
                         */
                         if (i == 3) {
                           continue; // Fin de dicha iteracción PERO NO DEL BUCLE
                           // EL FIN DE LA ITERRACIÓN LO DITAMINA LA CONDICIÓN DE
DICHO BUCLE
                         }
                         System.out.println("Instrucción " + i);
                }
```

```
                System.out.println("----------------------------------------");
                System.out.println("Ejemplo de bucle FOR con BREAK:");

                // Bucle con break en la 3 iteración
                for (int i = 1; i < 6; i++) {
                        /* Cuando inicia la ejecución del bucle (estructura repetitiva) for
                         * dentro del bloque de instrucciones que vamos a repetir
                         * analizamos con (una estructura condicional) if
                         * si la variable i tiene un valor de 3
                         * y si se cumple dicho caso, entre en el bloque de if
                         * ejecutando (la instruccion de salto) BREAK
                         * del bucle ya que  con la palabra reservada BREAK
                         * evitamos que se ejecuten el resto de las líneas de dicha
iteracción
                         * y también las del resto de iterraciones del bucle.
                         */
                         if (i == 3) {
                           break; // Fin de dicha iteracción Y TAMBIEN DEL BUCLE
                         }
                         System.out.println("Instrucción " + i);
                }

        }

}
```

5.3_1 Programación funcional: Creando métodos void (sin return) en la misma clase

Main.java

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		// Forma 1
		Main main = new Main(); //Instanciamos es decir
declaramos/construimos la clase a la que pertenece dicho método
		main.saludarA("David"); //Llamamos al método correspondiente

		// Forma 2: Cuando el método tiene static
		saludoGenerico();
	}

	// Forma 1: Cuando NO asignamos static al método
	public void saludarA(String nombre){ //
		System.out.println("Hola " + nombre);
	}

	// Forma 2: Cuando asignamos static al método
	public static void saludoGenerico(){
		System.out.println("Hola!");
	}

}
```

Hola David
Hola!

No se necesita instanciación del objeto para poder utilizarlo

5.4_1 Programación funcional: Creando métodos (sin void) en la misma clase

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		// Forma 1
		Main main = new Main(); //Instanciamos es decir
declaramos/construimos la clase a la que pertenece dicho método
		main.saludarA("David"); //Llamamos al método correspondiente

		// Forma 2: Cuando el método tiene static
		saludoGenerico();
	}

	// Forma 1: Cuando NO asignamos static al método
	public String saludarA(String nombre){ //
		return "Hola " + nombre;
	}

	// Forma 2: Cuando asignamos static al método
	public static String saludoGenerico(){
		return "Hola!";
	}

}
```

NO se muestra por pantalla porque no lo hemos imprimido

5.4_2

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		// Forma 1
		Main main = new Main(); //Instanciamos es decir
declaramos/construimos la clase a la que pertenece dicho método
		System.out.println(main.saludarA("David")); //Llamamos al método
correspondiente

		// Forma 2: Cuando el método tiene static
		System.out.println(saludoGenerico());
	}

	// Forma 1: Cuando NO asignamos static al método
	public String saludarA(String nombre){
		 return "Hola " + nombre;
	}
```

```
	// Forma 2: Cuando asignamos static al método
	public static String saludoGenerico(){
		return "Hola!";
	}

}
```

5.5_1
Saludo.java

```
package entradaSalida;

public class Saludo {
		// Forma 1: Cuando NO asignamos static al método
		public void saludarA(String nombre){ //
			System.out.println("Hola " + nombre);
		}

		Sin instanciar objeto
		// Forma 2: Cuando asignamos static al método
		public static void saludoGenerico(){
			System.out.println("Hola!");
		}
}
```

5.5_2
Main.java

```
package entradaSalida;

public class Main {

	public static void main(String[] args) {
		// Forma 1: Cuando el método no tiene static
		Saludo saludar = new Saludo(); //Instanciamos (llamar) a la clase
		saludar.saludarA("David"); // Ejecutamos el método

		// Forma 2: Cuando el método tiene static
		Saludo.saludoGenerico(); // Clase.metodo();
	}
}
```

5.5_3

```
package entradaSalida;

public class Main {
	public static void main(String[] args) {
	  System.out.println(saludar("David"));
	  System.out.println(saludar("Juan"));
	}

	public static String saludar(String txt) {
		return txt.equals("David")?txt:null; // Si el parametro introducido no es
David devolverá null
	}
}
```

David null

6.5_1 Entrada de datos con Scanner

```
package entradaSalida;

import java.util.Scanner; // importamos la clase Scanner del package java.util

public class Main {

	public static void main(String[] args) {
		/* Instanciamos (creamos) la clase Scanner
		 * y le inciamos que vamos a entrar datos
		 * desde la terminar con System.in
		 */
		Scanner sc = new Scanner(System.in);

			System.out.print("Introduce un nombre: "); // Escribimos un
mensaje para que el usuario sepa que tiene que hacer
			String name = sc.nextLine(); // Guardamos el String (con el
método nextLine()) en la variable name
			System.out.println("Hola " + name + "!"); //Mostramos el nombre
recibido por la consola

		sc.close() // Finalizamos el uso de Scanner
	}
}
```

Ejemplo de cómo capturar un String con Scanner

Objeto de Scanner

Código más rápido y eficiente al cerrar el método

6.6_1 Explicando cómo y por qué utilizar el método close al finalizar el uso de Scanner

```
package entradaSalida;

import java.util.Scanner; // importamos la clase Scanner del package java.util

public class Main {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);

				System.out.print("Introduce un nombre: ");
				String name = sc.nextLine();
				System.out.println("Hola " + name + "!");

		sc.close(); // Cerramos el uso de Scanner por lo que a partir de aquí no
podriamos utilizar el objeto Scannner

		System.out.print("Introduce un nombre: ");
		name = sc.nextLine();
		System.out.println("Hola " + name + "!");
	}
}
```

6.6_2

```
package entradaSalida;

import java.util.Scanner; // importamos la clase Scanner del package java.util

public class Main {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);

				System.out.print("Introduce un nombre: ");
				String name = sc.nextLine();

				System.out.print("Introduce tu apellido: ");
				String apellido = sc.nextLine();
				System.out.println("¡Hola " + name + " " + apellido + "!");

		sc.close(); // Cerramos el uso de Scanner por lo que a partir de aquí no
podriamos utilizar el objeto Scannner
	}
}
```

6.8_1

```
package entradaSalida;

import java.util.Scanner;

public class Main {

	public static void main(String[] args) {
		System.out.print("Introduce un número entero: ");
		Scanner sc = new Scanner(System.in); // Creando Scanner
		int num = sc.nextInt();
		System.out.println("Genial, me gusta el " + num);
		sc.close(); // Cerrando Scanner
	}
}
```

6.8_2

```
package entradaSalida;

import java.util.Scanner;

public class Main {

	public static void main(String[] args) {
		try {
			scanner();
		} catch (Exception e) {
			System.err.println("Lo has introducido mal, intentalo de nuevo");
			scanner();
		}

	}

	public static void scanner() {
		System.out.print("Introduce un número entero: ");
		Scanner sc = new Scanner(System.in); // Creando Scanner
		int num = sc.nextInt();
		System.out.println("Genial, me gusta el " + num);
		sc.close(); // Cerrando Scanner
	}
}
```