Primero creamos un proyecto nuevo, al cual hemos llamado “PSP01_Ej1_p1” y desarrollamos el programa dedicado a pedir un conjunto indefinido de números por teclado y ordenarlos, para así completar la primera parte del primer ejercicio.

Como se puede apreciar, se ha añadido el método de control de excepciones ‘try catch’, para así capturar los posibles errores, que puedan producirse durante la introducción y lectura de los datos por teclado, para así evitar que finalice inesperadamente la ejecución de nuestra aplicación. Y también hemos hecho uso de las clases nativas InputStreamReader, BufferedReader y String, mediante la utilización de alguno de sus métodos preestablecidos, para así recoger y procesar adecuadamente los datos de entrada, y poder ejecutar adecuadamente el conjunto de operaciones solicitadas.

```
34      public static void main(String[] args) {
35          try {
36              InputStreamReader isr = new InputStreamReader(System.in);
37              BufferedReader bf = new BufferedReader(isr);
38
40              String entradaTeclado = null;
41
42              System.out.print("Introduce los números a ordenar a través del teclado y separados por un espacio: \n");
43              while((entradaTeclado = bf.readLine()) != null) {
44                  String datos[] = entradaTeclado.split(" ");
45                  int numeros[] = new int[datos.length];
46
47                  for (int i = 0; i < numeros.length; i++) {
48                      numeros[i] = Integer.parseInt(datos[i]);
49                  }
50
51                  Arrays.sort(numeros);
52
53                  System.out.println("A continuación, se muestran ordenados los números introducidos: ");
54                  for(int i = 0; i < numeros.length; i++) {
55                      System.out.print(numeros[i] + " ");
56                  }
57                  System.out.println(" ");
58              }
59          } catch (IOException ex) {
60              Logger.getLogger(OrdenarNumeros.class.getName()).log(Level.SEVERE, null, ex);
61          }
62
63      }
```

Seguidamente, pasamos a desarrollar la clase Aleatorios, tal y como se solicita en el enunciado y ubicándola por ende en otro proyecto independiente.

```
11  public class Aleatorios {
12
13      public static void main(String[] args) {
14
15          int cantidadNumerosGenerados = 40;
16
17          for (int i = 0; i < cantidadNumerosGenerados; i++) {
18              System.out.print(generarNumeroAleatorio(0, 100) + " ");
19          }
20
21      }
22
23      /**
24      * Método para generar números aleatorios contenidos entre un valor mínimo y
25      * un máximo introducidos.
26      * @param min
27      * @param max
28      * @return int
29      */
30      public static int generarNumeroAleatorio(int min, int max) {
31          int num = (int) (Math.random()*(max - min + 1) + (min));
32          return num;
33      }
34  }
```

En la captura de la página anterior se puede apreciar la **clase Aleatorios**, con un método **main**, otro **método dedicado a la generación de números aleatorios** (de uno en uno) y también un pequeño **comentario**, usando la herramienta Java Doc.

Los **comentarios de Java Doc** se definen mediante el uso de una 'barra diagonal hacia la derecha (/)' y dos asteriscos (*), a modo de apertura, y un asterisco (*) junto a otra 'barra (/)', a modo de cierre del comentario con Java Doc. En él se explica brevemente cual resulta la función del método referenciado y esté lo está debido a encontrarse justo debajo de la etiqueta (del comentario), aunque también se pueden añadir otras variables como los parámetros (de entrada) o su retorno, entre otras.

El **método 'main'** resulta el primer método que es ejecutado y sirve para inicializar y definir qué otros métodos y/o variables queremos que se ejecuten.como iniciador de la variable auxiliar requerida a la que hemos.

En el **'método auxiliar'** se genera un número aleatorio (o 'random') mediante el uso del método 'random()' perteneciente a la clase 'Math'. En dicho 'método auxiliar' se definen e introducen dos parámetros (min y max) y se devuelve un número entero.

Nuestro siguiente paso consiste en actualizar el archivo pom.xml, relacionado con Maven, para así añadir la dependencia del 'plugin' dedicado a generar el archivo ejecutable .jar, de cada proyecto.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.kevinzamora</groupId>
    <artifactId>PSP01_ej1_p1</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>21</maven.compiler.source>
        <maven.compiler.target>21</maven.compiler.target>
        <exec.mainClass>Ejercicio1.OrdenarNumeros</exec.mainClass>
    </properties>
    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.34</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <!-- Build an executable JAR -->
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <version>3.1.0</version>
                <configuration>
                    <archive>
                        <manifest>
                            <mainClass>Ejercicio1.OrdenarNumeros</mainClass>
                        </manifest>
                    </archive>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

En él podemos definir la versión de java con la que queremos exportar nuestro ejecutable y también cuál resulta nuestra clase principal, entre otras configuraciones.

PD: Cabe destacar que nos estaba dando un error al intentar ejecutar el '.jar exportado', al intentar utilizarlo con la versión 11 de java (en local) y haber exportado dicho ejecutable con la versión 21.

Para solucionarlo, nos hemos limitado a actualizar la 'versión local' por esa otra versión LTS o 'Long Time Support', la versión 21.

Por último y para finalizar con el ejercicio 1, procedemos con la ejecución de ambos archivos ejecutables .jar, mediante el siguiente comando y haciendo uso de una 'tubería', para lograr conectar ambos 'ejecutables' y transferir el resultado del ejecutable 'aleatorios.jar' al otro ejecutable 'ordenarNumeros.jar'. El citado comando es el siguiente:

java -jar aleatorios.jar | java -jar ordenarNumeros.jar

Este fue el resultado obtenido: