Ampliación de clases.

Índice:

1. Derechos de acceso

Acceso privado (private). Sólo se puede acceder al atributo desde métodos de la clase, o sólo puede invocarse el método desde otro método de la clase. Las clases no pueden ser definidas como privadas.

Acceso público (public). Los elementos públicos se pueden utilizar libremente. Desde cualquier clase se puede acceder a un atributo, método o clase “public”.

Acceso protegido (protected). Sólo pueden ser utilizados en la clase que los define, en aquellas que la extiendan y cualquier clase definida en el mismo paquete.

Acceso por defecto (sin modificador). Si no se indica especificador, el acceso es libre dentro de clase que lo define y en cualquier clase del mismo paquete.

2. Clases predefinidas

Java nos proporciona gran cantidad de clases predefinidas especializadas en comunicaciones, web, interfaz de usuario, matemáticas, etc. A continuación, se describen algunas.

2.1 La clase String

Como sabemos, Java nos permite hacer uso de cadenas de caracteres como objetos mediante el uso de la clase String.

En este punto, comentaremos algunos de los métodos más interesantes de esta clase.

- int length(). Devuelve el número de caracteres de la cadena.
- concat. Unir o concatenar cadenas. Este método devuelve una cadena como unión de string1 y string2. **String resultado= string1.concat(string2);**
  También pueden unirse cadenas mediante el operador + (más).
- char charAt(int index). Devuelve el carácter situado en la posición indicada por index.
- int compareTo(String otra). Devuelve 0, si las cadenas son iguales, 1 si la cadena asociada al método es mayor y -1 si otra es mayor.
- int compareToIgnoreCase(String otra). Igual pero ignorando mayúsculas y minúsculas.
- boolean endsWith(String sufijo). Devuelve true si la cadena termina con los caracteres que se indican en sufijo.
- boolean equals(String cadena). Devuelve true o false según las cadenas sean o no iguales.

- int indexOf(String str). Devuelve el índice de la cadena donde se ha encontrado la secuencia pasada, -1 en caso de no encontrarse.
- String replace(char oldChar, char newChar). Reemplaza todos los caracteres que coincidan con oldChar por newChar.
- String [] split(String expresión). Devuelve un array de String a partir de la cadena original, tomando como separador la cadena situada en expresión.

  Ejemplo:

```
String meses = "enero,febrero,marzo,abril,mayo";
String [] arrayMeses = meses.split(",");

System.out.println(Arrays.toString(arrayMeses));
```

```
[enero, febrero, marzo, abril, mayo]
```

- String toLowerCase(). Convierte a minúsculas.
- String toUpperCase(). Convierte a mayúsculas.
- char [] toCharArray(String s). Convierte la cadena en un array de char.

1.1. La clase Character

Permite trabajar con caracteres.

- Character(char value). Constructor
- static boolean isLetter(). Determina si un carácter es una letra o no.
- static boolean isDigit(). Determina si un carácter es un dígito.
- static boolean isWhiteSpace(). Determina si el carácter es un blanco.
- char toLowerCase(). Convierte a minúscula.
- char toUpperCase(). Convierte a mayúscula.
- toString(). Convierte el carácter en cadena.

1.2. La clase Integer

Esta clase es un envoltorio (wrap) para enteros primitivos.

- Integer(int value). Constructor.
- Integer(String s). Constructor
- int compareTo(Integer otroEntero). Devuelve 0, 1 o -1 según los enteros sean iguales, mayor el primero o el segundo.

- double doubleValue(). Convierte el entero a double.
- int intValue(). Devuelve el Integer como int.
- long longValue(). Devuelve el Integer como long.
- static int parseInt(String s). Convierte la cadena en entero.
- static String toBinaryString(int i). Convierte el entero a binario en un String.

1.3. La clase Math

Esta clase proporciona métodos para operar con números, tales como potencias, logaritmos, funciones trigonométricas, …

- static double abs(double a). Devuelve el valor absoluto del doble pasado. También existe para float, long e int.
- static double max(double a, double b). Devuelve el mayor valor de los pasados como parámetros. Está sobrecargada para long, int y float.
- static double min(double a, double b). Devuelve el menor valor de los pasados. También está sobrecargada para long, int y float.
- static double pow(double base, double exponente). Obtiene la potencia, base elevada a exponente.
- static double random(). Devuelve un valor doble mayor o igual que 0 y menor 1.
- static long round(double a). Redondea el doble a long.

1.4. La clase Arrays

La clase Arrays contiene una serie de métodos que permiten trabajar de forma rápida y ágil con arrays. Estos métodos lanzan la excepción NullPointerException si el array es nulo.

- static <T> List<T> asList(T... a). Devuelve un List a partir de un array.
- static int binarySearch(int[] a, int key). Devuelve la posición del array en la que se encuentra el valor “key”. Si no lo encuentra devuelve un valor negativo (-puntoDeInserción). El array debe estar ordenado. Está sobrecargada para los tipos primitivos long, char, float, …
- static int binarySearch(int[] a, int fromIndex, int toIndex, int key). Variante de la anterior.
- static int[] copyOf(int[] original, int newLength). Obtiene un array a partir de otro. El tamaño lo determina newLength. Está sobrecargada.

- static int[] copyOfRange(int[] original, int from, int to). Copia parte de un array en otro. Está sobrecargada.
- static boolean equals(long[] a, long[] a2). Devuelve true si los arrays son iguales.
- static void fill(char[] a, char val). Rellena cada elemento del array a con el valor indicado en val. Está sobrecargada.
- static void sort(float[] a). Ordena de forma ascendente el array a.
- static String toString(double[] a). Devuelve un String a partir de los valores del array. Está sobrecargada.

1.5. La clase Random

Esta clase se utilizada para generar números pseudoaleatorios.

- Random(). Constructor.
- Random(long seed). Constructor usando semilla.
- int nextInt(). Genera un entero al azar.
- int nextInt(int value). Genera un entero entre 0 y value-1.

2. Fechas en Java 8.

En el paquete java.time Java nos proporciona una serie de clases para el manejo de fechas y horas.

Algunos ejemplos:

- LocalDate. Para representar fechas (sin hora).

```
//creación de fecha con los valores del equipo
LocalDate fecha1 = LocalDate.now();
//creación de fecha con valores concretos
LocalDate fecha2 = LocalDate.of(2019, 11, 12);
//toString
System.out.println("Fecha1: " + fecha1.toString());
System.out.println("Fecha2: " + fecha2.toString());
//obtener día, mes y año
System.out.println("Día: " + fecha1.getDayOfMonth());
System.out.println("Mes: " + fecha1.getMonthValue());
System.out.println("Año: " + fecha2.getYear());

System.out.println("Fecha1 es posterior a fecha2: "
        + fecha1.isAfter(fecha2));
```

- LocalTime. Para manejar horas (sin fecha).

```
LocalTime hora1 = LocalTime.now();
LocalTime hora2 = LocalTime.of(20, 22);
System.out.println("Hora 1: " + hora1.toString());
System.out.println("Hora 2: " + hora2.toString());
System.out.println("Hora1 es anterior a hora2?" +
        hora1.isBefore(hora2));
//add 25 min.
System.out.println(hora2.plusMinutes(25));
```

- LocalDateTime. Para manejo de fechas y horas.

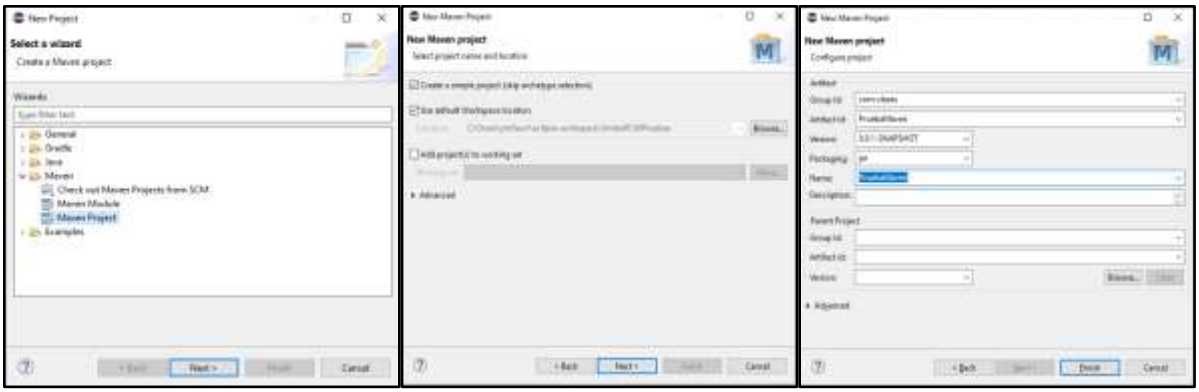
```
LocalDateTime tiempo1 = LocalDateTime.now();
LocalDateTime tiempo2 = LocalDateTime.of(2017,
        Month.APRIL, 20, 8, 33);

System.out.println("Tiempo1: " + tiempo1.toString());
System.out.println("Tiempo2: " + tiempo2.toString());

//restar 10 días
tiempo2 = tiempo2.minusDays(10);
System.out.println("Tiempo2: " + tiempo2.toString());
```

3. Creación de un proyecto Maven.

En nuestros proyectos Java, podemos usar APIs de terceros. Una API no es más que un conjunto de clases diseñadas por terceros que aportan funcionalidad a nuestros proyectos.

Para agregar estas clases (archivos jar) podemos añadirlos directamente al proyecto (Build Path-Add External Archives) o mejor aún, podemos crear un proyecto Maven e incluir las dependencias de estas Apis. Con Maven podemos gestionar las dependencias (las librerías de clases de terceros) de una forma rápida y ágil.

Para crear un proyecto Maven: New- Project- Maven Project.



Abriremos el fichero pom.xml y en él incluimos las dependencias que necesitemos. Estas dependencias hacia APIS se obtienen desde la web de la API o bien desde la web del repositorio de Maven, https://mvnrepository.com/.

4. Uso de la API JodaTime

Si queremos trabajar con fechas/horas de forma distinta a como lo hacemos con las clases vistas en el paquete java.time, disponemos de la API JodaTime.

Entre otras clases, podemos usar LocalDate para fechas, LocalTime para horas y DateTime para fechas y horas.,

Pincha aquí si quieres saber más sobre esta API.

```
//instanciar objeto fecha
LocalDate fecha1 = new LocalDate();
LocalDate fecha2 = new LocalDate(2000,01,16);

System.out.println(fecha1.toString());
System.out.println(fecha2.toString());

//día de la semana
System.out.println(fecha1.getDayOfWeek());
//son iguales
System.out.println("Son iguales?" + fecha1.isEqual(fecha2));
```

5. Generar el .jar de un proyecto.

Para generar el .jar de un proyecto, o lo que es lo mismo, el ejecutable para la máquina virtual de Java (JVM), debemos seguir los siguientes pasos:

- Seleccionar el proyecto y pulsar el botón derecho.
- Seleccionar Exportar.
- Elegir la carpeta Java y dentro la opción Runneable Jar File
- Pulsar Next
- Elegir el proyecto de nuevo (Launch Configuration).
- Elegir la carpeta de destino donde dejar el .jar
- Pulsar Finish.

Podemos ejecutar un .jar desde la línea de comandos mediante:

java -jar NombreProyecto.

Puede que obtengamos algún error al ejecutar la aplicación debido a incompatibilidades de versiones. Es decir, la JVM puede no estar preparada para la versión del JRE que hemos usado en el proyecto.

6. JavaDoc

La documentación de un proyecto es imprescindible para su posterior utilización y/o mantenimiento. Documentar un proyecto consiste en explicar de forma breve y concisa, para que sirve cada clase, que hace cada método, sus parámetros, valor de retorno, etc.

Javadoc es una utilidad que permite generar la documentación de un proyecto en formato HTML. Para que esta herramienta pueda generar la documentación del proyecto es necesario seguir una serie de reglas:

- La documentación debe escribirse como comentarios empezando con /** y terminando con */.
- El comentario debe situarse antes del elemento a comentar: clase, método, atributo…
- Javadoc se ayuda de las siguientes etiquetas:
    - @author. Para indicar el nombre del desarrollador.
    - @deprecate. Elemento obsoleto.
    - @param. Parámetro de un método.
    - @return. Informa del valor de retorno del método.

- @see. Asocia con otro método o clase. Si queremos saltar a ese otro método o clase debemos escribir:
    - paquete.clase#método() nombre
- @version. Versión del método o clase que se está describiendo.

En Eclipse podemos generar el Javadoc desde el proyecto de la siguiente forma:

Project-Generate Javadoc

- seleccionar el proyecto
- elegir la carpeta destino de la documentación
- pulsar Next
- pulsar Finish