

4.- Las clases del paquete java.time

- A partir de la introducción de la versión Java 8, el manejo de las fechas y el tiempo ha cambiado en Java. Desde esta versión, se ha creado una nueva API para el manejo de fechas y tiempo en el paquete java.time, que resuelve distintos problemas que se presentaban con el manejo de fechas y tiempo en versiones anteriores (la clase Date, la clase Calendar y la clase GregorianCalendar).
- Por lo tanto, el paquete java.time es el principal API para el manejo de fechas, horas, instantes y duraciones.

45

PROGRAMACIÓN

4.- Las clases del paquete java.time

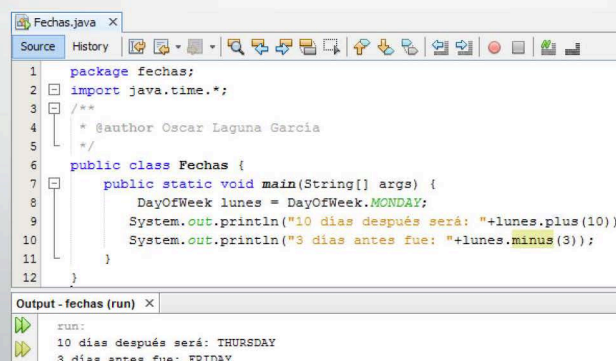
- Por lo tanto, si trabajamos con Java 8 o superior, se recomienda el uso de las clases incluidas dentro del paquete java.time para el manejo de tiempos y fechas (clases como Clock, Duration, Instant, LocalDate, LocalTime, MonthDay, OffsetDateTime, OffsetTime, Period, Year, YearMonth, ZonedDateTime, ZoneId y ZoneOffset).
- Estas clases contienen multitud de métodos para trabajar con fechas. A diferencia de la clase Math, estas clases no tienen métodos estáticos, por lo que para utilizar sus métodos deberemos instanciarlas a un objeto.

46

PROGRAMACIÓN

4.- Las clases del paquete java.time

- La clase **DayOfWeek** nos permite trabajar con los días de la semana:



```

1 package fechas;
2 import java.time.*;
3 /**
4  * @author Oscar Laguna Garcia
5  */
6 public class Fechas {
7     public static void main(String[] args) {
8         DayOfWeek lunes = DayOfWeek.MONDAY;
9         System.out.println("10 días después será: "+lunes.plus(10));
10        System.out.println("3 días antes fue: "+lunes.minus(3));
11    }
12 }

```

Output - fechas (run) X

```

run:
10 días después será: THURSDAY
3 días antes fue: FRIDAY

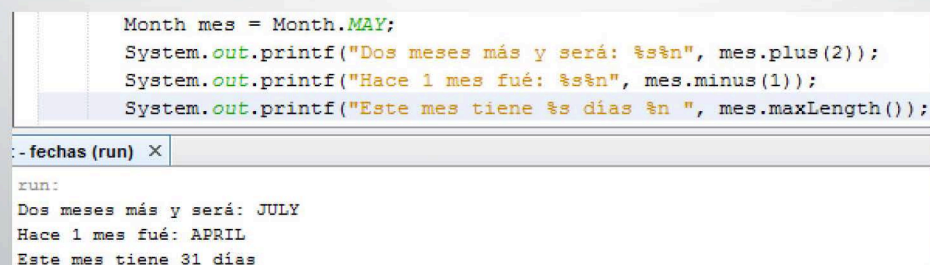
```

47

PROGRAMACIÓN

4.- Las clases del paquete java.time

- La clase **Month** nos permite trabajar con los meses del año:



```

Month mes = Month.MAY;
System.out.printf("Dos meses más y será: %s\n", mes.plus(2));
System.out.printf("Hace 1 mes fué: %s\n", mes.minus(1));
System.out.printf("Este mes tiene %s días\n", mes.maxLength());

```

Output - fechas (run) X

```

run:
Dos meses más y será: JULY
Hace 1 mes fué: APRIL
Este mes tiene 31 días

```

48

PROGRAMACIÓN

4.- Las clases del paquete java.time

- Con la clase **LocalDate** nos permite establecer una fecha determinada con la que trabajar:

```

    LocalDate date = LocalDate.of(1986, Month.FEBRUARY, 22);
    DayOfWeek dia=date.getDayOfWeek();
    System.out.println("El día que nací fue el "+date+" y era un "+dia);

```

- fechas (run) X

run:
El día que nací fue el 1986-02-22 y era un SATURDAY

```

    LocalDate date = LocalDate.now();
    DayOfWeek dia=date.getDayOfWeek();
    System.out.println("Hoy es "+date+" y es "+dia);

```

- fechas (run) X

run:
Hoy es 2015-05-25 y es MONDAY

49

PROGRAMACIÓN

4.- Las clases del paquete java.time

- Para representar el mes de un año específico, usamos la clase **YearMonth**. Podemos obtener la cantidad de días de ese mes, útil cuando manejamos los años bisiestos:

```

    YearMonth mes = YearMonth.now();
    System.out.println("Mes actual "+mes+" y tiene "+mes.lengthOfMonth()+" días");
    mes = YearMonth.of(2004, Month.FEBRUARY);
    System.out.printf("El mes %s tuvo %d días. \n", mes, mes.lengthOfMonth());

```

- fechas (run) X

run:
Mes actual 2015-05 y tiene 31 días
El mes 2004-02 tuvo 29 días.

50

PROGRAMACIÓN

4.- Las clases del paquete java.time

- La clase **LocalTime** es muy útil para representar horas y tiempos de un día, tales como la hora de inicio de una película o el horario de atención de una biblioteca:

```
LocalTime ahora = LocalTime.now();
System.out.println("En este momento son las "+ahora.getHour()+" horas ");
System.out.println("con "+ahora.getMinute()+" minutos y "+ahora.getSecond()+" segundos");
```

<

- fechas (run) X

run:

En este momento son las 18 horas
con 36 minutos y 4 segundos

51

PROGRAMACIÓN

4.- Las clases del paquete java.time

- La clase **LocalDateTime** es usada para representar la fecha (año, mes, día) junto con la hora (hora, minuto, segundo, nanosegundo), siendo la combinación de LocalDate y LocalTime.

```
LocalDateTime ahora = LocalDateTime.now();
System.out.println("La fecha y hora completa es: "+ahora);
System.out.println("Hace seis meses fue: "+LocalDateTime.now().minusMonths(6));
```

<

- fechas (run) X

run:

La fecha y hora completa es: 2015-06-25T00:43:04.936
Hace seis meses fue: 2014-11-25T00:43:04.983

52

PROGRAMACIÓN

- 53

EJERCICIOS

- 54

EJERCICIOS

- **Pistas:**

- *Para calcular el periodo que hay entre dos fechas necesitaras utilizar métodos de la clase **Period**.*
- *Para calcular cuantos días llevas vividos necesitarás utilizar métodos de la clase **ChronoUnit**.*

55

PROGRAMACIÓN

EJERCICIOS

- **Ejercicio 19.- (OPTATIVO)** Crea un programa que te pida tu edad y la fecha de expedición de tu DNI y te calcule en que fecha tienes que renovarlo, utilizando el paquete java.time.
- **PISTA:** Plazos de validez del DNI
- *Con carácter general, el DNI tendrá un período de validez, a contar desde la fecha de la expedición o de cada una de sus renovaciones, de:*
 - *2 años, cuando el niño sea menor de 5 años.*
 - *5 años, cuando el titular no haya cumplido los 30 en el momento de la expedición o renovación.*
 - *10 años, cuando el titular haya cumplido los 30 y no haya alcanzado los 70.*
 - *Permanente, cuando el titular haya cumplido los 70 años.*

56

PROGRAMACIÓN

EJERCICIOS

- **Ejercicio 20.- (OBLIGATORIO)** Teniendo en cuenta que el instituto abre a las 08:00 de la mañana, y cierra a las 21:20, realiza un programa que, utilizando el paquete *java.time* te calcule:
 1. Cuantas horas y minutos lleva abierto hasta la hora actual, o si ya está cerrado.
 2. Cuantas horas y minutos quedan para que cierre, o si ya está cerrado.

57

PROGRAMACIÓN

EJERCICIOS

- **Ejercicio 21.- (OPTATIVO)** Crea un programa que te pida una fecha de nacimiento y te calcule cual es su signo del zodiaco.
- Pista: Utiliza un switch y el paquete *java.time*.

58

PROGRAMACIÓN