Sesión 4:
Varias clases predefinidas

java.util y java.lang

Los paquetes *java.util* y *java.lang*, son dos de los paquetes más utilizados en cualquier tipo de desarrollo informático basado en Java.

Son conjuntos bastantes grandes de interfaces y clases, que son de gran **utilidad**.

Vamos a descubrir algunas de las clases más usadas y más útiles de estos paquetes.

Se puede consultar la documentación oficial para consultar el resto de utilidades.

- **java.util:**

https://docs.oracle.com/javase/8/docs/api/java/util/package-summary.html

- **java.lang:**

https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/package-summary.html

Clases Envoltorio (wrappers)

En Java hemos de distinguir entre:

- Variables de tipos primitivos/simples: Almacenan el valor.
- Variables objeto: Almacenan la referencia al objeto.

Para usar un tipo básico como objeto, es necesario convertir a la variable en un objeto. De esta forma tendremos disponibles los métodos predefinidos para los objetos.

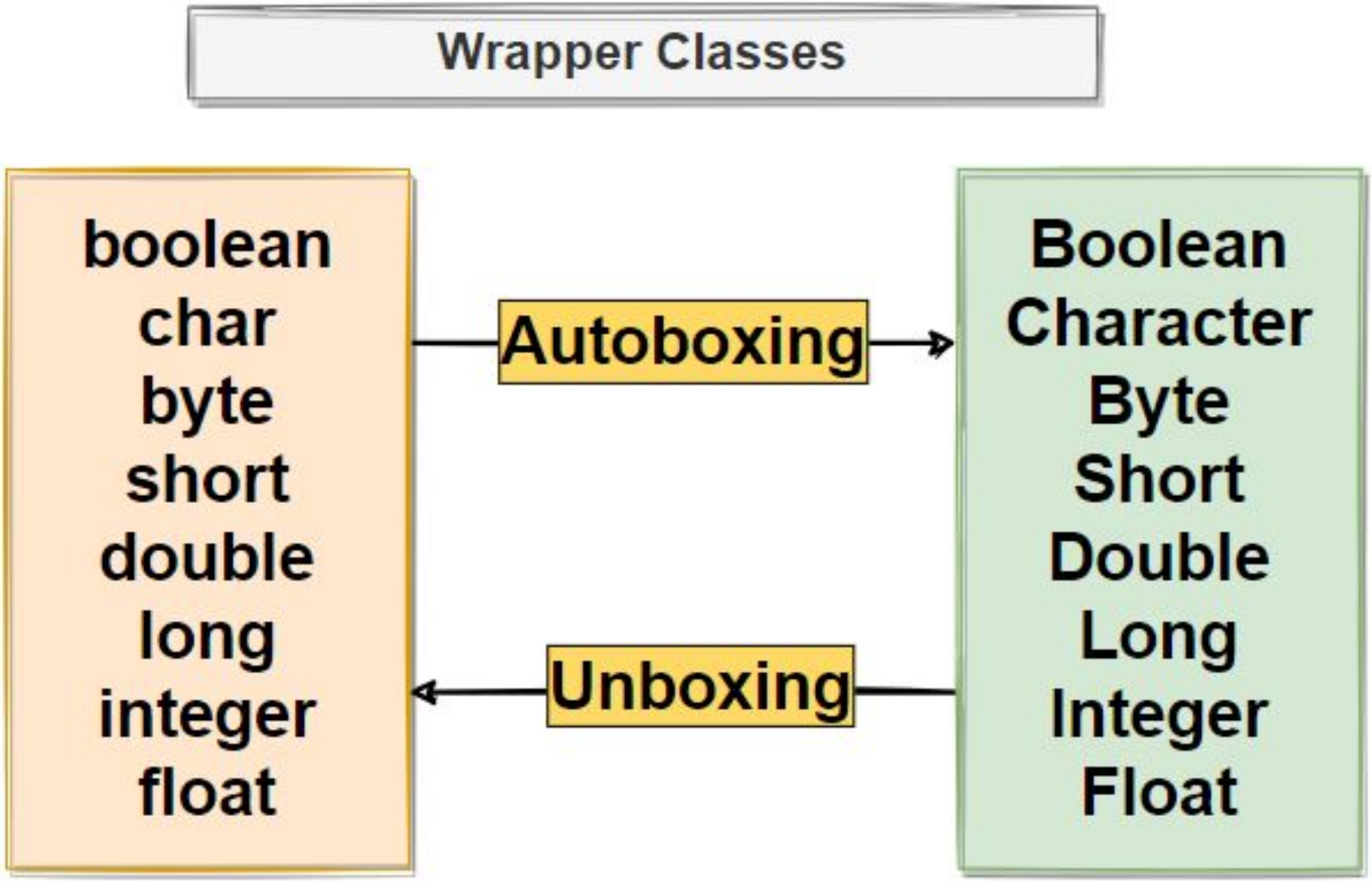
Se encuentran en el paquete *java.lang.*

Para declararlos, se usa el nombre de la clase en vez del nombre de el tipo básico o primitivo.

Ejemplo: Byte b = 2;

Tipo básico	**Envoltorio**
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

Clases Envoltorio (wrappers)



Clases Envoltorio (wrappers)

Al crear el envoltorio, podemos usar métodos que nos dan información sobre el objeto:

- Estos devuelven el valor asociado al objeto:

  intValue(), byteValue(), shortValue(), longValue(), floatValue(), doubleValue(), charValue(), booleanValue()

- Otro método útil es toString(), que lo convierte a un String.

```
public class Main {
  public static void main(String[] args) {
    Integer myInt = 100;
    String myString = myInt.toString();
    System.out.println(myString.length());
  }
}
```

Trabajo con fechas

Date

Date representa un instante en el tiempo con una precisión de milisegundos.

- Constructores: `Date date = new Date (); //Date con fecha actual`

  `Date date = new Date (long milisegundos); //Date pasando fecha`

La fecha trabaja con long porque almacena el valor de milisegundos transcurridos desde el 1 de enero de 1970 a las 00:00.

https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/Date.html

boolean after (Date when)	Devuelve true si la fecha que llama es posterior a la fecha *when*
boolean before (Date when)	Devuelve true si la fecha que llama es anterior a la fecha *when*
long getTime()	Devuelve el valor en milisegundos desde el 1/1/1970 a las 00:00 horas
void setTime(long hora)	Establece fecha hora pasándole el valor en milisegundos

SimpleDateFormat

Permite mostrar un valor Date en el formato de fecha que se quiera.

- Constructores:

```
SimpleDateFormat formatter = new SimpleDateFormat("dd-MM-yy hh:mm");

Date date = new Date();

String fecha = formatter.format(date);
```

En el constructor, se pasa un String con una serie de letras que especifican el formato en el que se va a sacar la fecha. Se puede ver el uso de cada letra en la documentación:

https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/text/SimpleDateFormat.html

- Validar si un String está en el formato deseado.

```
SimpleDateFormat formatter = new SimpleDateFormat("dd-MM-yyyy");

formatter.parse(fechax);//Comprueba si fechax está en el formato adecuado
```

Calendar

Permite convertir un tiempo en milisegundos en varios componentes útiles.

Proporciona una serie de métodos que permitirán hacer cálculos con fechas.

No tiene constructores, se crea un Calendar con la fecha y hora del momento así:

```
Calendar calendar = Calendar.getInstance();
```

También se puede obtener a partir de un Date, una vez utilizado getInstance();

```
calendar.setTime(unDate);
```

- Obtener datos

```
int dia = calendar.get(Calendar.DAY_OF_MONTH); //dia del mes
int mes = calendar.get(Calendar.MONTH);  //mes, de 0 a 11
int anio = calendar.get(Calendar.YEAR);  //año
int hora24 = calendar.get(Calendar.HOUR_OF_DAY); //hora
formato 24hs
int minutos = calendar.get(Calendar.MINUTE);
int segundos = calendar.get(Calendar.SECOND);
```

Calendar

- Cambiar valores

```
calendar.set(Calendar.HOUR_OF_DAY, 23);
calendar.set(Calendar.MINUTE, 59);
```

- Realizar cálculos

```
calendar.add(Calendar.DAY_OF_MONTH, 1);
calendar.add(Calendar.WEEK_OF_MONTH, 1);
calendar.add(Calendar.HOUR_OF_DAY, -2);
```

Se puede obtener el Date de Calendar para representar la fecha.

```
Date fecha = calendar.getTime();
```

Documentación:

https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/Calendar.html

Otras clases

- GregorianCalendar: Es una subclase de Calendar que implementa el calendario gregoriano (el nuestro), aunque lo normal es que Calendar ya lo implemente así.

```
Calendar cal = GregorianCalendar.getInstance();
```

- TimeZone y SimpleTimeZone: Permite trabajar con zonas horarias.

```
TimeZone timeZone = TimeZone.getTimeZone("Europe/Copenhagen");
calendar.setTimeZone (timeZone);
```

- Locale: para describir zonas regiones geográficas y culturales.

```
Locale localeVal = Locale.FRANCE;
SimpleDateFormat spf = new SimpleDateFormat("dd/mm/yy,localeVal);
```

- Random: generador de números aleatorios.

Date Time API

- Nueva API de Java para el manejo de fechas
- Incorporado en Java 8
- Basado en la librería joda-time
- Nos permite trabajar con fechas y horas bajo el estándar ISO8601 (JSR 310)

El tratamiento de fechas en Java hasta ahora daba muchos problemas y era bastante tedioso. La clase Date tenía muchos métodos obsoletos desde Java 2, y para tratar fechas había que combinar el uso de las clases Date y Calendar.

Date Time API

Java incorporó la nueva librería para el manejo de fechas con el fin de estandarizar el tratamiento de las mismas.

Clases Principales:

- **LocalDate**
- **LocalTime**
- **LocalDateTime**
- ZonedDateTime
- Duration (Horas)
- Period (Fechas)
- …

Nos va a ayudar a lidiar con problemas complejos:

- Años bisiestos
- Zonas horarias
- Horario de verano
- ...

Paquete *java.time*

https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/time/package-summary.html

Date Time API

Principios de diseño

Claro - El comportamiento de los métodos tiene que ser claro y esperado

Fluido - Fácil de leer, que permita encadenar varias llamadas a métodos. El código resultante tiene que poder leerse fácilmente.

Inmutable - Muchos de los objetos del api son inmutables, es decir, no modificables después de ser creados. Para modificar los valores debemos crear uno nuevo.

Extensible - Hasta donde los programadores sean capaces. Por ejemplo, permitiendo crear nuestro propio sistema de calendario.

Date Time API

Métodos comunes

El API de fechas de Java 8 posee distintas clases para manejarlas. La mayoría de ellas sigue esta convención sobre el nombre de los métodos que sirven para manejarlas, añadiéndole alguno de estos prefijos.

Nombre	Uso
of	Crea una instancia del objeto a partir de los datos de entrada
from	Convierte el parámetro de entrada en una instancia de la clase
parse	Procesa la cadena de entrada y construye una instancia
format	Procesa los datos para producir un String
get	Devuelve una parte del objeto

Date Time API

Métodos comunes

Nombre	Uso
is	Consultar el estado del objeto
with	Devolver una copia del objeto con uno o varios elementos modificados
plus	Devolver una copia del objeto con una cantidad de tiempo añadido
minus	Devolver una copia del objeto con una cantidad de tiempo sustraído
to	Convertir el objeto en otro Tipo
at	Combinar el objeto con otro objeto

La clase System

La clase System

Todos los programas de Java importan automáticamente el paquete java.lang que define la clase System. Contiene **tres variables de flujo predefinidas:**

- System.in - hace referencia al flujo estándar de entrada, que es el teclado.
- System.out - hace referencia al flujo estándar de salida, que es la consola.
- System.err - hace referencia al flujo de error estándar que también es la consola de forma predeterminada.

Métodos:

exit: termina el programa Java. Más bien termina una instancia de JVM que se está ejecutando actualmente. Cualquier línea insertada debajo del método de salida será inalcanzable y no se ejecutará. Puede devolver un “estado de salida”.

gc: invoca al *garbage collector* de java. Limpia el heap.

currentTimeMillis: Nos devuelve el tiempo en milisegundos

arrayCopy: copia una serie de elementos de un array a otro

La clase System

Esta clase nos ayuda a interactuar con el sistema, o el entorno en que el se está ejecutando. Cuando arranca el Sistema se fijan una serie de propiedades con formato pares clave/valor que podemos consultar gracias a esta clase.

getProperty(String key):

Devuelve el valor de la propiedad cuyo nombre se le pasa como argumento (se refiere al nombre de la clave)

setProperty(String key, String value):

modifica el valor del nombre de propiedad que se le pasa como argumento(se refiere al nombre de la clave), con el valor del segundo argumento.

Lista de las propiedades:

https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/lang/System.html#getProperties()

La clase Math

La clase Math

Incorpora constantes y funciones matemáticas como:

- double E, double PI..

Métodos:

abs: valor absoluto

funciones trigonométricas: cos, sin, tan..

ceil / floor: devuelven el entero menor o mayor

log: logaritmo en base E

max / min: maximo o minimo de 2 valores

pow: eleva un número a una determinada potencia

round: redondea siempre al entero más próximo

sqrt: calcula la raiz cuadrada

random: devuelve un número aleatorio positivo entre 0.0 y 1.0