1º CFGS Desarrollo De Aplicaciones Multiplataforma

UD 4. Utilización de objetos.

Módulo: Programación

Centro de Enseñanza
Gregorio Fernández

Macarena Cuenca Carbajo

Paquetes (packages)

- Un **paquete (package)**, es un conjunto de clases relacionadas entre sí, las cuales están ordenadas de forma arbitraria.
- Por ejemplo, el paquete **java.io** agrupa las clases relacionadas con la entrada/salida.
- Un paquete también puede contener a otros paquetes.
- Evitan los conflictos entre los nombres de las clases.

Librerías

- Conjunto de clases, que poseen una serie de métodos y atributos.
- Permiten reutilizar código, así no tenemos que implementar nosotros mismos esas funcionalidades.
- Existen librerías propias de Java, y también podemos crear las nuestras propias para usarlas en nuestros proyectos.
- Básicamente un paquete Java puede ser una librería. Pero una librería Java completa puede estar formada por muchos paquetes.
- Para usar las clases de una librería, con sus métodos y atributos, debemos **importarla**.

Sentencia import

- Si queremos utilizar una clase contenida en una librería, la forma de hacerlo (generalmente) es utilizando la sentencia **import**.
- Podemos importar una clase individual, como por ejemplo:

```
import java.lang.System; // se importa la clase System
```

- O bien podemos importar todas las clases de un paquete:

```
import java.swing.*;
```

- También es posible utilizar una clase contenida en una librería sin utilizar la sentencia import. En este caso, se pone la "ruta" a la clase en la propia instrucción de uso de la clase. Por ejemplo:

```
javax.swing.JFrame fr =new javax.swing.JFrame("Ventana de ejemplo");
```

- Generalmente, no se utiliza esta última opción porque implica escribir más código.

API de Java

- Java está formado por una gran cantidad de paquetes y clases.
- Podemos usar el API de Java para encontrar todas las clases que componen el lenguaje, el paquete al que pertenecen, los métodos que poseen, los atributos y una breve explicación de cada uno de ellos.
- Está disponible en la página oficial de Oracle.

  https://docs.oracle.com/en/java/javase/21/docs/api/index.html

API de Java

- Algunos de los paquetes más comunes son:

Paquete o librería	Descripción
java.io	Paquete de E/S. Permite la comunicación del programa con ficheros y periféricos.
java.lang	Paquete con clases esenciales de Java. Se importa por defecto en los programas, por lo que no es necesaria ejecutar la sentencia import para utilizar sus clases.
java.util	Paquete con clases de utilidad.
java.applet	Paquete para desarrollar applets. Un applet es una aplicación Java que se ejecuta en la ventana de un navegador. Los applets se ejecutan en el cliente.
java.awt	Paquete para el desarrollo de GUIs.

API de Java

Paquete o librería	Descripción
java.net	En combinación con el paquete java.io, permite crear aplicaciones que realicen comunicaciones con la red local e Internet.
java.math	Paquete con utilidades matemáticas.
java.sql	Paquete especializado en el manejo y comunicación con bases de datos.
java.security	Paquete que implementa mecanismos de seguridad.
java.rmi	Paquete que permite el acceso a objetos situados en otros equipos (objetos remotos).
java.beans	Paquete que permite la creación y manejo de componentes *javabeans*. Un *javabean* es un componente reutilizable que encapsula varios objetos en uno solo. *Bean* significa vaina en inglés, y como su nombre indica, permite tener un único objeto en vez de varios más simples.

API de Java
I – Wrappers

- Clases “envolventes” de los tipos básicos de Java:

Clase	Representa al tipo básico…
Void	void
Boolean	boolean
Character	char
Byte	byte
Short	short
Integer	int
Long	long
Float	float
Double	double

- No son equivalentes.
- Incluidas en el paguete **java.lang**

API de Java
I – Wrappers

- Ejemplos de creación:

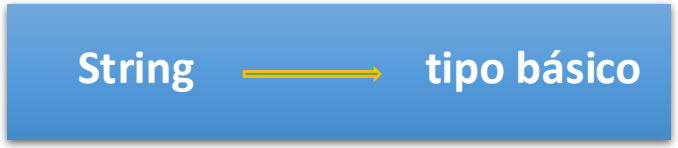
```
Integer i1=15;
Integer i2=Integer.valueOf(“24”);
Double d1=18.3;
Double d2=Double.valueOf(”23.5”);
```

- Si el texto pasado como parámetro no es convertible, se lanza una excepción del tipo **NumberFormatException**.
- Es común, realizar conversiones entre tipos básicos y Wrappers.

Macarena Cuenca Carbajo

Centro de Enseñanza
Gregorio Fernández

API de Java
I – Wrappers



- Método **estático** **parseX (String s)**
- X, es sustituido por el tipo básico a convertir.
- Ejemplos:

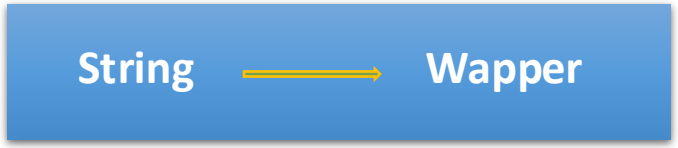
```
String s="2500";
int y=Integer.parseInt(s);          //String -> int
short z=Short.parseShort(s);        //String -> short
double c=Double.parseDouble(s);     //String -> double
```

- La clase Character no tiene el método **parse**, por eso para "convertir" un *String* a *char* tenemos que usar el método **charAt**.

Macarena Cuenca Carbajo

Centro de Enseñanza
Gregorio Fernández

API de Java
I – Wrappers



- Método **estático valueOf (String s)**
- Ejemplos:

```
String s="2500";
Integer a=Integer.valueOf(s);
Short b=Short.valueOf(s);
Double c=Double.valueOf(s);
```

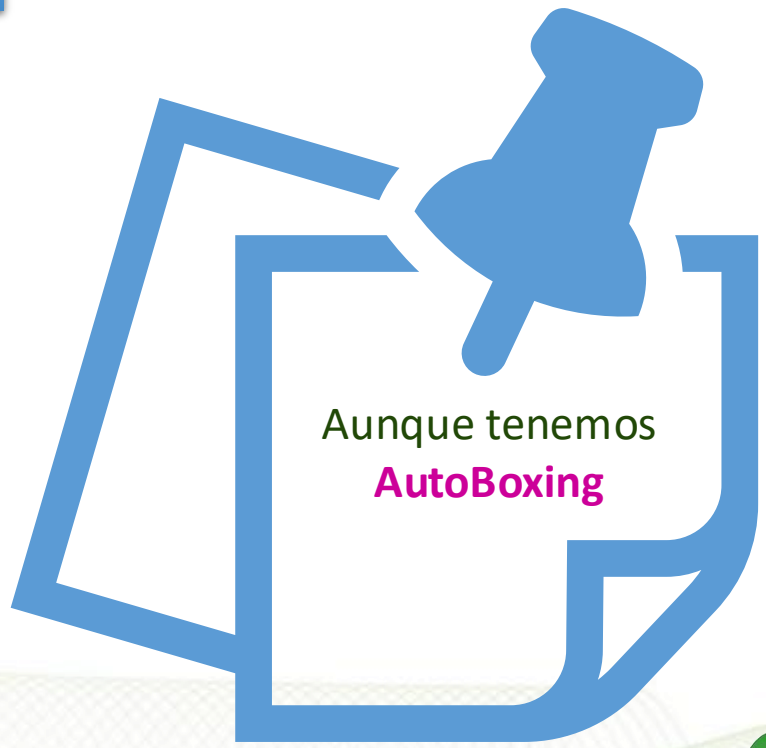

Macarena Cuenca Carbajo

API de Java
I – Wrappers

Wrapper ⟹ **tipo básico**

- Método **dinámico xValue()**
- x, es sustituido por el tipo básico a convertir.
- Ejemplos:

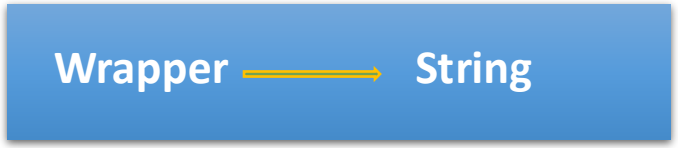
```
Integer i1=new Integer(4);
int i2=i1.intValue();
```



Macarena Cuenca Carbajo

Centro de Enseñanza
Gregorio Fernández

API de Java
I – Wrappers

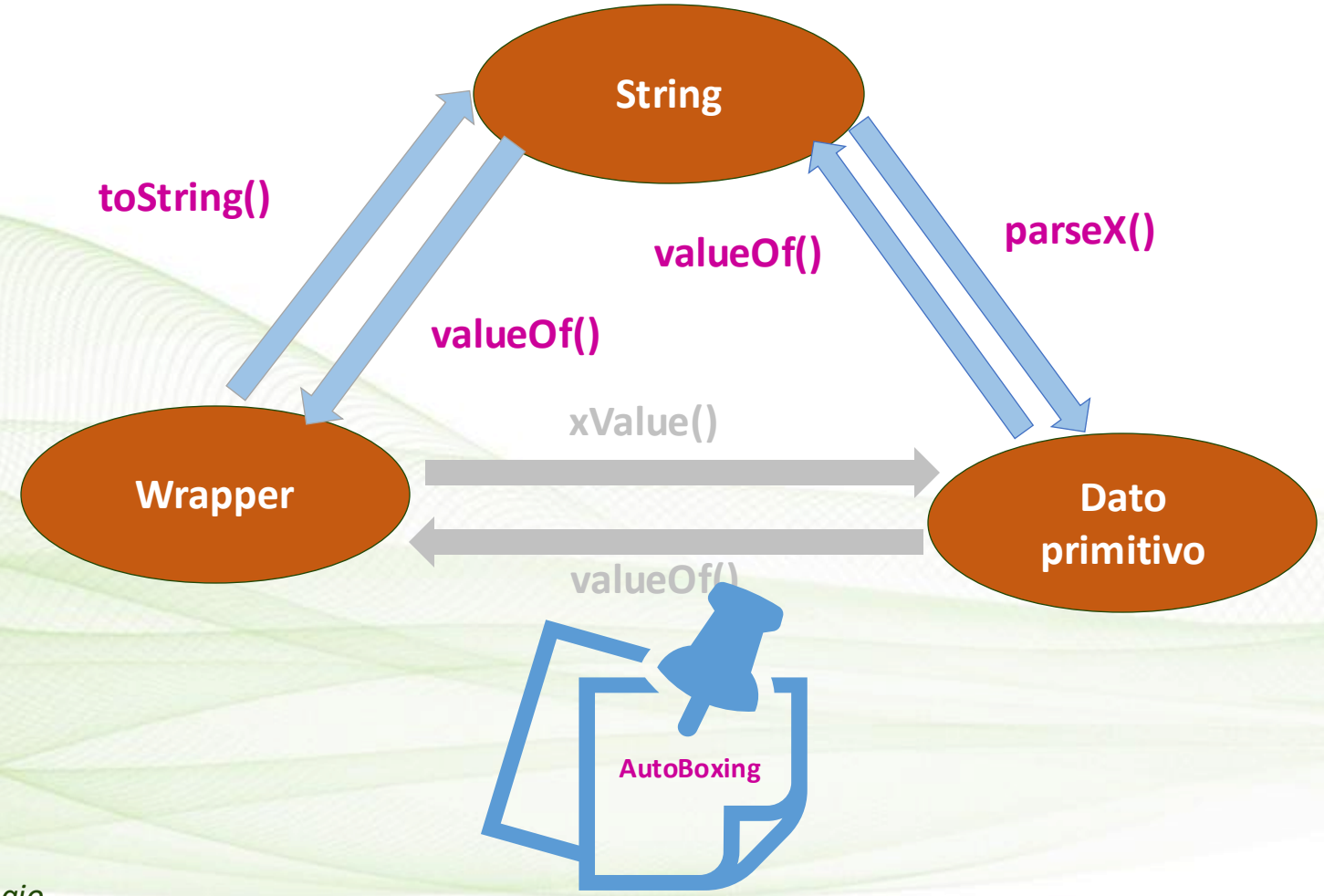


- Método **dinámico toString()**
- Ejemplos:

```
Integer i = new Integer(4);
String s = i.toString();
```

- Casi todos los objetos de java disponen de este método.

API de Java
I – Wrappers

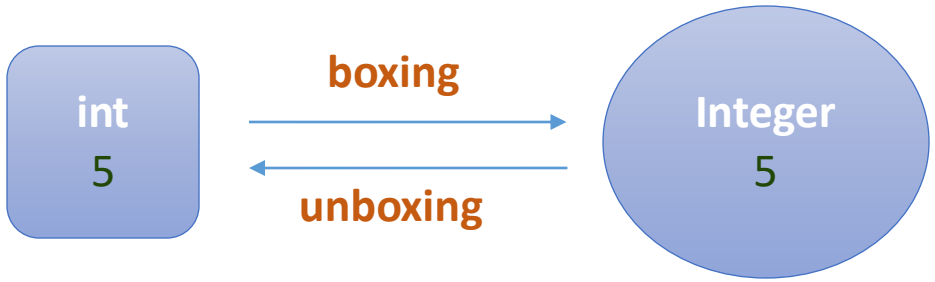


Macarena Cuenca Carbajo

Centro de Enseñanza
Gregorio Fernández

API de Java
I – Wrappers

- Conceptos:
  - ➜ Boxing
  - ➜ Unboxing
  - ➜ Autoboxing



- **Boxing**: convertir tipo básico en Wrapper.

  De esta forma podemos realizar operaciones con objetos.

API de Java
I – Wrappers

- **Autoboxing**: conversión automática que Java realiza entre tipos primitivos y Wrappers.

  Cuando por ejemplo, se asigna un dato primitivo a una variable de tipo Wrapper:

```
Character c = 'a';
```

  O al revés:

```
Integer i1 = Integer.valueOf(5);
int i2 = i1;
```

API de Java
II – clase Math

- Incluida en el paguete **java.lang**
- Posee métodos para realizar cálculos matemáticos.
- También dispone de las constantes **E** y **Pi**.

Macarena Cuenca Carbajo

Centro de Enseñanza
Gregorio Fernández

API de Java
III – clase Random

- Incluida en el paquete **java.util**
- Permite generar números aleatorios.
- Utiliza una **semilla** obtenida a partir de la fecha y la hora actual:

```
Random r1=Random();
```

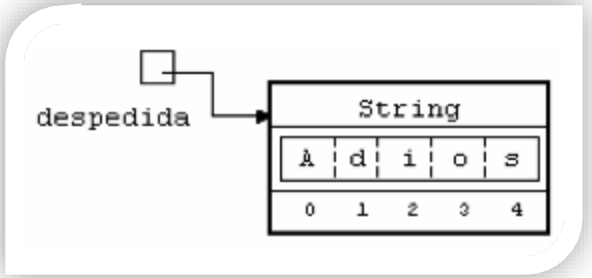
- También se puede cambiar de semilla:

```
Random r2=Random(182728L);
```

- Usar los métodos nextX, para generar aleatorios del tipo X. Ejemplos:

```
r1.nextInt();
r1.nextBoolean();
r1.nextDouble();
```

API de Java
IV – clase String



- Incluida en el paquete **java.lang**
- Permite crear **objetos** de cadenas de caracteres:

```
String cadena1=(“Hola”);
String cadena2=new String(“Adios”);
```

- Proporciona métodos para trabajar con las cadenas de caracteres.
- Los objetos de la clase **String** NO SON MODIFICABLES, es decir, los métodos que actúan sobre las cadenas devuelven un nuevo objeto con las modificaciones realizadas. Ejemplo:

```
String s="   Hola mundo          ";
s.trim(); //se crea un nuevo String
System.out.println(s); → "   Hola mundo          "
```


Macarena Cuenca Carbajo
Centro de Enseñanza
Gregorio Fernández

API de Java
V – Fechas

Clases:

- **java.util.Date**: representa un instante de tiempo (limitaciones).
- **java.util.Calendar**: también representa un instante de tiempo.
  - + Es mutable.
  - + Métodos para consultar, operar y modificar una fecha.
- **java.util.GregorianCalendar**: subclase de **Calendar**, representa fechas del calendario gregoriano.
- **java.text.DateFormat**: permite formatear fechas.
- **java.text.SimpleDateformat**: permite formatear fechas

Centro de Enseñanza
Gregorio Fernández

API de Java
V – Fechas

Clases:

- **Paquete java.time**
- A partir de Java 8 aparece este nuevo API para el manejo de fechas.
- Aporta diversas mejoras con respecto al “antiguo” tratamiento de fechas/horas, y por tanto se recomienda su uso siempre que se utilice una versión de Java que lo soporte.

API de Java
V – Fechas

Clase **java.util.Calendar**:

- Clase abstracta
- Permite crear objetos con el método **getInstance()**:

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

- Métodos **set**: establecen una fecha/instante concreta.
- Métodos **add**: permiten sumar valores a una fecha.
- Métodos **roll**: permite restar valores a una fecha.
- Atributos estáticos para hacer referencia a partes concretas de una fecha/instante de tiempo.

Java 8

API de Java
V – Fechas

Clase **java.util.GregorianCalendar**:

- Clase derivada/"hija" de **Calendar**.
- Hereda todas sus características.
- Permite trabajar con fechas/instantes de tiempo del calendario gregoriano.
- Constructores:

```
GregorianCalendar fecha1=new GregorianCalendar();

GregorianCalendar fecha2=new GregorianCalendar(2022,7,2);

GregorianCalendar fecha3=new GregorianCalendar(2022,Calendar.AUGUST,2);

GregorianCalendar fecha4=new GregorianCalendar(2022,7,2,12,30);

GregorianCalendar fecha5=new GregorianCalendar(2022,7,2,12,30,15);
```

Java 8

API de Java
V – Fechas

Clase **java.util.GregorianCalendar**:

- También podemos crear un objeto que represente la fecha actual de la siguiente forma:

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

- Método **get**: obtiene la parte de una fecha, indicada como parámetro (atributo de la clase **Calendar**).
- Método **set**: establece una parte de una fecha.
- Método **getTime**: obtiene el objeto **Date** equivalente.
- Método **setTime**: crea un objeto GregorianCalendar a partir de un objeto Date.

Java 8

Macarena Cuenca Carbajo

Centro de Enseñanza
Gregorio Fernández

API de Java
V – Fechas

Modo **lenient** / **non-lenient**:

- *Calendar* tiene 2 métodos de funcionamiento, lo que se llama “lenient” o “non-lenient” mode.
- Por defecto se trabaja en modo permisivo. Lo cual significa, que Java trata de encontrar siempre una fecha correcta.
- Por ejemplo, si creamos la fecha 32 de enero, al mostrarla por pantalla se mostrará el 1 de febrero.
- En cambio, si configuramos *Calendar* en modo no permisivo, el 32 de enero dará error.

API de Java
V – Fechas

Clase **java.util.Date**:

- Representa un **instante** de tiempo con precisión de milisegundos, desde el 1 de enero de 1970.
- En las versiones más recientes de Java muchos de los métodos de esta clase están "deprecated".
- Para reemplazar funcionalidades de esta clase aparece el paquete **java.time**.
- Ejemplos de creación de un objeto Date:

```
Date fecha1=new Date();
Date fecha2=(new GregorianCalendar(2022,7,6)).getTime();
```

- Métodos de utilidad: **after**, **before**, **equals**, **compareTo**,…

Java 8

Centro de Enseñanza
Gregorio Fernández

API de Java V – Fechas

Clases **java.time.LocalDate**

java.time.LocalTime

java.time.LocalDateTime

Java = Java 8

- Representan fechas, horas y fechas-horas respectivamente.
- Para instanciarlas se usan los métodos **now()** ya que sus constructores son privados.
- Para crear fecha/horas concretas se utilizan los métodos **of**.
- Hay disponibles una gran variedad de métodos para sumar/restar/comparar fechas: **plus**, **minus**, **isAfter**,...

API de Java V – Fechas

Java V=a8

- Clases **java.time.Period**
  java.time.Duration
- Podemos utilizar estas clase para modificar valores de fechas / horas o para obtener la diferencia entre 2 fechas / horas.

API de Java
VI – Formateo de fechas

Clase **java.text.DateFormat**:

- Permite formatear fechas.
- Formato básico:

```
DateFormat df = DateFormat.getInstance();
```

- Método **format**: devuelve un String con la fecha (Date) formateada, de acuerdo al formato creado con el objeto DateFormat. Ejemplo:

```
System.out.println(df.format(new Date()));
```

- Método **parse**: devuelve un Date que representa a la fecha pasada como parámetro (String).

  Puede lanzar excepciones de tipo **ParseException**.

API de Java
VI – Formateo de fechas

Clase **java.text.DateFormat**:

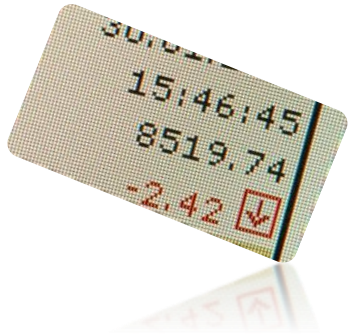
- Formatos sofisticados:
  - ✓ **DateFormat.getDateInstance.** Crea un formato de fecha sin la hora.
  - ✓ **DateFormat.getTimeInstance.** Crea un formato de hora sin la fecha.
  - ✓ **DateFormat.getDateTimeInstance**. Crea un formato de fecha hora.
- A estos métodos se les puede pasar alguno de los siguientes parámetros:
  - ✓ **DateFormat.SHORT.** Formato corto.
  - ✓ **DateFormat.MEDIUM.** Formato medio.
  - ✓ **DateFormat.LONG .** Formato largo.
  - ✓ **DateFormat.FULL.** Formato completo.

API de Java
VI – Formateo de fechas

Clase **java.text.SimpleDateFormat**:

- Clase derivada de **DateFormat** que permite formatear fechas de una forma más detallada.
- Siguiendo diferentes estilos, idiomas o regiones,...
- Para ello hay que crear un objeto utilizando plantillas.
- Consultar el **API** de Java.

API de Java
VII – Formatos numéricos



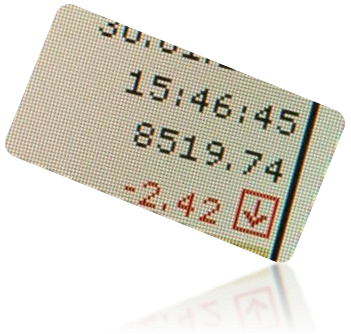
Clase **NumberFormat:**

- Permite formatear valores numéricos de varias formas.
- Para ello crearemos un objeto utilizando alguno de los siguientes métodos:

```
NumberFormat nf1=NumberFormat.getInstance();
NumberFormat nf2=NumberFormat.getNumberInstance();
NumberFormat nf3=NumberFormat.getCurrencyInstance();
NumberFormat nf4=NumberFormat.getPercentInstance();
```

- Método **format**: devuelve un String con el nº pasado como parámetro, de acuerdo, a las características del objeto NumberFormat creado.

API de Java
VII – Formatos numéricos



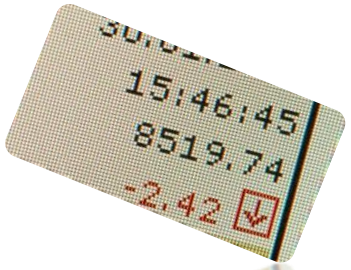
Clase **NumberFormat:**

- Método **parse**: inverso al anterior.
- Métodos de utilidad: existen diferentes métodos para consultar y configurar ciertas características del objeto NumberFormat.

Ejemplo:

```
NumberFormat nf=NumberFormat.getNumberInstance();
nf.setMaximumFractionDigits(2);
System.out.println(nf.format(123.4567));
```

API de Java
VII – Formatos numéricos



Clase **DecimalFormat:**

- Clase derivada de **NumberFormat** que permite dar formato a valores numéricos de una forma más detallada.
- Siguiendo diferentes estilos, idiomas o regiones,...
- Para ello hay que crear un objeto utilizando plantillas.
- Consultar el **API** de Java.

API de Java
VIII – Formateo de cadenas

Clase **String.format:**

- Es la forma más común de formatear una cadena en Java.
- Para salidas por consola también podemos usar el método **System.out.printf** ó **System.out.format**.
- Sintaxis:

  format (”cadena formateo”, argumentos)

- Donde:
  - ➜ **Cadena a formatear:** contiene **especificadores de formato** que puedes ver en las siguientes tablas. Cada uno de ellos produce una salida.
  - ➜ **Argumentos:** valores a formatear.

API de Java
VIII – Formateo de cadenas

	APLICABLE A	SALIDA
%a	floating point (except BigDecimal)	Hex output of floating point number
%b	Any type	"true" if non-null, "false" if null
%c	character	Unicode character
%d	integer (incl. byte, short, int, long, bigint)	Decimal Integer
%e	floating point	decimal number in scientific notation
%f	floating point	decimal number
%g	floating point	decimal number, possibly in scientific notation depending on the precision and value.
%h	any type	Hex String of value from hashCode() method.
%n	none	Platform-specific line separator.
%o	integer (incl. byte, short, int, long, bigint)	Octal number
%s	any type	String value
%t	Date/Time (incl. long, Calendar, Date and TemporalAccessor)	%t is the prefix for Date/Time conversions. More formatting flags are needed after this. See Date/Time conversion below.
%x	integer (incl. byte, short, int, long, bigint)	Hex string.

Macarena Cuenca Carbajo

API de Java
VIII – Formateo de cadenas

FLAG	NOTAS
%tA	Full name of the day of the week, e.g. "Sunday", "Monday"
%ta	Abbreviated name of the week day e.g. "Sun", "Mon", etc.
%tB	Full name of the month e.g. "January", "February", etc.
%tb	Abbreviated month name e.g. "Jan", "Feb", etc.
%tC	Century part of year formatted with two digits e.g. "00" through "99".
%tc	Date and time formatted with "%ta %tb %td %tT %tZ %tY" e.g. "Fri Feb 17 07:45:42 PST 2017"
%tD	Date formatted as "%tm/%td/%ty"
%td	Day of the month formatted with two digits. e.g. "01" to "31".
%te	Day of the month formatted without a leading 0 e.g. "1" to "31".
%tF	ISO 8601 formatted date with "%tY-%tm-%td".
%tH	Hour of the day for the 24-hour clock e.g. "00" to "23".
%th	Same as %tb.
%tI	Hour of the day for the 12-hour clock e.g. "01" – "12".
%tj	Day of the year formatted with leading 0s e.g. "001" to "366".
%tk	Hour of the day for the 24 hour clock without a leading 0 e.g. "0" to "23".
%tl	Hour of the day for the 12-hour click without a leading 0 e.g. "1" to "12".
%tM	Minute within the hour formatted a leading 0 e.g. "00" to "59".
%tm	Month formatted with a leading 0 e.g. "01" to "12".
%tN	Nanosecond formatted with 9 digits and leading 0s e.g. "000000000" to "999999999".
%tp	Locale specific "am" or "pm" marker.
%tQ	Milliseconds since epoch Jan 1 , 1970 00:00:00 UTC.
%tR	Time formatted as 24-hours e.g. "%tH:%tM".
%tr	Time formatted as 12-hours e.g. "%tI:%tM:%tS %Tp".
%tS	Seconds within the minute formatted with 2 digits e.g. "00" to "60". "60" is required to support leap seconds.
%ts	Seconds since the epoch Jan 1, 1970 00:00:00 UTC.
%tT	Time formatted as 24-hours e.g. "%tH:%tM:%tS".
%tY	Year formatted with 4 digits e.g. "0000" to "9999".
%ty	Year formatted with 2 digits e.g. "00" to "99".
%tZ	Time zone abbreviation. e.g. "UTC", "PST", etc.
%tz	Time Zone Offset from GMT e.g. "-0800".

Formateo de fechas y horas

Macarena Cuenca Carbajo

API de Java
VIII – Formateo de cadenas

Formateo de argumentos

- Podemos indicar cuál de los argumentos quiero formatear de la lista de argumentos, utilizando un índice acabado en **$**.
- Ejemplo de formateo de enteros:
  - ➜Con una longitud de 20 (rellena con espacios):

```
String.format("|%20d|", 93); // |                  93|
```

  - ➜Justificado a la izquierda:

```
String.format("|%-20d|", 93); // |93
```

  - ➜Rellenando con ceros:

```
String.format("|%020d|", 93);|00000000000000000093|
```

Actividades

Actividades Tema4

Macarena Cuenca Carbajo

Centro de Enseñanza
Gregorio Fernández