

# Programació

UT5. API de Java. Creació  
i maneig d'objectes.



# Introducció

- Quan treballem amb la classe `String`, hem vist que tots el mètodes que conté es poden consultar a través de seua API.
- Ara coneixerem què és l'API de Java i a moure'ns per la documentació.
- A més vorem algunes classes definides que poden ser útils per al desenvolupament de programes futurs.



# API de Java

- Java disposa d'un repositori de classes ja creades, que seran auxiliars per a la creació de programes propis.
- A este repositori l'anomenem API (Application Programming interface)
- En funció de la versió de Java consultada, este repositori varia.
- Els paquets més comuns es mantenen pràcticament en totes les versions fins a l'última versió (Java 17).



# API de Java (Versió 11)

<https://docs.oracle.com/en/java/javase/11/docs/api/>

B  
U  
S  
C  
A  
R

OVERVIEW

MODULE

PACKAGE

CLASS

USE

TREE

DEPRECATED

INDEX

HELP

Java SE 11 & JDK 11

ALL CLASSES

SEARCH:

## Java® Platform, Standard Edition & Java Development Kit Version 11 API Specification

This document is divided into two sections:

Java SE

The Java Platform, Standard Edition (Java SE) APIs define the core Java platform for general-purpose computing. These APIs are in modules whose names start with java.

JDK

The Java Development Kit (JDK) APIs are specific to the JDK and will not necessarily be available in all implementations of the Java SE Platform. These APIs are in modules whose names start with jdk.

All Modules

Java SE

JDK

Other Modules

Module	Description
java.base	Defines the foundational APIs of the Java SE Platform.
java.compiler	Defines the Language Model, Annotation Processing, and Java Compiler APIs.
java.datatransfer	Defines the API for transferring data between and within applications.
java.desktop	Defines the AWT and Swing user interface toolkits, plus APIs for accessibility, audio, imaging, printing, and JavaBeans.
java.instrument	Defines services that allow agents to instrument programs running on the JVM.
java.logging	Defines the Java Logging API.
java.management	Defines the Java Management Extensions (JMX) API.
java.management.rmi	Defines the RMI connector for the Java Management Extensions (JMX) Remote API.
java.naming	Defines the Java Naming and Directory Interface (JNDI) API.
java.net.http	Defines the HTTP Client and WebSocket APIs.
java.prefs	Defines the Preferences API.
java.rmi	Defines the Remote Method Invocation (RMI) API.
java.security	Defines the Java Security API.



# API Java. Mòduls destacats

- Dins de **java.base** destaquen els paquets:
  - `java.lang` - operacions essencials dels tipus de dades del llenguatge
  - `java.util` - propòsit general
  - `java.io` - entrada/eixida
- Dins de **java.desktop** destaca:
  - `javax.swing` - creació d'interfícies gràfiques bàsiques (A dia de hui pràcticament en desús professional, però útil per adquirir coneixements bàsics de programació d'interfícies gràfiques)



# API Java

- L'apartat "Constructor Summary" mostra totes les possibilitats amb les quals una classe pot ser inicialitzada.
- Estos mètodes es coneixen com a **constructors**
- Per exemple, per a la classe Random:

## Constructor Summary

Random()

Creates a new random number generator.

Random(long seed)

Creates a new random number generator using a single long seed.



# Invocació de constructors

- Per a invocar un constructor, hem de fixar-nos en primer lloc, quants paràmetres d'entrada requereix.
- Una volta coneguda esta informació, podrem crear un objecte d'eixa classe fent ús de la paraula reservada **new**.
- Exemple per al **primer constructor de Random**:

**Random objecteRandom = new Random ();**



no oblidem  
que és una  
referència



# Pràctica 1

- Crea un programa que genere 2 nombres reals aleatoris i els mostre per pantalla, fent ús de la classe Random (constructor sense paràmetres). Busca esta classe en l'API de Java i utilitza el mètode que consideres més adequat.



# Pràctica 2

- Modifica el programa anterior per a que utilitze el constructor que requereix un paràmetre de tipus long (el segon).
- Quina diferència hi ha entre crear un objecte Random amb un o un altre constructor?



# Mètodes estàtics

- Fins ara, per a poder invocar a mètodes de l' API, hem vist que es requereix crear un objecte de la classe.
- Existeixen també mètodes que no requereixen d'esta inicialització per a ser usats. Els identifiquem perquè apareixen marcats amb la paraula reservada **static**.
- Un exemple d'este tipus de mètodes els trobem per exemple a la classe **Math**



# Mètodes estàtics:

## Exemple per a la classe Math.

<code>static long</code>	<code><u>abs</u>(long a)</code> Returns the absolute value of a long value.
<code>static double</code>	<code><u>acos</u>(double a)</code> Returns the arc cosine of a value; the returned angle is in the range 0.0 through $\pi$ .
<code>static double</code>	<code><u>asin</u>(double a)</code> Returns the arc sine of a value; the returned angle is in the range $-\pi/2$ through $\pi/2$ .
<code>static double</code>	<code><u>atan</u>(double a)</code> Returns the arc tangent of a value; the returned angle is in the range $-\pi/2$ through $\pi/2$ .

```
double resultat = Math.sqrt(36);
```



# Pràctica 3

- Crea un programa que genere un nombre real aleatori i l'arrodonisca. Utilitza la classe **Random** i **Math** per a fer-ho.



# Pràctica 4

- Crea un programa que calcule la hipotenusa d'un triangle. L'usuari haurà d'introduir els dos valors dels catets. Utilitza la classe Math per a calcular el quadrat i l'arrel quadrada.



# La classe Arrays

- Pertany al paquet `java.util`
- Ofereix mètodes estàtics per a realitzar operacions sobre arrays:
  - Ordenació
  - Cerca
  - Còpia
  - Comparació
  - Transformació a text



# Pràctica 5

- Realitza proves sobre un array ja creat, fent ús dels diferents mètodes **static** que realitzen les operacions que s'han enumerat a la diapositiva anterior.



# La classe `StringBuilder`

- És una classe similar a `String`, amb la diferència de que en este cas l'objecte `StringBuilder` sí que és mutable.
- Són més flexibles que `String`, ja que es pot inserir o afegir informació, a diferència d'`String`, que una volta creat l'objecte no es pot realitzar cap d'eixes accions.
- **`StringBuilder`** pot ser després transformat a **`String`** i viceversa.
- Pot ser mostrat el seu valor a través de **`System.out`**



# Constructors of `StringBuilder`

## `java.lang.StringBuilder`

- `+StringBuilder()`
- `+StringBuilder(capacity: int)`
- `+StringBuilder(s: String)`

- Constructs an empty string builder with capacity 16.
- Constructs a string builder with the specified capacity.
- Constructs a string builder with the specified string.



# Mètodes que modifiquen un objecte

## StringBuilder

### java.lang.StringBuilder

```
+append(data: char[]): StringBuilder
+append(data: char[], offset: int, len: int):
  StringBuilder
+append(v: aPrimitiveType): StringBuilder

+append(s: String): StringBuilder
+delete(startIndex: int, endIndex: int):
  StringBuilder
+deleteCharAt(index: int): StringBuilder
+insert(index: int, data: char[], offset: int,
  len: int): StringBuilder
+insert(offset: int, data: char[]):
  StringBuilder
+insert(offset: int, b: aPrimitiveType):
  StringBuilder
+insert(offset: int, s: String): StringBuilder
+replace(startIndex: int, endIndex: int, s:
  String): StringBuilder
+reverse(): StringBuilder
+setCharAt(index: int, ch: char): void
```

Appends a `char` array into this string builder.

Appends a subarray in `data` into this string builder.

Appends a primitive type value as a string to this builder.

Appends a string to this string builder.

Deletes characters from `startIndex` to `endIndex-1`.

Deletes a character at the specified index.

Inserts a subarray of the data in the array into the builder at the specified index.

Inserts data into this builder at the position offset.

Inserts a value converted to a string into this builder.

Inserts a string into this builder at the position offset.

Replaces the characters in this builder from `startIndex` to `endIndex-1` with the specified string.

Reverses the characters in the builder.

Sets a new character at the specified index in this builder.



# Altres mètodes de StringBuilder

## java.lang.StringBuilder

```
+toString(): String  
+capacity(): int  
+charAt(index: int): char  
+length(): int  
+setLength(newLength: int): void  
+substring(startIndex: int): String  
+substring(startIndex: int, endIndex: int):  
    String  
+trimToSize(): void
```

Returns a string object from the string builder.  
Returns the capacity of this string builder.  
Returns the character at the specified index.  
Returns the number of characters in this builder.  
Sets a new length in this builder.  
Returns a substring starting at `startIndex`.  
Returns a substring from `startIndex` to `endIndex-1`.  
  
Reduces the storage size used for the string builder.



# Pregunta 1

Suposant que s1 i s2 es declaren de la següent forma:

```
StringBuilder s1 = new StringBuilder("Java");  
StringBuilder s2 = new StringBuilder("HTML");
```

Quin valor adquireix s1 després de cada sentència?

```
a. s1.append(" is fun");  
b. s1.append(s2);  
c. s1.insert(2, "is fun");  
d. s1.insert(1, s2);  
e. s1.charAt(2);  
f. s1.length();
```

```
g. s1.deleteCharAt(3);  
h. s1.delete(1, 3);  
i. s1.reverse();  
j. s1.replace(1, 3, "Computer");  
k. s1.substring(1, 3);  
l. s1.substring(2);
```



# Pregunta 2

Què es mostra per pantalla?

```
public class Pregunta2 {  
  
    public static void main(String[] args) {  
        Pregunta2 pregunta2 = new Pregunta2();  
        pregunta2.inicio();  
    }  
  
    public void inicio() {  
        String s = "Java";  
        StringBuilder builder = new StringBuilder(s);  
        change(s, builder);  
  
        System.out.println(s);  
        System.out.println(builder);  
    }  
  
    private void change(String s, StringBuilder builder) {  
        s = s + " and HTML";  
        builder.append(" and HTML");  
    }  
}
```



## Pregunta 3

Quin mètode de l' API d'**StringBuilder** usaries per obtindre un **String** a partir d'un **StringBuilder**?



# Pràctica 6

- Prova en un programa nou alguns dels mètodes mostrats d'**StringBuilder** per a tindre encara més clar què fa cadascun.



# Classes envoltori (Wrapper classes)

- Es denominen classes envoltants o envoltori a les que permeten tractar una dada de tipus bàsic com a un objecte
- Existeixen ja que molts mètodes de l' API de Java, demanden com a paràmetre un objecte i no un tipus de dada primitiu.
- Són molt fàcil de conèixer ja que el seu nom és igual que el tipus de dada primitiu, però començant amb majúscula.

**Boolean, Integer, Double, Float, Byte, Short, Long i Character.**



# Classes Integer i Double

## java.lang.Integer

-value: int  
+MAX\_VALUE: int  
+MIN\_VALUE: int

Propietats

+Integer(value: int)  
+Integer(s: String)  
+byteValue(): byte  
+shortValue(): short  
+intValue(): int  
+longValue(): long  
+floatValue(): float  
+doubleValue(): double  
+compareTo(o: Integer): int  
+toString(): String  
+valueOf(s: String): Integer  
+valueOf(s: String, radix: int): Integer  
+parseInt(s: String): int  
+parseInt(s: String, radix: int): int

Mètodes

## java.lang.Double

-value: double  
+MAX\_VALUE: double  
+MIN\_VALUE: double

+Double(value: double)  
+Double(s: String)  
+byteValue(): byte  
+shortValue(): short  
+intValue(): int  
+longValue(): long  
+floatValue(): float  
+doubleValue(): double  
+compareTo(o: Double): int  
+toString(): String  
+valueOf(s: String): Double  
+valueOf(s: String, radix: int): Double  
+parseDouble(s: String): double  
+parseDouble(s: String, radix: int): double



# Pregunta 4

¿Compilen les següents sentències?

- a. `Integer i = new Integer("23");`
- b. `Integer i = new Integer(23);`
- c. `Integer i = Integer.valueOf("23");`
- d. `Integer i = Integer.parseInt("23", 8);`
- e. `Double d = new Double();`
- f. `Double d = Double.valueOf("23.45");`
- g. `int i = (Integer.valueOf("23")).intValue();`
- h. `double d = (Double.valueOf("23.4")).doubleValue();`
- i. `int i = (Double.valueOf("23.4")).intValue();`
- j. `String s = (Double.valueOf("23.4")).toString();`



# Pregunta 5

Com es converteix ....

- Un enter a String?
- Un String numèric a enter?
- Un real a un String?
- Un String numèric a real?



# Pregunta 6

Què es mostra per pantalla?

```
public class Test {  
    public static void main(String[] args) {  
        Integer x = new Integer(3);  
        System.out.println(x.intValue());  
        System.out.println(x.compareTo(new Integer(4)));  
    }  
}
```



# Pregunta 7

Què es mostra per pantalla?

```
public class Test {  
    public static void main(String[] args) {  
        System.out.println(Integer.parseInt("10"));  
        System.out.println(Integer.parseInt("10", 10));  
        System.out.println(Integer.parseInt("10", 16));  
        System.out.println(Integer.parseInt("11"));  
        System.out.println(Integer.parseInt("11", 10));  
        System.out.println(Integer.parseInt("11", 16));  
    }  
}
```



# Conversió automàtica entre classes envoltori i tipus primitius

- Fer la conversió d'un tipus de dada primitiu a un objecte de la classe envoltori es denomina ***boxing***
- El contrari ***unboxing***
- El compilador pot realitzar ***boxing*** i ***unboxing*** de manera automàtica (***autoboxing*** i ***autounboxing***)




# Exemple de conversió automàtica

```
Integer intObject = new Integer (2);
```

EQUIVALENT A

```
Integer intObject = 2;
```

autoboxing





# Pregunta 8

On es produeix *autoboxing* i on *autounboxing*? Són totes correctes?

- a. `Integer x = 3 + new Integer(5);`
- b. `Integer x = 3;`
- c. `Double x = 3;`
- d. `Double x = 3.0;`
- e. `int x = new Integer(3);`
- f. `int x = new Integer(3) + new Integer(4);`



# Pregunta 9

Què es mostra per pantalla?

```
public class Test {  
    public static void main(String[] args) {  
        Double x = 3.5;  
        System.out.println(x.intValue());  
        System.out.println(x.compareTo(4.5));  
    }  
}
```