P. O. O. en Java : modelo de clase	
Variables miembro | atributos :	Guardan valores del objeto, representan una propiedad determinada suya.
	Pueden ser tipos primitivos o otros objetos (agregación).
Métodos | Rutinas :	Componente de un objeto que lleva a cabo una determinada acción o tarea con los atributos.
	En principio, todas las variables y rutinas de un programa de Java deben pertenecer a una clase.
-	
Clases:	Una clase representa al conjunto de objetos que comparten una estructura y un comportamiento comunes.
	Una clase es una combinación específica de atributos y métodos y puede considerarse un tipo de dato de cualquier tipo no primitivo.
	Una clase también puede estar compuesta por métodos estáticos que no necesitan de objetos (como main).
	[modificadores] class IdentificadorClase { //Declaraciones de atributos y metodos}
Constructor	
llamar a constructor principal para coger ya definidas	public KeywordTest(String name, int age) { this();
call the parameterized constructor from the no argument constructor and pass some arguments:	public KeywordTest() { this("John", 27); }
a reference to the current instance.	printInstance(this);
access the outer class instance from within the inner class:	public class KeywordTest { private String name; class ThisInnerClass { boolean isInnerClass = true; public ThisInnerClass() { KeywordTest thisKeyword = KeywordTest.this; String outerString = KeywordTest.this.name; } } }
acceder a constructor de la superclase	super()
Atributos de instancia	Diferentes para cada objeto
Atributos de clase	public static int cantidadDeOjos = 2;
Variable local	variable en scope de método

Scope	
El segmento del programa donde la variable es válida y puede usarse.	
de clase	Las variables private dentro de la clase pero fuera del método tienen Scope de clase : pueden usarse dentro de la misma clase epro no fuera
de método (variable local)	Variable declarada dentro de un método que sólo peude usarse dentro de ese método.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Acceso a miembros de una clase	
Nivel general	public o sin modificar explícito
Nivel de miembros	public, private, protected o sin modificar explícito

Modificadores de visibilidad	
public	acceso desde cualquier clase
protected	acceso desde la misma clase o clases heredadas, acceso desde su mismo paquete (package-private)
private	acceso desde la misma clase que los define

Modificadores	
static	
	static means that the method belongs to the MyClass class and not an object of the MyClass class.
	Se crea una sóla copia o instancia del miembro, no importa cuántas instancias de clase se creen.
	Se pueden llamar sin instancia.
	Es posible acceder a ellos sin crear instancias de clase.
	Usar cuando el valor es común para todos los objetos.
	USAR CUANDO SU VALOR ES INDEPENDIENTE DEL OBJETO.
	Cuando tiene sentido usar el mimebro aunque el objeto aún no se haya construído
	Cuando el mimebro no va a cambiar entre instancias de Clase u Objeto
static (en variables)	Delimita un valor para la variable de clase que compartirán todas las instancias de ese objeto
	Declarándoles un valor como variable de clase, hacemos que esa variable sea común para todas las instancias del objeto.
static (métodos)	Sólo pueden acceder a atributos estáticos o llamara otros métodos estáticos
	Usan propiedades del objeto sin crearlo.
	To access/manipulate static variables and other static methods that don't depend upon objects
	tatic methods in Java are resolved at compile time. Since method overriding is part of Runtime Polymorphism, so static methods can't be overridden
static Blocks	public static List<String> ranks = new LinkedList<>(); static { ranks.add("Lieutenant"); ranks.add("Captain"); ranks.add-("Major"); }
clases estáticas anidadas	public class Singleton { private Singleton() {} private static class SingletonHolder { public static final Singleton instance = new Singleton(); }
final	Define valor permanente a variable.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Modificadores (cont)

Acceso	
public	The code is accessible for all classes
private	The code is only accessible within the declared class
default	The code is only accessible in the same package.
protected	The code is accessible in the same package and subclasses.
No acceso (clase)	
final	The class cannot be inherited by other classes
abstract	The class cannot be used to create objects (To access an abstract class, it must be inherited from another class.
No acceso (atributos y métodos)	
final	Attributes and methods cannot be overridden/modified
static	Attributes and methods belongs to the class, rather than an object
abstract	Can only be used in an abstract class, and can only be used on methods. The method does not have a body, for example abstract void run();. The body is provided by the subclass (inherited from).
transient	Attributes and methods are skipped when serializing the object containing them
synchr-onized	Methods can only be accessed by one thread at a time
volatile	The value of an attribute is not cached thread-locally, and is always read from the "main memory"

? - Si no pongo static y modifico una variable de una instancia con this, cambia de valor en la instancia?

Dudas

? - Sumar hasta llegar a un número
-

```
int num = 10;
int sumas = 0;
int suma = 0;
while(suma < num){
   suma += sumas;
   System.out.println("sumas "+sumas+" suma:"+suma);
    sumas++;
}
```

? -Conseguir método privado desde main
- Hacer un método que devuelva el método privado (si es un boolean, p. e.)
?Simplificar if var x == x, x == y etc
?Clases abstractas en Java
?Suclases y sueprclases en java
?Ficheros aleatorios
?Interfaces
?métodos sobrecargados

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Bucles

for (int n : numeros){}

The break statement can also be used to jump out of a loop.

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

Clase Scanner

Leer char	char x1 = sc.next().charAt(0);

Abstracción

Clase sobre la que no se peuden ahcer objetos

debe tener mínimo un método abstracto que todas sus subclases van a heredar obligatoriamente

puede tener otros métodos normales, los cuales se puden hacer override o llamar desde subclase o superclase para el método original(el override -> especialización)

pueden tener constructor y sus subclases pueden usarlo

a diferencia de als interfaces, no hay por qué heredar todos los métodos

ABSTRACTA	SUBCLASE
abstract A();	@Override A(){"a"}
public B(){"padre"}	@Override C(){"hija"}
public C(){"padre"}	
-	
SUBCLASE.A();	"a"
SUBCLASE.B();	"padre"
SUBCLASE.C();	"hija"

| SUBCLASE

@Override

Interfaces

Métodos de String

str.substring(indexStart[, indexEnd])	console.log(str.substring(1, 3)); // expected output: "oz"
	console.log(str.substring(2)); // expected output: "zilla"

char charAt (int indice): devuelve el carácter que se encuentra en la posición de índice.

int compareTo (String cadena)	If first string is lexicographically greater than second string, it returns positive number (difference of character value). If first string is less than second string lexicographically, it returns negative number and if first string is lexicographically equal to second string, it returns 0

int compareToIgnoreCase (String cadena): compara dos cadenas (igual que el anterior) pero no diferencia entre mayúsculas y minúsculas.

- Boolean equals (Object objeto): devuelve True si el objeto que se pasa por parámetro y el string son iguales. Si no, devuelve False.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Métodos de String (cont)

int indexOf (int carácter): devuelve la posición de la primera vez que aparece el carácter en la cadena de caracteres. Como el carácter es de tipo entero, se debe introducir el valor del carácter correspondiente en código ASCII.	
boolean isEmpty (): si la cadena es vacía, devuelve True, es decir, si su longitud es cero.	
- int length (): devuelve el número de caracteres de la cadena.	
String replace (char caracterAntiguo, char caracterNuevo): devuelve una cadena que reemplaza el valor de carácterAntigüo por el valor del carácterNuevo.	The java string replace() method returns a string replacing all the old char or CharSequence to new char or CharSequence.
String [] Split (String expresión): devuelve un array de String con los elementos de la cadena expresión.	
String toLowerCase (): devuelve un array en el que aparecen los caracteres de la cadena que hace la llamada al método en minúsculas.	
String toUpperCase (): devuelve un array en el que aparecen los caracteres de la cadena que hace la llamada al método en mayúsculas	
- String trim (): devuelve una copia de la cadena, pero sin los espacios en blanco.	
String valueOf (tipo variable): devuelve la cadena de caracteres que resulta al convertir la variable del tipo que se pasa por parámetro	Returns the string representation of the x argument.

Estructuras de lista

Arrays	-Permiten guardar valores del mismo tipo -Tamaño fijo
Arrays (crear)	int numeros[] = new int[10];
inicializar	myNum = {10, 20, 30, 40};
Arrays (crear e inicializar)	int n[] = new int[]{1,2,3,4,5,6,7,8,9,10};
acceder	System.out.println(cars[0]);
	cars[0] = "Opel";
Arrays multidimensionales (crear)	int[][] temperaturas = new int[7][2];
crear e inicializar	int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
Arrays multidimensionales (acceder)	for (int i = 0; i < myNumbers.length; ++i) { for(int j = 0; j < myNumbers[i].length; ++j) {
	int x = myNumbers[1][2];
ArrayList	-Permiten crear listas de objetos. -Tamaño dinámico
ArrayList (crear)	ArrayList<String> cars = new ArrayList<String>();
ArrayList de más de un tipo (crear)	ArrayList a = new ArrayList<>(); a.add("a"); a.add(true); a.add(1);
ArrayList - métodos	al.size(); int
Añade al final	add(E e); true
Añade en pos	add(int index, Object element); true
Sustituye el elemento en la pos con el nuevo	cars.set(0, "Opel"); el elemento anterior
Devolver el elemento en la pos especificada	.get(int index);
Borra el elmento en la pos especificada	remove(int index); el elemento borrado
Borra el elemento si coincide	remove(Object o); bool

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Estructuras de lista (cont)	
	clear()
	Collections.sort(cars);
	isEmpty(); bool
	contains(Object elem); bool
	indexOf(Object elem); int
	lastIndexOf(Object elem); int
	clone(); arrList
	toArray(); arr
	No se puede añadir en pos donde esa pos no existe aka arrl está vacío
HashMap	
Create a HashMap object called capitalCities that will store String keys and String values:	HashMap<String, String> capitalCities = new HashMap<String, String>();
añadir key y valor	// Add keys and values (Country, City) capitalCities.put("England", "London");
acceder emdiante key	capitalCities.get("England");
borrar mediante key	capitalCities.remove("England");
	capitalCities.clear();
	capitalCities.size();
loop keys	// Print keys for (String i : capitalCities.keySet()) { System.out.println(i); }
loop values	for (String i : capitalCities.values()) { System.out.println(i); }
llXi	

Ficheros	
Ler dende ficheiro	FileInputStream fis = new FileInputStream(ficheiro);
	ObjectInputStream ois = new ObjectInputStream(fis);
Leer objeto de un fichero	Humidade objL = (Humidade)ois.readObject();
Leer objetos de un fichero	

Excepciones	
	} catch (Exception e) {
	The finally statement lets you execute code, after try...catch, regardless of the result
custom error	f (age < 18) { throw new ArithmeticException("Access denied - You must be at least 18 years old.");

Packages	
Crear	package mypack; class MyPackageClass {javac MyPackageClass.java

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Herencia

Heredar	class Vehicle {
	protected String brand = "Ford";
	public void honk() {
	System.out.println("Tuut, tuut!");
	}
	}
-	
	class Car extends Vehicle {
	private String modelName = "Mustang"; // Car attribute
	// Car attribute public static void main(String[] args) {
	// Create a myCar object Car myCar = new Car();
	// Call the honk() method (from the Vehicle class) on the myCar object myCar.honk();
	// Display the value of the brand attribute (from the Vehicle class) and the value of the modelName from the Car class System.out.println(myCar.brand + " " + myCar.modelName);
	}
	}
	Subclase hereda atributos y miembros
-	Brand is protected so subclass can use it.
	La subclase puede usar lo heredado o hacer override (a menos que sea final), y puede crear nuevos métodos
	Sólo se puede heredar de una clase

Poliformismo

abstracción, herencia, especialización	https://jarroba.com/polimorfismo-en-java-parte-i-con-ejemplos/

java.util.regex

< $, ^, ., *, +, ?, [,], . >

util.regex.Pattern	Used for defining patterns
compile the given regular expression into a pattern	compile(String regex)
	Pattern patron = Pattern.compile("camion");
util.regex.Matcher	Used for performing match operations on text using patterns
	Matcher encaja = patron.matcher()
matcher(CharSequence input)	It is used to create a matcher that will match the given input against this pattern.

???

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Estructuras de control

switch(a){ case 1: break; default: }

Declarar un String

//Declaración de un array que pasaremos a String char [] array = new char[]{'l','i','t','e','r','a','l'}; //Formas de declarar un String String forma1 = new String ("literal_cadena_caracteres"); String forma2 = "literal_cadena_caracteres"; String forma3 = new String (array); String forma4 = new String (forma2);

Colecciones/Collections

A Collection is a group of individual objects represented as a single unit. Java provides Collection Framework which defines several classes and interfaces to represent a group of objects as a single unit.

The Collection interface (java.util.Collection) and Map interface (java.util.Map) are the two main “root” interfaces of Java collection classes.

Need for Collection Framework :	Before Collection Framework (or before JDK 1.2) was introduced, the standard methods for grouping Java objects (or collections) were Arrays or Vectors or Hashtables. All of these collections had no common interface.
Set (Conjunto)	an unordered collection of objects in which duplicate values cannot be stored.
	Set is implemented by HashSet, LinkedHashSet or TreeSet
	https://www.geeksforgeeks.org/set-in-java/
List	an ordered collection of objects in which duplicate values can be stored. Since List preserves the insertion order, it allows positional access and insertion of elements.
	List Interface is implemented by ArrayList, LinkedList, Vector and Stack classes.
	https://www.geeksforgeeks.org/list-interface-java-examples/
Stack (PIla)	Java Collection framework provides a Stack class which models and implements Stack data structure. The class is based on the basic principle of last-in-first-out. In addition to the basic push and pop operations, the class provides three more functions of empty, search and peek. The class can also be said to extend Vector
	https://www.geeksforgeeks.org/stack-class-in-java/
	- Introducir un nuevo elemento sobre la cima (push).
	Eliminar un elemento de la cima (pop).

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Colecciones/Collections (cont)	
Queue (Cola)	The Queue interface is available in java.util package and extends the Collection interface. The queue collection is used to hold the elements about to be processed and provides various operations like the insertion, removal etc. It is an ordered list of objects with its use limited to insert elements at the end of the list and deleting elements from the start of list i.e. it follows the FIFO or the First-In-First-Out principle.
	https://www.geeksforgeeks.org/queue-interface-java/
	Encolar (enqueue): para ir añadiendo elementos
	Desencolar (dequeue): para eliminar elementos.
-	
Iterators	
Este iterador nos proporcionará unos métodos propios de este, que facilitarán recorrer este objeto collection, a continuación, vemos los métodos más utilizados: - next (): devuelve el siguiente elemento en la iteración. - hasNext (): devuelve verdadero si la iteración tiene más elementos, en caso contrario devuelve falso. - remove(): elimina de la colección subyacente el último elemento devuelto por este iterador.	
Iterator <String> iterador = nombre.iterator(); while (iterador.hasNext()) { String nombre = iterador.next(); //código bucle while } //Con las listas podemos utilizar un bucle for mejorado for (String.nombre : nombres) { //código bucle for each }	
--	
Not a collection -	
Map	The java.util.Map interface represents a mapping between a key and a value. The Map interface is not a subtype of the Collection interface. Therefore it behaves a bit different from the rest of the collection types
	A Map cannot contain duplicate keys and each key can map to at most one value. Some implementations allow null key and null value like the HashMap and LinkedHashMap, but some do not like the TreeMap.
	There are two interfaces for implementing Map in java: Map and SortedMap, and three classes: HashMap, TreeMap and LinkedHashMap.
	https://www.geeksforgeeks.org/map-interface-java-examples/
	Maps are perfect to use for key-value association mapping such as dictionaries. The maps are used to perform lookups by keys or when someone wants to retrieve and update elements by keys.
---LIST INTERFACES	

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Colecciones/Collections (cont)	
ArrayList	
//Instancia de tipo Genérico (Cuando no sabemos qué tipo de datos vamos a introducir en la colección):	ArrayList nombre = new ArrayList ();
//Instancia de colección con tipo específico:	ArrayList <Tipo_de_dato> nombre = new ArrayList <Tipo_de_dato> ();
get (int):	obtiene el objeto que se encuentra en la posición pasada por parámetro.
indexOf (Object):	
isEmpty ()	
set (int, Objeto)	sobrescribe el elemento que se encuentra en la posición int por un objeto que se indica por parámetro
toArray(): devuelve una lista de objetos y, en cada casilla de esta lista, inserta un objeto de ArrayList.	
toString()	
.add	
.remove	
LinkedList	
//Instancia de tipo Genérico (Cuando no sabemos qué tipo de datos vamos a introducir en la colección): LinkedList nombre = new LinkedList (); //Instancia de colección con tipo específico: LinkedList <Tipo_de_dato> nombre = new LinkedList <Tipo_de_dato> ();	
removeFirst(): elimina el primer elemento de la lista enlazada. - addFirst(): añade un elemento al principio de la lista. - addLast(): añade un elemento al final de la lista. - getFirst(): devuelve el primer elemento de la lista. - getLast(): devuelve el último elemento de la lista	
Vector	Dispone de un array de objetos que puede aumentar o disminuir de forma dinámica según las operaciones que se vayan a llevar a cabo
//Instancia de tipo Genérico (Cuando no sabemos qué tipo de datos vamos a introducir en la colección): Vector nombre = new Vector (); //Instancia de colección con tipo específico: Vector <Tipo_de_dato> nombre = new Vector <Tipo_de_dato> (); Algunos de sus métodos más importantes son: - firstElement (): devuelve el primer elemento del vector. - lastElemento (): devuelve el último elemento del vector. - capacity (): devuelve la capacidad del vector. - setSize (int): elige un nuevo tamaño para el vector. En el caso de que sea más grande que el que tenía en un principio, inicializa a null los nuevos valores. En el caso de que sea menor que el inicial, elimina los elementos restantes.	
---SET INTERFACES	
HashSet	Es rápida en cuanto a operaciones básicas (inserción, borrado y búsqueda), no admite duplicados, la iteración a través de sus elementos es más costosa, ya que, necesitará recorrer todas las entradas de la tabla y la ordenación puede diferir del orden en el que se han insertado los elementos.
//Instancia de tipo Genérico (Cuando no sabemos qué tipo de datos vamos a introducir en la colección): HashSet nombre = new HashSet (); //Instancia de colección con tipo específico: HashSet <Tipo_de_dato> nombre = new HashSet <Tipo_de_dato> ();	

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Colecciones/Collections (cont)	
- isEmpty(): devuelve si el conjunto está vacío o contiene valores con un valor booleano. - clone(): devuelve una copia superficial de esta instancia de HashSet: los elementos en sí no están clonados. - clear(): borra todos los elementos del conjunto.	
Recorrer	for(Double s : colSetInt){ System.out.println(s); }
TreeSet	el coste para realizar las operaciones básicas será logarítmico con el número de elementos que tenga el conjunto.
	CÓDIGO: //Instancia de tipo Genérico (Cuando no sabemos qué tipo de datos vamos a introducir en la colección): TreeSet nombre = new TreeSet (); //Instancia de colección con tipo específico: TreeSet <Tipo_de_dato> nombre = new TreeSet <Tipo_de_dato> (); Algunos de sus métodos más importantes son: - first(): devuelve el primer elemento actual (el más bajo) en este conjunto. - last(): devuelve el último elemento actual (el más alto) en este conjunto. - floor(): devuelve el elemento más grande en este conjunto, menor o igual que el elemento dado, o nulo si no hay dicho elemento. - tailSet(): devuelve una vista de la parte de este conjunto, cuyos elementos son mayores que o iguales al elemento pasado por parámetro.
---QUEUE INTERFACES	
ArrayDeque	
- push(): añade un elemento al principio de la cola. - pop(): elimina el elemento de la cola que se ha insertado primero. - pollFirst(): elimina el primer elemento de la cola. - pollLast(): elimina el ultimo elemento de la cola.FMAP	
---MAP	
Un map es un conjunto de valores, con el detalle de que cada uno de estos valores tiene un objeto extra asociado	
A los primeros se les llama claves o keys, porque nos permiten acceder a los segundos. Los valores clave no aceptan valores duplicados.	
get (Object): obtiene el valor correspondiente a una clave. Devuelve null si la clave no existe en el map. put (clave, valor): añade un par clave--valor al map. Si ya había un valor para esa clave lo remplaza. keySet(): devuelve todas las claves (devuelve un Set, es decir, sin duplicados). values(): todos los valores (en este caso sí pueden estar duplicados).entrySet(): todos los pares clave-valor (devuelve un conjunto de objetos Map.Entry, cada uno de los cuales devuelve la clave y el valor con los métodos getKey() y getValue() respectivamente)	
add key and value	coloresPreferidos.put(ILERNA, Color.AZUL);
change key value	coloresPreferidos.put(ILERNA, Color.AZUL);
get	Color colorPreferido = coloresPreferidos.get(ILERNA);
remove	ColoresPreferidos.remove(ILERNA);
HashMap	

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Colecciones/Collections (cont)

Es rápida en cuanto a operaciones básicas (inserción, borrado y búsqueda), no admite duplicados, la iteración a través de sus elementos es más costosa, ya que necesitará recorrer todas las entradas de la tabla y la ordenación puede diferir del orden en el que se han insertado los elementos

//Instancia de tipo Genérico (Cuando no sabemos qué tipo de datos vamos a introducir en la colección): HashMap nombre = new HashMap (); //Instancia de colección con tipo específico: HashMap <Tipo_de_dato_clave, Tipo_de_dato_valor> nombre; nombre = new HashMap < Tipo_de_dato_clave, Tipo_de_dato_valor > ();

https://www.geeksforgeeks.org/collections-in-java-2/

Generic Methods and Classes

es posible que queramos crear alguna clase o método genético de forma que no sepamos previamente el tipo de dato con el que vamos a trabajar. A esto es a lo que llamamos clase y métodos genéricos

Crear clases genéricas

modificador_de_acceso class nombre_clase <T> { T variable; }	Donde “T” representa un tipo de referencia válido en el lenguaje Java: String, Integer, Alumno, Libro, Coche o cualquier otro tipo

Los parámetros de clases por tipos no se limitan a un único parámetro (T). Por ejemplo, la clase HashMap que indicamos a continuación, permite dos parámetros: CÓDIGO: class Hash <A, B> { } En este caso, tenemos dos parámetros A y B que hacen referencia al tipo de clave y el valor

También podemos aplicar esta sintaxis en interfaces: CÓDIGO: public interface nombre_interface<K,V>{ public K getKey(); public V getValue(); }

Y esta interface podría ser implementada por una clase: CÓDIGO: public class n_clase<K,V> implements n_interface<K,V>{ private K key; private V value; public n_clase(K key, V value){ this.key = key; this.value = value; } public K getKey(){return key;} public K getValue(){return value;} } Donde K representa la clave y V representa el valor.

Podemos crear métodos para que puedan utilizar los tipos parametrizados, bien sea en las clases genéricas o en las normales. La sintaxis para crear un tipo genérico es: CÓDIGO: public static <T> T metodogenerico (T parametroFormal) { //código método } Y, para invocar este método: CÓDIGO: claseDelMetodoGenerico.<TipoConcreto> método (ParametroReal); Aunque, en algunos casos, puede que el compilador deduzca qué tipo de parámetro se va a utilizar y, en este caso, se puede obviar. CÓDIGO: ClasedelMetodoGenerico.metodo (parametroReal);

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com