Java – Chuleta Temas 1 - 7

String	
int length()	Devuelve tamaño de la cadena
String toUpperCase()	Convertir a mayúsculas
String toLowerCase()	Convertir a minúsculas
String concat(*str*)	Concatena *str*
String substring(*i*)	Devuelve la subcadena desde i hasta final
String substring(*i,f*)	Devuelve la subcadena desde i hasta f-1
boolean equals (*str*)	Compara con cadena *str*
int compareTo(*str*)	0 si iguales,<0 si menor que *str*, >0 si mayor que *str*
int indexOf(*ch*)	Posición donde ch (-1 no encnt) (tb.ch=String)
int indexOf(*ch,indx*)	Posición donde ch a partir de posición *indx*(-1 no encnt)
char charAt(*indx*)	Devuelve e carácter en posición *indx*.
String replace(*ch1,ch2*)	Reemplaza todos los *ch1* por *ch2 (tb. ch1,ch2 String)*
String replaceAll (*regx,str*)	Reemplaza expresión regular por texto.
String valueOf(*num*)	(static) Convierte a String un tipo primitivo: int, double...
String toString (*num,base*)	Convierte a String un número en la base indicada
int Integer.parseInt(*str*)	(static) Convierte *str* a entero
boolean Character.isLetter(*c*)	(static) true si el carácter –*c*- es letra (análogo: isDigit, isSpaceChar, isUpperCase, isLowerCase
String String.format(*formato*)	formato "%flag width .prec tipo"
System.out.printf(*formato*)	**flag**: + 0 **width**: min.car, **.prec**:decimales **tipo**:d,f,s,b,c,t

StringBuilder / StringBuffer	
StringBuilder()	Constructor vacío (lo crea con 16 caracteres)
StringBuilder(int *x*)	Constructor (lo crea con x caracteres)
StringBuilder(str)	Constructor inicializado con la cadena pasada
append(*str*)	Añade str al final
int length()	Cantidad de caracteres
reverse()	Invierte el orden de los caracteres del SB.
char charAt (*indx*)	Devuelve el carácter de la posición *indx*
int indexOf(str)	Posición donde str (-1 no encontr) *(Solo str, no char)*
int indexOf(str,indx)	Posición donde str a partir de indx (-1 no encontrado)
String substring(i,f)	Devuelve la subcadena desde i hasta f-1
String toString()	Convierte a cadena
setCharAt(indx,ch)	Cambia el carácter de la posición indx por ch
insert(indx, str)	Inserta *str* a partir de la posición *indx*
delete(i, f)	Borra los caracteres entre *i* y *f-1*
deleteCharAt(indx)	Borra el carácter de la posición *indx*
replace(i,f,str)	Sustituye lo que haya entre *i* y *f-1* por *str*

Enum	
enum Enu { VALOR1 (40), VALOR2 (20); int param; En (int p){ param = p;} }	Enumeración. Los valores pueden tener parámetros.
Enu [ ] values()	Devuelve array con los valores de la enum.
Enu valueIOf(str)	Valor corrrespon al String str
int ordinal()	Número de orden en la enum.

Array	
new *Tipo* [cant]	Constructor array de *Tipo* con *cant* elem.
int length	(atributo) Cantidad elementos del array
void System.arraycopy (arr1, pos1,arr2,pos2,cant)	(static) copia arr1 desde pos1 a arr2 en pos2, la cantidad de elementos.
boolean Arrays.equals(arr1,arr2)	(static) *true* si arrays iguales
args[0],args[1]…args[args.length-1]	Parámetros pasados al programa
Arrays.toString(arr1)	Muestra arr1 entre llaves, con comas

ArrayList	
ArrayList <Tipo> al = new ArrayList <> ();	Constructor vacío Tipo(p.ej:Integer)
List <Tipo> *aL* = Arrays.asList (*array1)*	Crea ArrayList *aL* a partir de array1
int size()	Devuelve tamaño del ArrayList
boolean add(x)	Añade x al final. Devuelve *true*
boolean add (pos, x)	Añade x en la posición *pos*
Tipo get (pos)	Devuelve el elemento en *pos*
Tipo remove(pos)	Elimina elemento situado en *pos*
boolean remove (x)	Elimina la primera vez que encuentra x
void clear()	Vacía el arrayList
Tipo set(pos,x)	Sustituye el elemento en *pos*
boolean contains (x)	Comprueba si contiene a X
int indexOf (x)	Devuelve posición X. Si no existe -1
boolean equals (list)	Devuelve *true* si es igual a otro arrayList
Metódos estáticos de Collections:	*Tipo arrList implementará Comparable*
void sort(arrList1)	Ordena ascendentem *arrList1*
Tipo max(arrList1) *//tb.min*	Devuelve máximo/min (según compareTo)
void reverse(arrList1)	Ordena descendentem *arrList1*
void shuffle(arrList1)	Desordena aleatoriam. *arrList1*
int frequency(arrList1, *obj*)	Ocurrencias de *obj* (usa: *equals*)
int binarySearch(arrList1ordenado,obj)	Posición de *obj*. -1 no encontr (usa: equals)
Tipo [ ] arr=aL.toArray(new *Tipo[aL.size()]*)	Crea array con el contenido del arraylist *aL*

Clases útiles	Math, Random y LocalDate/LocalDateTime
num Math.abs(n) *num* Math.pow(base,exp) *num* Math.sqrt (n) *long* Math.round (n) *double* Math.round (n*100)/100d *num* Math.min(x,y) Math.max(x,y) *double* Math.random()	(static) Devuelve valor absoluto de *n* (static) Devuelve *base* elevado a *exp* (static) Devuelve raíz cuadrada de *n* (static) Redondea *n* sin decimales (static) Redondea *n* a 2 decimales (static) Devuelve mínimo y máximo de los dos (static) Devuelve núm. aleatorio entre 0 y <1

Random r = new Random();
int r.nextInt(límite);
int r.nextInt(limInferior, limSuperior);
float r.nextFloat();
	Crea instancia de Random (num.aleatorios)
	Devuelve entero aleatorio >=0 y < límite
	Devuelve entero aleat. >=limInfer y < límiteSuperior
	Devuelve decimal aleatorio >=0 y <1

LocalDate LocalDate.now() *LocalDate* LocalDate.of(a,m,d) *LocalDate* LocalDate.parse(str) *LocalDate* LocalDate.parse(str,form) *LocalDate* fec1.with(ajuste)	Devuelve instancia de LocalDate con fecha actual Devuelve instancia de LocalDate con a, m, d Devuelve instancia LocalDate "AAAA-MM-DD" Devuelve instancia LocalDate a patir de str con formato (ver abajo) Devuelve fecha aplicando ajuste sobre fec1. Ejem. ajuste:*TemporalAdjusters.lastDayOfMonth()*
LocalDateTime LocalDateTime.parse(str)	Devuelve hora "AAAA-MM-DDThh:mm"
int getYear(), getMonthValue() *int* getDayOfWeek().getValue() *int* lengthOfMonth()	Devuelve año, mes, etc Devuelve dia semana 1:lunes… 7 domingo Devuelve cantidad días del mes
boolean isBefore(*fec*), isAfter(*fec*) isEqual(fec), isLeapYear()	*True* si es antes que *fec*, si está después, si es igual, si es año bisiesto
LocalDate plus (*cant*, *unidades*) LocalDate minus (cant, *unidades*)	Suma/resta la cantidad de unidades. *Unidades*:ChronoUnit.HOURS, DAYS, YEARS…
long until (fec, *unidades*) *long* *Unidades*.between(f1, f2)	Devuelve cantidad de unidades hasta *fec* Devuelve cant. de unidades f2 menos f1
DateTimeFormatter f = DateTimeFormatter.ofPattern("dd/MM/yyyy:HH:mm:ss"); System.out.print(f.format(fec))	
Locale loc=Locale.of("gl", "ES") ds.getDisplayName(TextStyle.FULL, loc));	Define idioma para textos, p.ej: día de la semana Devuelve p.ej "luns" siendo ds un *DayOfWeek*

Java – Chuleta Temas 8 - 14

Orientación a objetos	
class claseH extends claseP {	Definición de claseH que es hija de claseP
super()	Llamar constructor padre (en 1ªlínea constr. hija)
super.metodo()	Llamar método padre desde su redefinición en hijo.
getClass().getSimpleName()	Nombre de la clase a la que pertenece
x instanceof *clase1*	*true* si x es una instancia de *clase1* o superior
equals(x)	*true* si la instancia es igual a *x*
abstract class clase1 { }	definición de clase abstracta
interface nomInterf {. . .	definición de interface (pública)
clase1 implements nomInterf {…	definición de clase que implementa interface
Niveles de acceso:	**Quién puede accede a ella?**
private	Solo la propia clase
default	Todas las clases de su paquete
protected	Todas las clases del paquete y clases hijas
public	Todas las clases del proyecto

Excepciones	
try {/*Código */ } catch(Excepcion ex) { ex.printStackTrace(); }	Código que lanza excepciones.

Swing	
setSize(w,h)	Tamaño (ancho, alto)
setTitle(str), getTitle();	Fijar/Obtener título
setVisible(bool)	Elemento visible: sí,no.
setDefaultCloseOperation(const)	Operación al cerrar la ventana
setLocationRelative (null)	Ventana centrada en pantalla
isSelected()	
setEnable(boolean)	
setText(str), getText()	
setName(str), getName()	
setBounds (x,y,w,h)	
setLocation(x,y), setSize(w,h)	
List1.addItem()	Devuelve el carácter de la posición *indx*
List1.getItemAt()	Posición donde str (-1 no encontr) *(Solo str, no char)*
List1.getSelectedIndex()	Posición donde str a partir de indx (-1 no encontrado)
List1.getItemAt (List1.getSelectedIndex())	Devuelve la subcadena desde i hasta f-1
List1.setModel (DefaultListModel m)	Convierte a cadena
m.add….	
JOptionPane.showMessageDialog(	
JOptionPane.showConfirmDialog(	JOptionPane.OK_OPTION
JOptionPane.showInputDialog(	Devuelve String
Evento al seleccionar un elemento	
jButton1.addActionListener(**new** java.awt.event.ActionListener**(){** public void actionPerformed**(**java.awt.event.ActionEvent evt**) {** TareaAlPulsarBoton**(**evt**);} });** private void TareaAlPulsarBoton **(**java.awt.event.ActionEvent evt**)** { /*código*/ }	

Ficheros

Lectura secuencial de fichero de texto:

```
Path ruta = Path.of("ruta"+ File.separator +archivo.txt"); String linea;
try(BufferedReader br =
                    Files.newBufferedReader(ruta,StandardCharsets.UTF_8)) {
        while ((linea = br.readLine()) != null) {
          //tratamiento de la línea
        }
} catch (IOException e) { . . .}
```

Lectura rápida de fichero pequeño:

```
try{ List<String> lineas =
      Files.readAllLines(Path.of("ruta\\archivo.txt"),StandardCharsets.UTF_8);}
catch (IOException ex) { . . .}
```

Escritura secuencial de fichero de texto:

```
try (BufferedWriter bw = Files.newBufferedWriter(Path.of("ruta\\archivo.txt"),
                                  StandardCharsets.UTF_8, StandardOpenOption.CREATE)) {
    bfw.write("texto"); bfw.newLine();  //puede ser bucle de escritura
 } catch (IOException ex) {   System.err.printf("Error:%s",ex.getMessage());}
```

Línea ficheros csv: String[] partes = linea.split(";");

Serializar

La clase debe implementar Serializable:

Escribir:

```
    try( ObjectOutputStream oos = new ObjectOutputStream(new
                                          FileOutputStream("fichero.dat"))) {
        oos.writeObject(obj); }
catch (IOException ex) {System.err.println("Error:"+ ex.getMessage()); }
```

Leer:

```
  try (ObjectInputStream ois = new ObjectInputStream(new
                                        FileInputStream("fichero.dat"))) {
    while (!eof) {   personas.add ((Persona) ois.readObject());    }
    } catch (EOFException e) { eof = true;
   } catch (IOException | ClassNotFoundException e) {. . .      }
```

Properties

Escribir:

```
Properties config = new Properties();
config.setProperty("user", miUsuario);
try {config.store(new FileOutputStream("fich.props"),"cabecera.");}
catch (IOException ioe) {ioe.printStackTrace();}
```

Leer:

```
Properties config = new Properties();
try { config.load(new FileInputStream("fich.props"));
    usuario = config.getProperty("user");
} catch (IOException ioe) {ioe.printStackTrace();}
```

Java – Chuleta Temas 15-19

Colecciones	*definir siempre equals() y hashCode() de la clase*
boolean add(obj) *boolean* remove (obj) *boolean* isEmpty() *void* clear() *int* size() for (Miclase obj : MiColeccion)	Añadir a la colección Eliminar de la colección True si está vacía la colección Vaciar la colección Cantidad de elementos de la colección Recorrer colección (no mapas)
List	
void add(indx, obj) *obj* get (indx) *int* indeoxOf(obj) *obj* remove (indx) *obj* set(indx,obj)	Añadir a la lista en la posición *indx* Devuelve el objeto en la posición *indx* Devuelve la posición del objeto. -1 si no existe Elminar objeto posición *indx* Susituye elemento posición *indx* con *obj*
LinkedList (además de los de List)	
void addFirst(ojb) / addLast(obj) *obj* getFirst, getLast() *obj* removeFirst() / removeLast()	Añade en la primera / última posición Devuelve objeto en primera / última posición Elimina objeto en primera / última posición
Map	

objVal get (objKey)
objVal put (objKey,objVal)
objKey remove (objKey)
Set keyset()
boolean containsKey(objKey)
objVal getOrDefault(clave, defaultVal)
for (String k:mapa.keySet())
mapa.get(k)
for (Persona p. mapa.values())
System.out.println (p);
	Devuelve el valor con clave objKey
	Sustituye/Añade par clave,valor
	Elimina objeto con clave *objKey*
	Devuelve el conjunto de claves,valor
	True si el mapa contiene esa clave
	Get con valor por defecto si no existe en el mapa
	Recorrer claves del mapa
	y acceder a sus valores
	Recorrer los valores del mapa y se
	accede a ellos directamente

TreeMap (además de los de Map) firstKey(), firsEntry() lastKey(), lastEntry()	La clase clave necesita compareTo() Obtener primera clave, primer par clave/valor Obtener última clave, último par clave/valor

Ordenar colecciones (por defecto)
Collection.sort(miColec)
Comparable
class Elem implements Comparable {
int compareTo (Object o) {}
//devuelve <0 , 0, >0
	Ordenar colecciones
	Collection.sort(mColec, instancComparador)
	Comparator
	class Comparador implements Comparator {
	int compare (Object o1, Object o2) {
	//devueve <0, 0,>0

Orientación a objetos avanzada	
public **record** Punto(double x, double y) {}	Record. Clase inmutable. Tiene métodos de de acceso: x(), y y()
int Integer.MAX_VALUE, MIN_VALUE, SIZE *String* Integer.toHexString(int i)	(static) Devuelve valor máximo, minimo y tamaño del tipo wrapper. Devuelve i pasado a hexadecimal (cadena)
regexp = "\\d{4}\\-\\d{2}\\-\\d{2}"; *boolean* cadena.matches(regexp) *boolean* Pattern.matches(regexp, txt) Pattern p= Pattern.compile(regexp); Matcher matcher = p.matcher(txt); matcher.find(); (matcher.group(1));	Definir expresión regular Forma simple de comprobar expr. Reg. Comprueba si txt cumple la expr.regular Compilar la expresión regular. Comprobar si la cumple y encontrar partes (en la expr.reg. cada parte entre paréntesis)

public class Cuadr <T extends Number> {
private T lado;
Cuadr (T lado) { this.lado = lado;}
}
	Clase genérica con un atributo genérico.
	El tipo genérico debe ser descendiente de Number.

Collections.sort(lista,
new Comparator<Pers>(){
@Override
public int compare(Pers p1, Pers p2) {
return p1.nom.compare(p2.nom);
} }
);
	Clase anónima (segundo param. de *sort)*
	Crea instancia de clase hija de Comparator sobreescribiendo compare.

Iterator <Integer> iterator = *lista1*.iterator();
while(iterator.hasNext())
if (*condición*) iterator.remove());
	Borrar elemento de colección *lista1* con iterador, según *condicion.*

Conversiones entre colecciones: List <Obj> lista = Arrays.asList(arr); Set conjunto = new HashSet (lista); ArrayList<Tipo> keyList = new ArrayList<Tipo>(mapa.keySet()); ArrayList<Tipo> valList = new ArrayList<Tipo>(mapa.values());	

Tratamiento XML

Leer archivo XML a árbol DOM:

```
File file = new File("archivo.xml");
      try (FileInputStream fis = new FileInputStream(file);
           InputStreamReader isr = new InputStreamReader(fis, "UTF-8") )   {
          DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
          DocumentBuilder dB = factory.newDocumentBuilder();
          Document doc = dB.parse(new InputSource(isr));
      } catch(Exception e) {    e.printStackTrace();       }
```

Escribir árbol DOM a archivo:

```
File ficheroSalida = new File("archivo2.xml");
TransformerFactory tFactory = TransformerFactory.newInstance();
Transformer transformer = tFactory.newTransformer();
transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
transformer.transform(new DOMSource(doc), new StreamResult(ficheroSalida));
```

Operaciones:	
Element raiz = doc.getDocumentElement();	Leer el elemento raíz
NodeList lista = doc.getElementsByTagName("etiq"); for(int i = 0; i < lista.getLength() ; i++) { *Element* elem = (Element) lista.item(i); *String* elem.getElementsByTagName("etiqhija1"). item(0).getTextContent(); }	Creamos la lista de elementos Recorremos la lista Cast a *Element* de cada elemento Acceder al contenido de las etiquetas del elemento.
String at=elem.getAttribute("*atributo*")	Leer Atributo de un nodo
Boolean elem.hasAttribute("*atributo*")	Existe el atributo en el nodo
elem.setTextContent(*"texto"*);	Modificar texto del nodo
elem.setAttribute("atributo","valor");	Modificar atributo

elem = doc.createElement("nuevaEtiqueta");
elem.appendChild(doc.createTextNode("valor"));
elemPadre.appendChild(elem);
	Añadir un nuevo nodo al árbol.
	Se crea el nodo con su texto
	Y se añade desde el padre.

if (elemHijo!=null) elemPadre.removeChild(elemHijo);	Eliminar nodo, si existe

Acceso a BD

Conexión/Desconexión a MySQL:

```
try ( Connection conexion = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/nombreBD", "usuario","contraseña");
      PreparedStatement ps = conexion.prepareStatement(sentenciaSql)) {
      . . .
    } catch (SQLException e) {
        System.out.println("Cód.err.:" + e.getErrorCode() + "\n" +
          "SLQState: " + e.getSQLState() + "\n" +"Mensaje: " + e.getMessage() + "\n");}
```

Consultas y ResultSet:

```
ResultSet rs = ps.executeQuery();
while (rs.next()) {. . .}   // int f= rs.getRow(), Str txt = rs.getString(1), float f=rs.getFloat(2)
                            // LocalDate fec=rs.getDate(3).toLocalDate();
```

SQL update,delete,insert: int cantFilas = ps.executeUpdate();

Parametros: interrogaciones en la sentencia SQL.
ps.setFloat(1,3.14f); ps.setDate(1, java.sql.Date.valueOf(fec));

Actualización por ResultSet:

```
PreparedStatement ps = conexion.prepareStatement(sql,
        ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
```

- rs.updateFloat(2, 100f); /*y luego: */ rs.updateRow();
- rs.deleteRow();
- rs.moveToInsertRow(); rs.updateString(1,"hola"); rs.insertRow(); rs.next();

Programación Funcional

Interfaces Funcionales / método abstr:	
Predicate / *boolean* test (T t)	Evaluar el parámetro
Consumer / *void* accept (T t)	Consume los datos recibidos
Function / *R* apply (T t)	Transformar el objeto
Supplier / *T* get ()	Obtiene objeto
lista.stream().map(x->x*x) .forEach(System.out::println);	Muestra el cuadrado de los elementos de la lista.
Set cuadradosPares = numeros.stream() .filter(x -> x%2==0).map(x->x*x) .collect(Collectors.toSet());	Obtener un Set con los elementos de la lista pares, elevados al cuadrado.
int tot = lista.stream().map(z -> z.getValor()) .mapToInt(Integer::intValue).sum();	Obtiene el acumulado del getter getValor() de toda la lista.