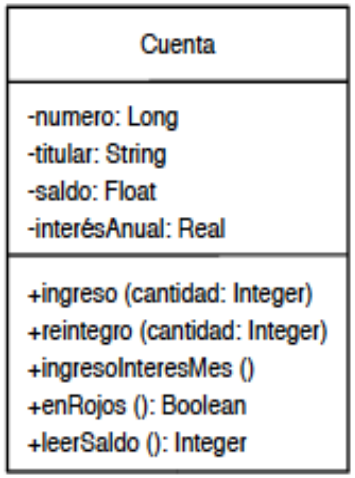
Introducción al lenguaje de programación JAVA

Escuela Superior de Cómputo

Dra. Miriam Pescador Rojas

Definición de una Clase en Java (sintaxis)




```
class Cuenta {
    private long numero;
    private String titular;
    private float saldo;
    private float interesAnual;

    public void ingreso (float cantidad) {
        saldo += cantidad;
    }

    public void reintegro (float cantidad) {
        saldo -= cantidad;
    }

    public void ingresoInteresMes () {
        saldo += interesAnual * saldo / 1200;
    }

    public boolean enRojos () {  return saldo < 0; }
    public float leerSaldo () { return saldo; }
}
```

Constructores en Java

```
class Login {

    protected String username;

    public Login() {
        username = "PPEREZ";
    }

    public Login(String username) {
        this.username = username;
    }

}
```

Getter y setter en Java

```
class Login {

    protected String username;

    public Login() {
    }

    public Login(String username) {
        this.username = username;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

}
```

Métodos en Java

```java
class Login {

    private String username;
    private String password;

    public Login() {
        username = "PPEREZ";
        password = "1234";
    }

    public Login(String username) {
        this.username = username;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public boolean cambiarPass(String pPassword){
        if (validaPass(pPassword)) {
            password = pPassword;
            return true;
        }
        return false;
    }

    private boolean validaPass(String pPassword) {
        return pPassword.length() > 5;
    }

}
```

Instanciar clase en Java

```
*/
public static void main(String[] args) {

    Login loLog = new Login();
    loLog.cambiarPass("56789");

    Login loLog2 = new Login("DGUERRERO");
    loLog2.cambiarPass("34567");
```

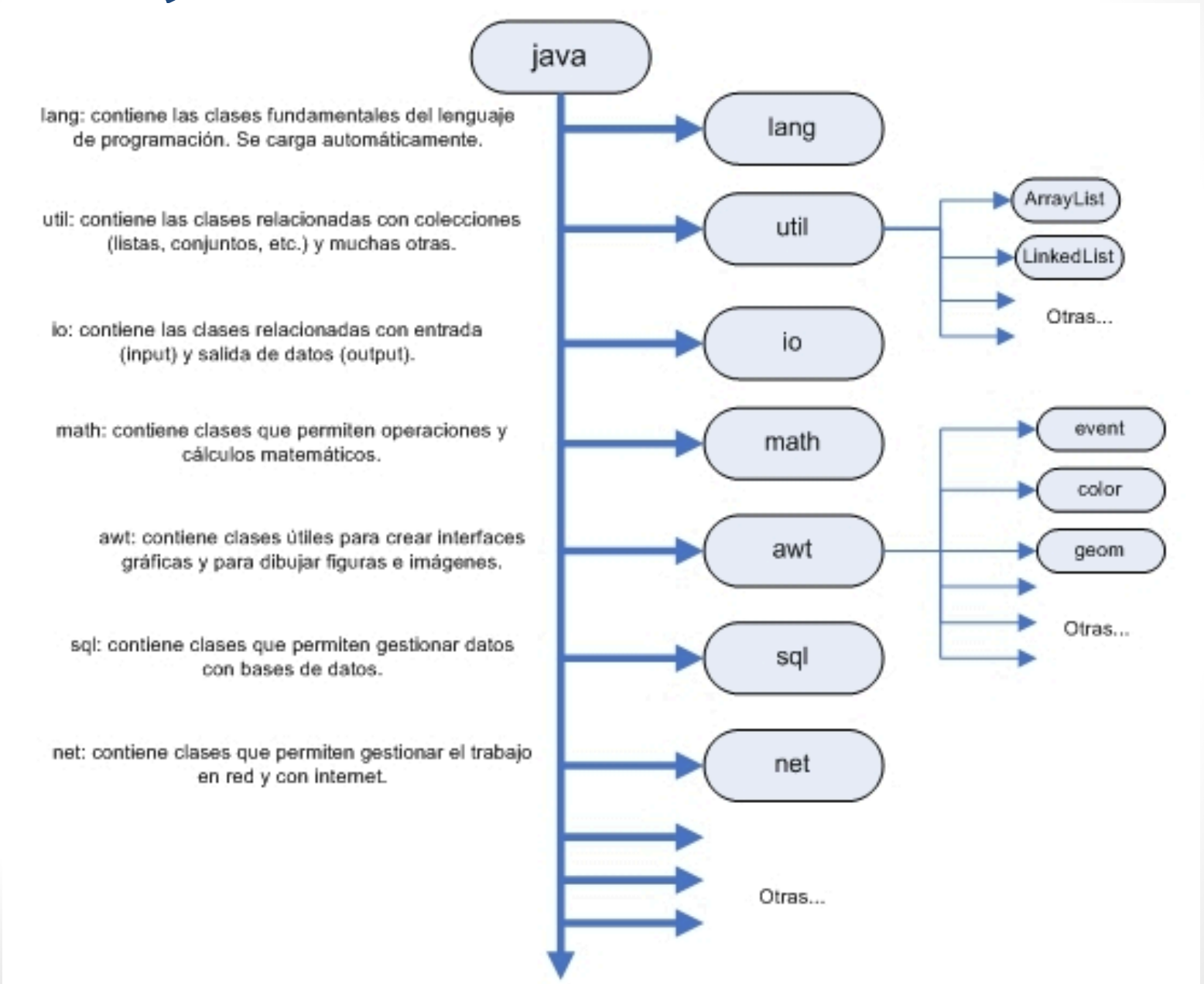
Lectura de Datos consola Java

- Import del paquete java.util.Scanner.
- Crear instancia de la clase Scanner
- El parámetro del constructor será la consola -> System.in
- Para comenzar a leer, utilizar método next, nextLine, nextInt

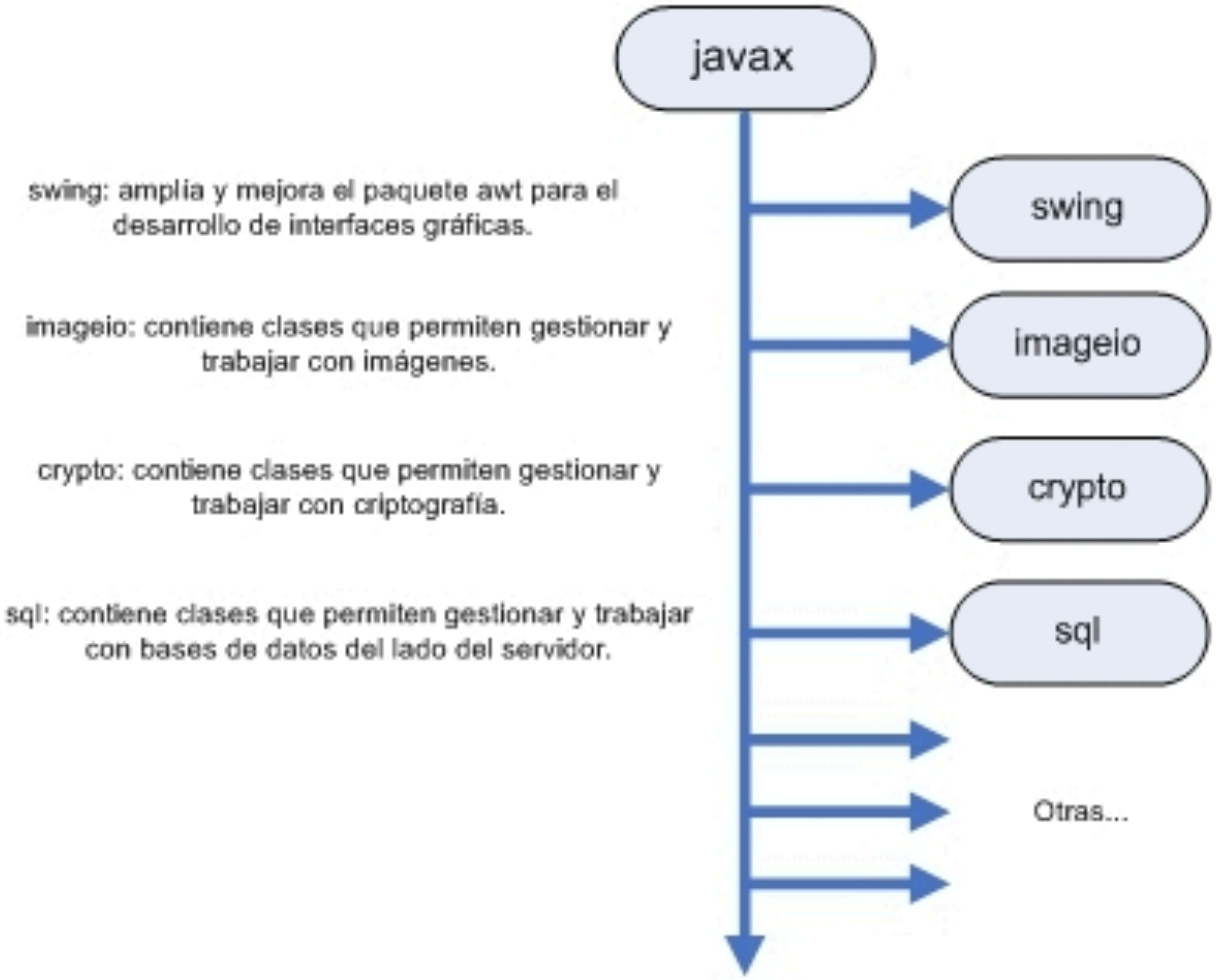
```
public class TestMath {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        String lsEntrada;

        Scanner loLector = new Scanner(System.in);
        lsEntrada = loLector.nextLine();
        System.out.println("texto leído: " + lsEntrada);
        lsEntrada = loLector.next();
        System.out.println("texto leído: " + lsEntrada);

    }

}
```

```
run:
Hola Mundo
texto leído: Hola Mundo
Hola Mundo
texto leído: Hola
BUILD SUCCESSFUL (total time: 12 seconds)
```

API Java


java
lang: contiene las clases fundamentales del lenguaje de programación. Se carga automáticamente.
lang
util: contiene las clases relacionadas con colecciones (listas, conjuntos, etc.) y muchas otras.
util
ArrayList
LinkedList
Otras...
io: contiene las clases relacionadas con entrada (input) y salida de datos (output).
io
math: contiene clases que permiten operaciones y cálculos matemáticos.
math
event
color
awt: contiene clases útiles para crear interfaces gráficas y para dibujar figuras e imágenes.
awt
geom
Otras...
sql: contiene clases que permiten gestionar datos con bases de datos.
sql
net: contiene clases que permiten gestionar el trabajo en red y con internet.
net
Otras...

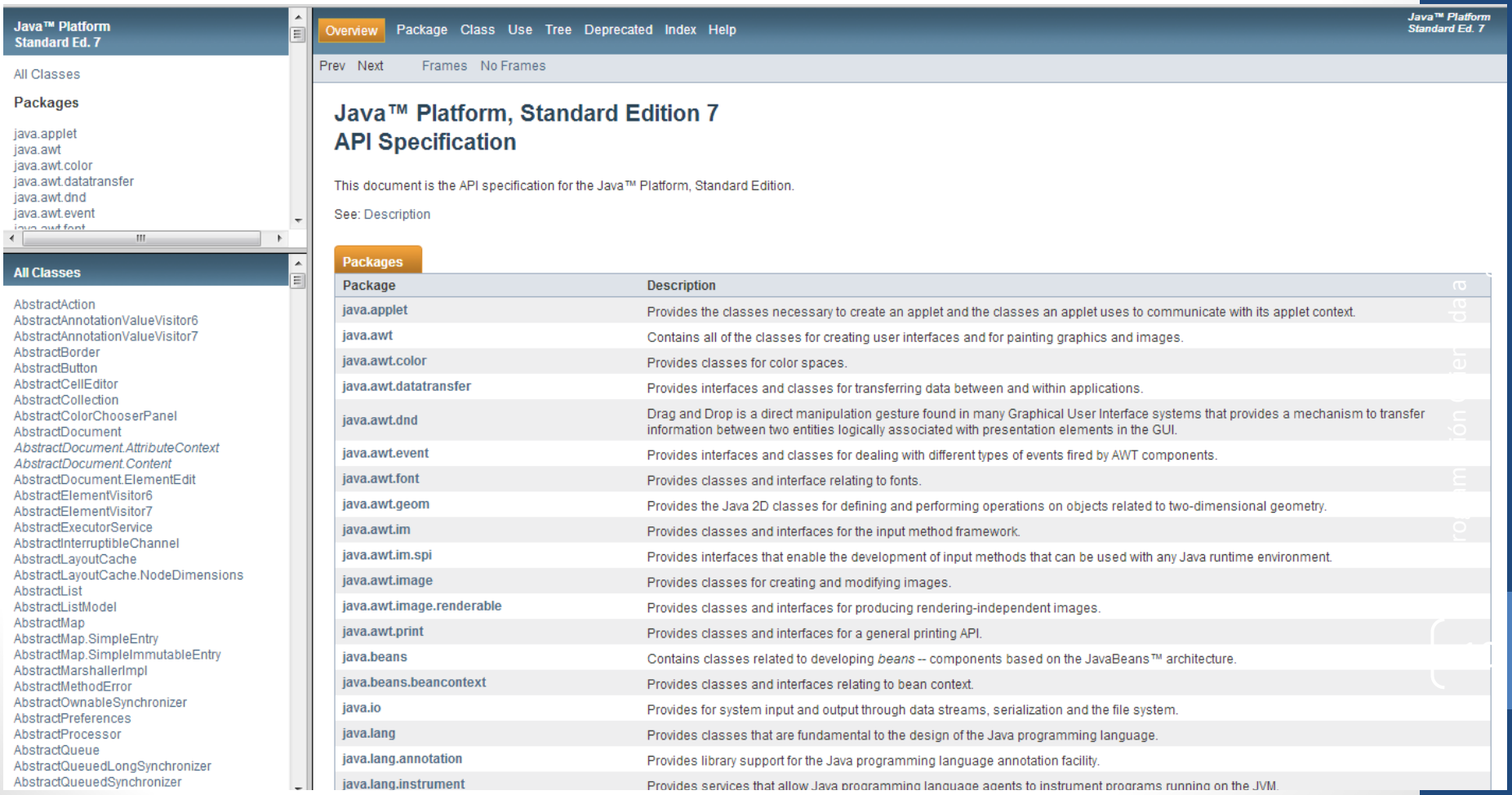
API Java


javax
swing: amplía y mejora el paquete awt para el desarrollo de interfaces gráficas.
imageio: contiene clases que permiten gestionar y trabajar con imágenes.
crypto: contiene clases que permiten gestionar y trabajar con criptografía.
sql: contiene clases que permiten gestionar y trabajar con bases de datos del lado del servidor.
swing
imageio
crypto
sql
Otras...

Documentación de Java

- Ingrese a la siguiente liga para utilizar la documentación de Java

  http://docs.oracle.com/javase/8/docs/api/



Clase String

Constructors

Constructor and Description
`String()` Initializes a newly created `String` object so that it represents an empty character sequence.
`String(byte[] bytes)` Constructs a new `String` by decoding the specified array of bytes using the platform's default charset.
`String(byte[] bytes, Charset charset)` Constructs a new `String` by decoding the specified array of bytes using the specified **charset**.
`String(byte[] ascii, int hibyte)` **Deprecated.** *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the `String` constructors that take a `Charset`, charset name, or that use the platform's default charset.*
`String(byte[] bytes, int offset, int length)` Constructs a new `String` by decoding the specified subarray of bytes using the platform's default charset.
`String(byte[] bytes, int offset, int length, Charset charset)` Constructs a new `String` by decoding the specified subarray of bytes using the specified **charset**.
`String(byte[] ascii, int hibyte, int offset, int count)` **Deprecated.** *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the `String` constructors that take a `Charset`, charset name, or that use the platform's default charset.*
`String(byte[] bytes, int offset, int length, String charsetName)` Constructs a new `String` by decoding the specified subarray of bytes using the specified charset.
`String(byte[] bytes, String charsetName)` Constructs a new `String` by decoding the specified array of bytes using the specified **charset**.
`String(char[] value)` Allocates a new `String` so that it represents the sequence of characters currently contained in the character array argument.
`String(char[] value, int offset, int count)` Allocates a new `String` that contains characters from a subarray of the character array argument.
`String(int[] codePoints, int offset, int count)` Allocates a new `String` that contains characters from a subarray of the **Unicode code point** array argument.
`String(String original)` Initializes a newly created `String` object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.
`String(StringBuffer buffer)` Allocates a new string that contains the sequence of characters currently contained in the string buffer argument.
`String(StringBuilder builder)` Allocates a new string that contains the sequence of characters currently contained in the string builder argument.

Clase String

Methods

Modifier and Type	Method and Description
char	charAt(int index) Returns the char value at the specified index.
int	codePointAt(int index) Returns the character (Unicode code point) at the specified index.
int	codePointBefore(int index) Returns the character (Unicode code point) before the specified index.
int	codePointCount(int beginIndex, int endIndex) Returns the number of Unicode code points in the specified text range of this String.
int	compareTo(String anotherString) Compares two strings lexicographically.
int	compareToIgnoreCase(String str) Compares two strings lexicographically, ignoring case differences.
String	concat(String str) Concatenates the specified string to the end of this string.
boolean	contains(CharSequence s) Returns true if and only if this string contains the specified sequence of char values.
boolean	contentEquals(CharSequence cs) Compares this string to the specified CharSequence.
boolean	contentEquals(StringBuffer sb) Compares this string to the specified StringBuffer.
static String	copyValueOf(char[] data) Returns a String that represents the character sequence in the array specified.
static String	copyValueOf(char[] data, int offset, int count) Returns a String that represents the character sequence in the array specified.
boolean	endsWith(String suffix) Tests if this string ends with the specified suffix.
boolean	equals(Object anObject) Compares this string to the specified object.
boolean	equalsIgnoreCase(String anotherString) Compares this String to another String, ignoring case considerations.
static String	format(Locale l, String format, Object... args) Returns a formatted string using the specified locale, format string, and arguments.
static String	format(String format, Object... args) Returns a formatted string using the specified format string and arguments.
byte[]	getBytes()

Información de un String

- str.length()
- str.startsWith()
- str.endsWith()
- str.indexOf();
- str.compareTo()
- str.substring()
- String.valueOf()

Convertir un String en número

- Cuando introducimos caracteres en un control de edición a veces es inevitable que aparezcan espacios ya sea al comienzo o al final. Para eliminar estos espacios tenemos la función miembro *trim*

```
String str=" 12 ";
String str1=str.trim();
```

Para convertir un string en número entero, primero quitamos los espacios en blanco al principio y al final y luego, llamamos a la función miembro estática *parseInt* de la clase *Integer* (clase envolvente que describe los números enteros)

```
String str="   12 ";
int numero=Integer.parseInt(str.trim());
```

Para convertir un string en número decimal (**double**) se requieren dos pasos: convertir el string en un objeto de la clase envolvente *Double*, mediante la función miembro estática *valueOf*, y a continuación convertir el objeto de la clase*Double* en un tipo primitivo **double** mediante la función *doubleValue*

```
String str="12.35 ";
double numero=Double.valueOf(str).doubleValue();
```

Se puede hacer el mismo procedimiento para convertir un string a número entero

```
String str="12"; int numero=Integer.valueOf(str).intValue();
```

Concatenación

- En ocasiones requerimos una secuencia de caracteres concatenados (una secuencia dentro de otra y seguida la segunda de la primera).
- El método public String concat(String str). Concatena la cadena especificada al final de esta cadena. Si la longitud de la cadena del argumento es 0, entonces este objeto String se devuelve.

- El operador + también nos permite concatenar:

```
String cadena = "hola";
Cadena += "mundo"
```

EJERCICIO 1.

- Investigar en el API que funciones realizan cada uno de los siguientes métodos:

  - charAt(int index)
  - replace(char old, char new)
  - split (String r)
  - toUpperCase()
  - toLowerCase()
  - toString()

EJERCICIO 2

- Implemente una función que reciba como parámetros una cadena y el carácter a buscar. Retornar como resultado el número de incidencias del carácter

- Ejemplo:

Entrada:

String cadena = “Pepe pecas pica papas con un pico”;

char carácter = ‘p’;

Salida

int numIncidencias; // numIncidencias = 7

EJERCICIO 3

- Implemente una función para generar el RFC (sin homoclave) de una persona. La función recibirá como parámetros: nombre, apellido paterno, apellido materno, fecha de nacimiento.
- Ejemplo:

Entrada:

String nombre = “Daniela”;

String paterno = “Guerrero”;

String materno = “Alvarez”

String fecha = “1984/01/23”;

Salida

string RFC; // GUAD840123

Ejercicio 4

1. Agregar un método a la clase Login de nombre acción que evalúe una letra y ejecute los siguientes métodos.
   C – Cambiar Password
   A – Autenticar Password
   R – Revisar que dentro del password no esté escrito el nombre de usuario

2. Agregar el método AutenticaPassword que devuelva un booleano que compare un texto ingresado con el password de la instancia.

3. Agregar el método RevisaPassword que devuelva un booleano que valide que el password no tenga el nombre de usuario.

4. Crear una instancia de Login que evalúe las acciones mediante una instancia de Scanner hasta que el usuario ingrese la letra S