PROGRAMADOR JAVA SE8 - INICIAL

UNIDAD 1: ¿QUÉ ES JAVA?

UNIDAD: 1

PRESENTACIÓN: En esta unidad desarrollaremos la historia de Java, como se compone y que características posee.

OBJETIVOS

- Comprender los inicios de este lenguaje de programación.
- Distinguir los componentes de Java y que fin tiene cada uno.
- La historia del lenguaje, su popularidad y beneficios por sobre otros lenguajes de programación.

TEMARIO:

¿Qué es Java? Un poco de historia

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso en la actualidad.

El lenguaje de programación Java fue originalmente desarrollado por James Gosling, de Sun Microsystems (la cual fue adquirida por la compañía Oracle), y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java), que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

La compañía Sun desarrolló la implementación de referencia original para los compiladores de Java, máquinas virtuales y librerías de clases en 1991, y las publicó por primera vez en 1995. A partir de mayo de 2007, en cumplimiento de las especificaciones del Proceso de la Comunidad Java, Sun volvió a licenciar la mayoría de sus tecnologías de Java bajo la Licencia Pública General de GNU. Otros también han desarrollado implementaciones alternas a estas tecnologías de Sun, tales como el Compilador de Java de GNU y el GNU Classpath.

El lenguaje Java se creó con cinco objetivos principales:

- Debería usar el paradigma de la programación orientada a objetos.
    - Este objetivo se desarrollará en la unidad 3 de este programa.
- Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.

    - Ver sección JVM más abajo
- Debería incluir por defecto soporte para trabajo en red.
- Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
- Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

JDK

Java Development Kit (**JDK**) es un software que **provee herramientas de desarrollo** para la creación de programas en Java. Puede instalarse en una computadora local o en una unidad de red.

Básicamente, el JDK consiste de:

- el compilador Java, javac
- el intérprete Java, java
- un visualizador de applets, appletviewer
- el debugger Java
- el generador de documentación, javadoc
- la herramienta jar, para empaquetar o manejar clases en un solo archivo.

En los sistemas operativos Microsoft Windows sus variables de entorno son:

- JAVAPATH: es una ruta completa del directorio donde está instalado JDK.
- CLASSPATH: son las bibliotecas o clases de usuario.
- PATH: variable donde se agrega la ubicación de JDK.

La JDK incluye también un ambiente de ejecución Java (**JRE**) y documentación de las API que incluye entre otras cosas.

JRE

Java Runtime Environment (JRE) es un paquete de software que contiene todo lo requerido para ejecutar un programa Java. Incluye una implementación de la Java Virtual Machine (**JVM**) junto con una implementación de la Java Class Library. Oracle Corporation, quién posee la marca Java, distribuye las JRE junto con su JVM llamada HotSpot

JVM

Una máquina virtual Java (en inglés Java Virtual Machine, JVM) es una máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el bytecode Java), el cual es generado por el compilador del lenguaje Java.

La JVM básicamente se sitúa en un nivel superior al hardware del sistema sobre el que se pretende ejecutar la aplicación, y este actúa como un puente que entiende tanto el bytecode como el sistema sobre el que se pretende ejecutar. Así, cuando se escribe una aplicación Java, se hace pensando que será ejecutada en una máquina virtual Java en concreto, siendo ésta la que en última instancia convierte de código bytecode a código nativo del dispositivo final.

La gran ventaja de la máquina virtual Java es aportar portabilidad al lenguaje, de manera que desde Sun Microsystems se han creado diferentes máquinas virtuales java para diferentes arquitecturas, y, así, un programa .class escrito en Windows puede ser interpretado en un entorno Linux. Tan solo es

necesario disponer de dicha máquina virtual para dichos entornos. De ahí el famoso axioma que sigue a Java: "escríbelo una vez, ejecútalo en cualquier parte", o "Write once, run anywhere".

Ahora todo junto

En el lenguaje de programación Java, todo el código fuente en texto plano con archivos extensión .java. Los archivos fuente son compilados gracias al compilador javac (parte de la JDK). Los archivos .class no contienen código nativo para ser ejecutado en el procesador; estos contienen bytecodes – el lenguaje máquina que interpreta la máquina virtual Java (JVM). La herramienta java se encarga de ejecutar la aplicación con una instancia de la Java Virtual Machine.

Por el hecho de que la Java VM está disponible en diferentes sistemas operativos, los mismo archivos .class son capaces de ejecutarse tanto en Microsoft Windows como Solaris OS o Mac OS. Existen algunas máquinas virtuales que ejecutan algunas tareas adicionales en pos de mejorar la performance de la aplicación, encontrar cuellos de botella y recompilar a código nativo secciones de uso común en los programas.

PROGRAMADOR JAVA SE8 - INICIAL

UNIDAD 2: Entornos de desarrollo (IDE)

UNIDAD: 2

PRESENTACIÓN: En esta unidad se introduce el concepto de IDE y lo que requiere el mismo para su instalación.

OBJETIVOS

- Que los participantes logren identificar las opciones de IDE, que herramientas incluye y los requerimientos para una primera ejecución de un programa Java.

TEMARIO:

IDE (Entorno de Desarrollo Integrado)

Es una aplicación que proporciona servicios para facilitarle al programador el desarrollo de software.

Consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDE tienen auto-completado inteligente de código (IntelliSense). Algunos IDE contienen un compilador, un intérprete, o ambos, tales como NetBeans y Eclipse.

Muchos IDE también cuentan con un navegador de clases, un buscador de objetos y un diagrama de jerarquía de clases, para su uso con el desarrollo de software orientado a objetos.

Eclipse

Eclipse.org posee permite descargar el IDE para diferentes sistemas operativos y configuraciones del mismo. Se recomienda seguir los pasos que se indican en la página para su instalación. Utilizar el siguiente link para comenzar el proceso: http://wiki.eclipse.org/Eclipse/Installation

Tener en cuenta todos los puntos mencionados en el link, prestar atención al momento de seleccionar la versión del sistema operativo (32 o 64 bits).

PROGRAMADOR JAVA SE8 - INICIAL

UNIDAD 3: Programación Orientada a Objetos vs Programación Estructurada

UNIDAD: 3

PRESENTACIÓN: En esta unidad se desarrollarán los conceptos de programación estructurada y programación orientada a objetos. Se presentará el contraste de una con otra con foco a introducir el paradigma de objetos con Java.

OBJETIVOS

- Que los participantes logren incorporar que conceptos posee la programación estructurada
- Que aprendan los pilares de la programación orientada a objetos
- Que analicen donde y como confluyen ambas vertientes

TEMARIO:

PROGRAMACIÓN ESTRUCTURADA

La programación estructurada es un paradigma de programación orientado a mejorar la claridad, calidad y tiempo de desarrollo de un programa de computadora recurriendo únicamente a subrutinas y tres estructuras básicas: **secuencia**, **selección** (if y switch) e **iteración** (bucles for y while); asimismo, se considera innecesario y contraproducente el uso de la instrucción de transferencia incondicional (GOTO), mucho más difícil de seguir y de mantener, y fuente de numerosos errores de programación.

Entre las ventajas de la programación estructurada sobre el modelo anterior (hoy llamado despectivamente código espagueti), cabe citar las siguientes:

Ventajas de la programación estructurada

- Los programas son fáciles de entender, pueden ser leídos de forma secuencial y no hay necesidad de tener que rastrear saltos de líneas dentro de los bloques de código para intentar entender la lógica interna.
- La estructura de los programas es clara, puesto que las instrucciones están más ligadas o relacionadas entre sí.
- Se optimiza el esfuerzo en las fases de pruebas y depuración. El seguimiento de los fallos o errores del programa (debugging), y con él su detección y corrección, se facilita enormemente.
- Se reducen los costos de mantenimiento. Análogamente a la depuración, durante la fase de mantenimiento, modificar o extender los programas resulta más fácil.
- Los programas son más sencillos y más rápidos de confeccionar.
- Se incrementa el rendimiento de los programadores.

Secuencia, selección e iteración

PROGRAMACIÓN ORIENTADA A OBJETOS

La programación orientada a objetos (**OOP**) es un paradigma de programación basado en el concepto de "objetos", los cuales pueden contener datos, en forma de *campos*, a menudo conocidos como atributos; y código, en forma de procedimientos, a menudo conocidos como *métodos*.

Una característica de los objetos es que los procedimientos de un objeto pueden acceder y, a menudo, modificar los campos de datos del objeto con el que están asociados (los objetos tienen una noción de "this" o "self"). En OOP, los programas de computadora se diseñan modelando objetos que interactúan entre sí.

Existe una diversidad significativa de lenguajes OOP, pero los más populares están basados en *clases*, lo que significa que los objetos son instancias de clases, que generalmente también determinan su *tipo*.

Está basada en varias técnicas, *incluyendo herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento.*

Un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, esto es gracias a tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados *métodos*, que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separa el estado y el comportamiento.

Los **métodos (comportamiento)** y **atributos (estado)** están estrechamente relacionados por la propiedad de conjunto. Esta propiedad destaca que una clase requiere de métodos para poder tratar los atributos con los que cuenta. El programador debe pensar indistintamente en ambos conceptos para no caer en el hábito erróneo de crear clases contenedoras de información por un lado y clases con métodos que manejen a las primeras por el otro. De esta manera se estaría realizando una "programación estructurada camuflada" en un lenguaje de POO.

DIFERENCIAS Y SIMILITUDES

La programación orientada a objetos difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida.

La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada solo se escriben funciones que procesan datos. Los programadores que emplean POO, en cambio, primero definen objetos para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos.

PROGRAMADOR JAVA SE8 - INICIAL

UNIDAD 4: Estructuras de datos, variables, bucles y condicionantes

UNIDAD: 4

PRESENTACIÓN: En esta unidad se desarrollarán algunas estructuras de datos, la declaración de variables, las estructuras de condicionantes y bucles.

OBJETIVOS

- Que los participantes comiencen a comprender que estructuras existen.
- Como se definen las estructuras o como se definen variables.
- Que mecanismos existen para la toma de decisión sobre el flujo y como se realizan iteraciones sobre las estructuras
- Que alcance o ambiente posee una variable.

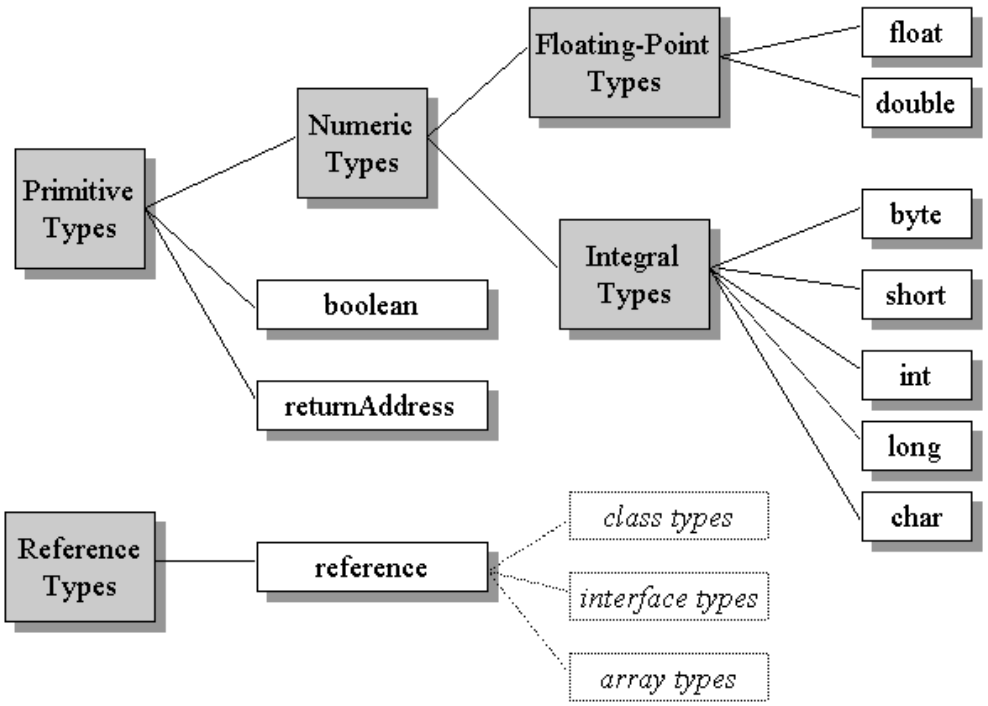
TEMARIO:

ESTRUCTURA DE DATOS

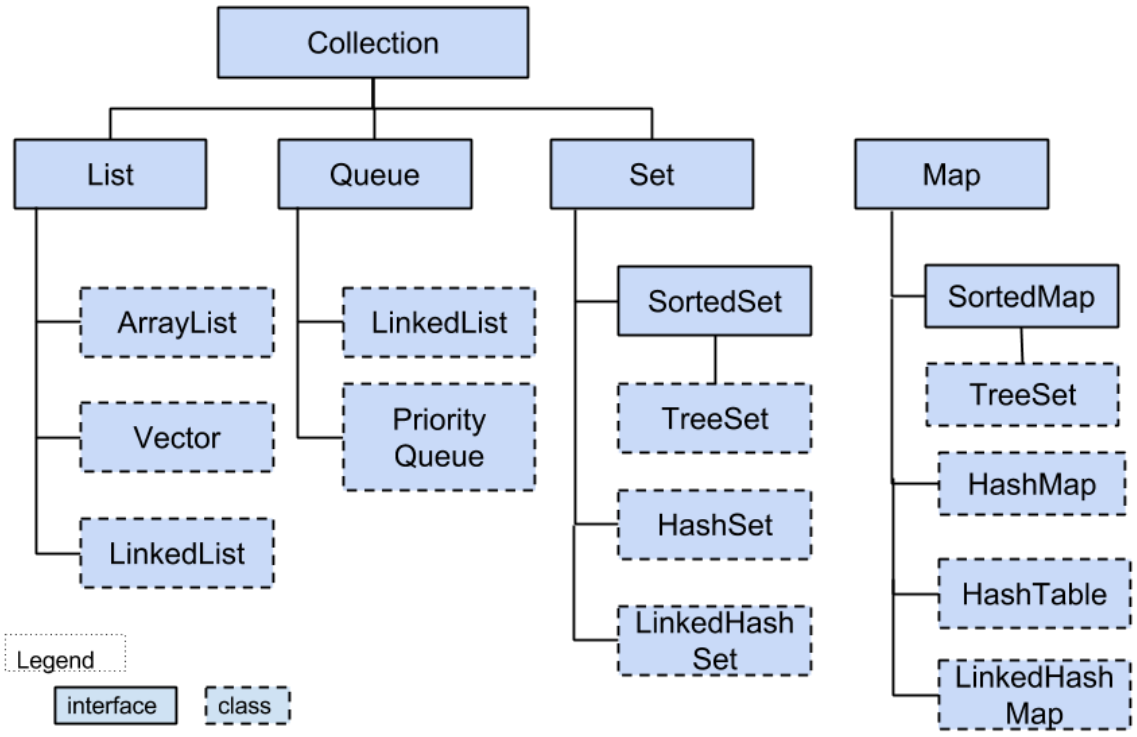
Es una forma particular de organizar datos en una computadora para que puedan ser utilizados de manera eficiente.

Por lo general, las estructuras de datos eficientes son clave para diseñar algoritmos eficientes.

Algunas estructuras de datos o tipos de datos en Java:



En lo que refiere a colecciones:



CONDICIONANTES

Los condicionantes permiten hacer evaluar una condición para hacer variar el flujo del programa en un sentido u otro.

```
if (condición 1) {
```

Este código se ejecutará la condición 1 se cumple

```
} else if (condición 2) {
```

Este código se ejecutará la condición 2 se cumple

```
} else {
```

Este código será ejecutado en caso de no cumplir con la condición anterior

```
}
```

BUCLES

- **FOR:**

  El bucle for cuenta con 3 partes:

  ```
  for(sección 1 o inicial; sección 2 o condición de corte; sección 3 o ejecución tras ejecución) {
  ```

  Esta parte será ejecutada tantas veces se verifique la condición de la sección 2

  ```
  }
  ```

Por ejemplo:

```
for(int i = 0; i < 3; i++){
    System.out.println(i);
}
```

La ejecución paso-a-paso:

1. Se inicializa la variable i en 0.
2. Se evalúa la condición i < 3
3. Se ejecuta la impresión del contenido de la variable i
4. Se ejecuta la instrucción de la sección 3 incrementando el valor de i en 1.
5. Se repiten las operaciones de la 2 a la 4 hasta que la condición de la 2da sección no se cumple para salir del bucle y continuar el programa.

- **WHILE:**

  Existen 2 posibles maneras operar con while

  ```
  while (condición) {
  ```

  Esta parte se ejecutará mientras se cumpla la condición

  ```
  }
  ```

  O

```
do {
```

Esta parte se ejecutará al menos una vez y luego se repetirá si la condición se cumple

```
} while (condición);
```

VARIABLES

En programación, una variable está formada por un espacio en el sistema de almacenaje (memoria principal de un ordenador) y un nombre simbólico (un identificador) que está asociado a dicho espacio. Ese espacio contiene una cantidad de información conocida o desconocida, es decir un valor.

En Java, la declaración de una variable se realiza de la siguiente manera:

```
Tipo_de_dato nombre_de_la_variable;
```

Los tipos de datos primitivos que existen en Java son los siguientes:

Tipo de dato	**Valor por defecto**
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (o algún otro objeto)	null
boolean	false

PROGRAMADOR JAVA SE8 - INICIAL

UNIDAD 5: Funciones y procedimientos

UNIDAD: 5

PRESENTACIÓN: En esta unidad desarrollaremos los conceptos de funciones, parámetros, valores por referencia y firma de un método.

OBJETIVOS

- Que los participantes logren incorporar el concepto de funciones o procedimientos.
- La primera noción del concepto de método, firma y alcance.

TEMARIO:

FUNCIÓN

Una función o subrutina se refiere a un segmento de código separado del bloque principal, el cual puede ser invocado en cualquier momento o como parte de otra subrutina.

En Java la función se llamada método, aquí un ejemplo de una declaración típica de un método:

```
public double calcularRespuesta(double wingSpan, int numberOfEngines,
                                double length, double grossTons) {
    //realizar el cálculo aqui
}
```

Los únicos elementos requiridos en la declaración son el tipo de dato que retorna, un par de parentesis () y el cuerpo del metodo entre llaves {}.

De todos, en general, un metodo esta compuesto por 6 componentes (enumerados en orden):

1. Modificadores —como ser public, private, otros que veremos más adelante.
2. El tipo de retorno— el tipo de dato del valor retornado por el metodo, o void si el metodo no retorna un valor.
3. El nombre del metodo—las reglas para la definicion de los metodos es similar a la definicion de los campos.
4. La lista de parametros entre parentesis— una lista de parametros de entrada precidos por sus tipos de datos. En caso de no tener parametros, debe incluirse los parentesis sin contenido.
5. Una lista de excepciones
6. El cuerpo del metodo, rodeado de llaves {}. Alli se encuentra el codigo, incluyendo la definicion de las variables locales que alli se utilizaran.

PROGRAMADOR JAVA SE8 - INICIAL

UNIDAD 6: Aprendiendo a crear clases y ver cómo es su comportamiento

UNIDAD: 6

PRESENTACIÓN: En esta unidad se desarrollará como se crea una clase y que comportamiento tiene.

OBJETIVOS

- Que los participantes logren identificar en el IDE como crear una clase.
- Que identifiquen la estructura de una clase
- Que identifiquen los componentes de una clase

TEMARIO:

CLASE

Una clase es la definicion de propiedades y comportamiento que tendrá un objeto concreto. La instanciación es tomar esa definición y crear un objeto a partir de ella.

Aquí la estructura de una clase:

`package` <nombre paquete>;

`import` <nombre de las clases adicionales que utilizará la clase>;

`public class` <nombre de la clase, por convención el nombre empieza en mayúscula> {

// Aqui se definirán los atributos de la clase

// Aqui se definirá el constructor de la clase

// Aqui se definirán los metodos de la clase o instancia

}

Por ejemplo:

```
package org.utn.inicial.java;

public class Alumno {


	private String nombre;

	private int edad;

	private long legajo;
```

```
	public Alumno(String nombre, int edad, long legajo) {
		this.nombre = nombre;
		this.edad = edad;
		this.legajo = legajo;
	}
	@Override
	public String toString() {
		return "Alumno [nombre=" + nombre + ", edad=" + edad + ", legajo=" +
legajo + "]";
	}
…..
}
```

En el IDE, Eclipse o cualquier otro de preferencia, uno puede crear la clase indicando sobre que paquete esta residirá. Asi mismo, dentro de la opción source se cuenta con asistentes para crear el constructor de la clase y metodos getters/setters para acceder a los atributos de las clases.

PROGRAMADOR JAVA SE8 - INICIAL

UNIDAD 7: Interacción entre clases

UNIDAD: 7

PRESENTACIÓN: En esta unidad se desarrollará como hacer uso de otras clases dentro de nuestros programas

OBJETIVOS

- Que los participantes conozcan algunas herramientas o clases accesorias para la resolución de los problemas planteados.
- Que aprendan a identicar los metodos que posee una clase
- Que conozcan los recursos disponibles en internet para leer acerca de los metodos que poseen las clases

TEMARIO:

Interacción con datos ingresados por el teclado

Clase java.util.Scanner: clase que permitirá, entre otras cosas, tomar información que ingrese el usuario por teclado.

Modo de uso para tomar datos ingresados por el usuario mediante el teclado:

```
Scanner scan = new Scanner(System.in);

String ingresoUsuario = scan.nextLine();
```

En este caso se instancia la clase Scanner pasandole como parametro la entrada estandar. La segunda linea indica que la variable ingresoUsuario de tipo String tomará el valor que haya ingresado el usuario.

Operaciones trigonométricas y matemáticas

Clase java.lang.Math: En esta clase encontraremos metodos para calculos de raices, potencia y constantes para el numero Pi o E.

Calculo de una potencia:

double potencia = Math.pow(2,4);

En este caso, la clase java.lang,Math posee un metodo **estático** que recibe 2 parametros. El primero es la base de la potencia y el segundo representa el exponente al cual se desea elevar la base.

Formateador de la salida estandar

Java.lang.String no solo es un tipo de dato para cadena de caracteres también posee algunos metodos muy útiles para gestionar su contenido.

Por ejemplo, String tiene un metodo estatico que permite componer una cadena de manera dinamica sin tener que usar o concatenar variables con un mensaje:

```
String.format(“Una cadena %s puede tener datos enteros %d o con punto flotante %f”, cadena, entero, doble);
```

Esta operación tiene como resultante la cadea que alli se muestra entre comillas reemplazando %s con el contenido de la variable cadena, el valor de la variable entero tomará el lugar de %d y doble lo mismo con %f.

Otro uso de la clase String pero esta vez como metodo de instancia es el metodo split. La operación es la siguiente:

```
String[] arrayStrings = unString.split(“,”);
```

como resultado, la operación split devolverá un array de String con todas las cadenas de la variable unString que se encuentren separadas por comas (separador indicado en la llamada al metodo split).