UNIDAD 1: Introducción a la Programación

1.- Introducción.

¿Cuántas acciones de las que has realizado hoy, crees que están relacionadas con la programación? Hagamos un repaso de los primeros instantes del día: te ha despertado la alarma de tu teléfono móvil o radio-despertador, has preparado el desayuno utilizando el microondas, mientras desayunabas has visto u oído las últimas noticias a través de tu receptor de televisión digital terrestre, te has vestido y puede que hayas utilizado el ascensor para bajar al portal y salir a la calle, etc. Quizá no es necesario que continuemos más para darnos cuenta de que casi todo lo que nos rodea, en alguna medida, está relacionado con la programación, los programas y el tratamiento de algún tipo de información.

El volumen de datos que actualmente manejamos y sus innumerables posibilidades de tratamiento constituyen un vasto territorio en el que los programadores tienen mucho que decir.

En esta primera unidad realizaremos un recorrido por los conceptos fundamentales de la programación de aplicaciones. Iniciaremos nuestro camino conociendo con qué vamos a trabajar, qué técnicas podemos emplear y qué es lo que pretendemos conseguir. Continuando con el análisis de las diferentes formas de programación existentes, identificaremos qué fases conforman el desarrollo de un programa, avanzaremos detallando las características relevantes de cada uno de los lenguajes de programación disponibles, para posteriormente, realizar una visión general del lenguaje de programación Java. Finalmente, tendremos la oportunidad de conocer con qué herramientas podríamos desarrollar nuestros programas, escogiendo entre una de ellas para ponernos manos a la obra utilizando el lenguaje Java.

2.- Programas y programación.

2.1.- Buscando una solución.

Generalmente, la primera razón que mueve a una persona hacia el aprendizaje de la programación es utilizar el ordenador como herramienta para resolver problemas concretos. Como en la vida real, la búsqueda y obtención de una solución a un problema determinado, utilizando medios informáticos, se lleva a cabo siguiendo unos pasos fundamentales. En la siguiente tabla podemos ver estas analogías.

Resolución de problemas	
En la vida real...	**En Programación...**
Observación de la situación o problema.	**Análisis del problema:** requiere que el problema sea definido y comprendido claramente para que pueda ser analizado con todo detalle.
Pensamos en una o varias posibles soluciones.	**Diseño o desarrollo de algoritmos:** procedimiento paso a paso para solucionar el problema dado.
Aplicamos la solución que estimamos más adecuada.	**Resolución del algoritmo elegido en la computadora:** consiste en convertir el algoritmo en programa, ejecutarlo y comprobar que soluciona verdaderamente el problema.

¿Qué virtudes debería tener nuestra solución?

- ✓ **Corrección y eficacia:** si resuelve el problema adecuadamente.
- ✓ **Eficiencia:** si lo hace en un tiempo mínimo y con un uso óptimo de los recursos del sistema.

Para conseguirlo, cuando afrontemos la construcción de la solución tendremos que tener en cuenta los siguientes conceptos:

1. **Abstracción:** se trata de realizar un análisis del problema para descomponerlo en problemas más pequeños y de menor complejidad, describiendo cada uno de ellos de manera precisa.
2. **Divide y vencerás,** esta suele ser considerada una filosofía general para resolver problemas y de aquí que su nombre no sólo forme parte del vocabulario informático, sino que también se utiliza en muchos otros ámbitos.
3. **Encapsulación:** consiste en ocultar la información para poder implementarla de diferentes maneras sin que esto influya en el resto de elementos.
4. **Modularidad:** estructuraremos cada parte en módulos independientes, cada uno de ellos tendrá su función correspondiente.

2.2.- Algoritmos y programas.

Después de analizar en detalle el problema a solucionar, hemos de diseñar y desarrollar el algoritmo adecuado. Pero, **¿Qué es un algoritmo?**

Algoritmo: *secuencia ordenada de pasos, descrita sin ambigüedades, que conducen a la solución de un problema dado.*

Los algoritmos son independientes de los lenguajes de programación y de las computadoras donde se ejecutan. Un mismo algoritmo puede ser expresado en diferentes lenguajes de programación y podría ser ejecutado en diferentes dispositivos. Piensa en una receta de cocina, ésta puede ser expresada en castellano, inglés o francés, podría ser cocinada en fogón o vitrocerámica, por un cocinero o más, etc. Pero independientemente de todas estas circunstancias, el plato se preparará siguiendo los mismos pasos.

La diferencia fundamental entre algoritmo y **programa** es que, en el segundo, los pasos que permiten resolver el problema, deben escribirse en un determinado **lenguaje de programación** para que puedan ser ejecutados en el ordenador y así obtener la solución.

Los lenguajes de programación son sólo un medio para expresar el algoritmo y el ordenador un procesador para ejecutarlo. El diseño de los algoritmos será una tarea que necesitará de la creatividad y conocimientos de las técnicas de programación. Estilos distintos, de distintos programadores a la hora de obtener la solución del problema, darán lugar a algoritmos diferentes, igualmente válidos.

En esencia, todo problema se puede describir por medio de un algoritmo y las características fundamentales que éstos deben cumplir son:

- ✓ Debe ser **preciso** e indicar el orden de realización paso a paso.
- ✓ Debe estar **definido**, si se ejecuta dos o más veces, debe obtener el mismo resultado cada vez.
- ✓ Debe ser **finito**, debe tener un número finito de pasos.

Pero cuando los problemas son complejos, es necesario descomponer éstos en subproblemas más simples y, a su vez, en otros más pequeños. Estas estrategias reciben el nombre de **diseño descendente***(Metodología de diseño de programas, consistente en la descomposición del problema en problemas más sencillos de resolver.)* o **diseño modular** (**top-downdesign**)*(Metodología de diseño de programas, que consiste en dividir la solución a un problema en módulos más pequeños o subprogramas. Las soluciones de los módulos se unirán para obtener la solución general del problema).* Este sistema se basa en el lema **divide y vencerás**.

Para representar gráficamente los algoritmos que vamos a diseñar, tenemos a nuestra disposición diferentes herramientas que ayudarán a describir su comportamiento de una forma precisa y genérica, para luego poder codificarlos con el lenguaje que nos interese. Entre otras tenemos:

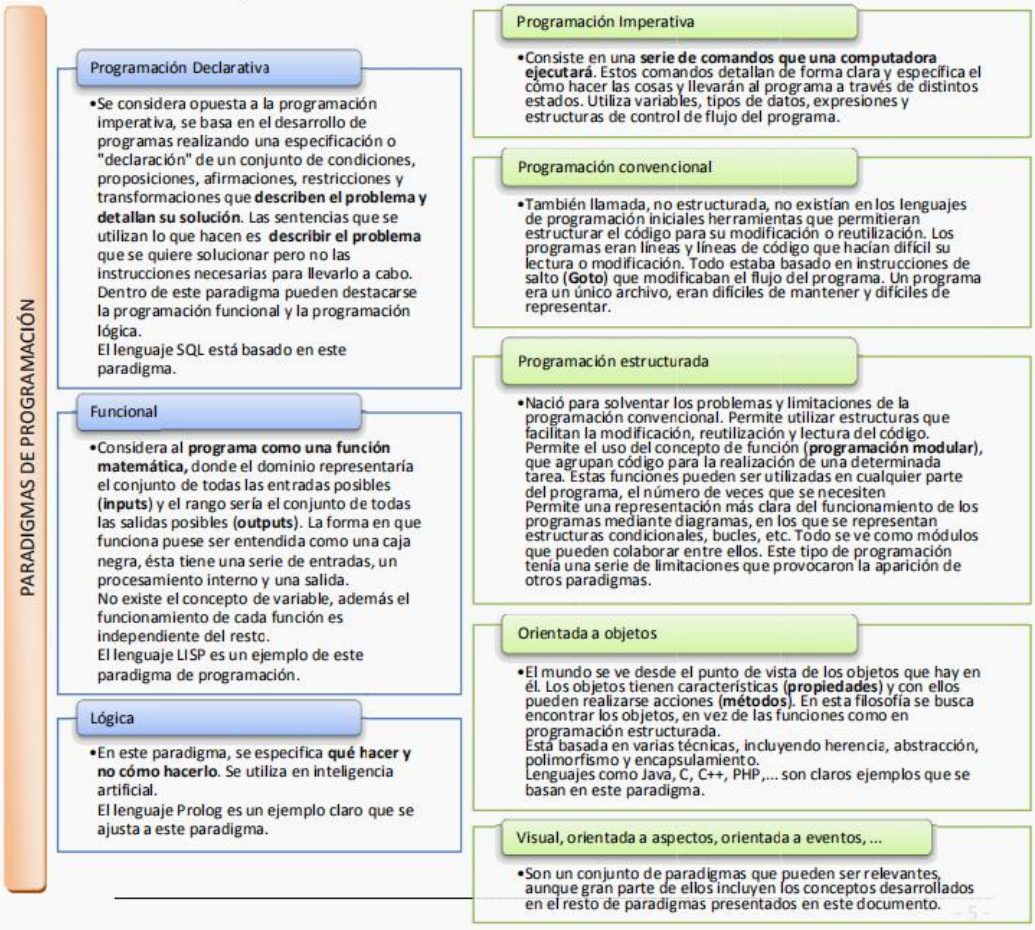
- ✓ **Diagramas de flujo:** Esta técnica utiliza símbolos gráficos para la representación del algoritmo. Suele utilizarse en las fases de análisis.
- ✓ **Pseudocódigo:** Esta técnica se basa en el uso de palabras clave en lenguaje natural, constantes (*Estructura de datos que se utiliza en los lenguajes de programación que no puede cambiar su contenido en el transcurso del programa.*), variables (*Estructura de datos que, como su nombre indica, puede cambiar de contenido a lo largo de la ejecución de un programa.*), otros objetos, instrucciones y estructuras de programación que expresan de forma escrita la solución del problema. Es la técnica más utilizada actualmente.
- ✓ **Tablas de decisión:** En una tabla son representadas las posibles condiciones del problema con sus respectivas acciones. Suele ser una técnica de apoyo al pseudocódigo cuando existen situaciones condicionales complejas.

3.- Paradigmas de Programación

¿Cuántas formas existen de hacer las cosas? Supongo que estarás pensando: varias o incluso, muchas. Pero cuando se establece un patrón para la creación de aplicaciones nos estamos acercando al significado de la palabra paradigma.

Paradigma de programación: *es un modelo básico para el diseño y la implementación de programas. Este modelo determinará como será el proceso de diseño y la estructura final del programa.*

El paradigma representa un enfoque particular o filosofía para la construcción de software. Cada uno tendrá sus ventajas e inconvenientes, será más o menos apropiado, pero no es correcto decir que exista uno mejor que los demás.



4.- Lenguajes de Programación

Como hemos visto, en todo el proceso de resolución de un problema mediante la creación de software, después del análisis del problema y del diseño del algoritmo que pueda resolverlo, es necesario traducir éste a un lenguaje que exprese claramente cada uno de los pasos a seguir para su correcta ejecución. Este lenguaje recibe el nombre de lenguaje de programación.

Lenguaje de programación: *Conjunto de reglas sintácticas y semánticas, símbolos y palabras especiales establecidas para la construcción de programas. Es un lenguaje artificial, una construcción mental del ser humano para expresar programas.*

Gramática del lenguaje: *Reglas aplicables al conjunto de símbolos y palabras especiales del lenguaje de programación para la construcción de sentencias correctas.*

Léxico: *Es el conjunto finito de símbolos y palabras especiales, es el vocabulario del lenguaje.*

Sintaxis: *Son las posibles combinaciones de los símbolos y palabras especiales. Está relacionada con la forma de los programas.*

Semántica: *Es el significado de cada construcción del lenguaje, la acción que se llevará a cabo.*

Hay que tener en cuenta que pueden existir sentencias sintácticamente correctas, pero semánticamente incorrectas. Por ejemplo, *"Un avestruz dio un zarpazo a su cuidador"* está bien construida sintácticamente, pero es evidente que semánticamente no.

Una característica relevante de los lenguajes de programación es, precisamente, que más de un programador pueda usar un conjunto común de instrucciones que sean comprendidas entre ellos. A través de este conjunto se puede lograr la construcción de un programa de forma colaborativa.

Los lenguajes de programación pueden ser clasificados en función de lo cerca que estén del lenguaje humano o del lenguaje de los computadores. El lenguaje de los computadores son códigos binarios, es decir, secuencias de unos y ceros. Detallaremos seguidamente las características principales de los lenguajes de programación.

LENGUAJES MÁQUINA, ENSAMBLADORES Y DE ALTO NIVEL

Los programadores escriben instrucciones en diversos lenguajes de programación, algunos de los cuales los comprende directamente la computadora, mientras que otros requieren pasos intermedios de *traducción*. En la actualidad se utilizan cientos de lenguajes de computación. Éstos se dividen en tres tipos generales:

1. Lenguajes máquina
2. Lenguajes ensambladores
3. Lenguajes de alto nivel

Lenguajes máquina

Cualquier computadora sólo puede entender de manera directa su propio **lenguaje máquina**, el cual se define según su diseño de hardware. Por lo general, los lenguajes máquina consisten en cadenas de números (que finalmente se reducen a unos y ceros) que instruyen a las computadoras para realizar sus operaciones más elementales, una a la vez. Los lenguajes máquina son *dependientes de la máquina* (es decir, un lenguaje máquina en particular puede usarse sólo en un tipo de computadora). Dichos lenguajes son difíciles de comprender para los humanos. Por ejemplo, he aquí una sección de uno de los primeros programas de nómina en lenguaje máquina, el cual suma el pago de las horas extras al sueldo base y almacena el resultado en el sueldo bruto:

```
+1300042774
+1400593419
+1200274027
```

Lenguajes ensambladores y ensambladores

La programación en lenguaje máquina era demasiado lenta y tediosa para la mayoría de los programadores. En vez de utilizar las cadenas de números que las computadoras podían entender de manera directa, los programadores empezaron a utilizar abreviaturas del inglés para representar las operaciones elementales. Estas abreviaturas formaron la base de los **lenguajes ensambladores**. Se desarrollaron *programas traductores* conocidos como **ensambladores** para convertir los primeros programas en lenguaje ensamblador a lenguaje máquina, a la velocidad de la computadora. La siguiente sección de un programa en lenguaje ensamblador también suma el pago de las horas extras al sueldo base y almacena el resultado en el sueldo bruto:

```
load        sueldobase
add         sueldoextra
store       sueldobruto
```

Aunque este código es más claro para los humanos, las computadoras no lo pueden entender sino hasta que se traduce en lenguaje máquina.

Lenguajes de alto nivel y compiladores

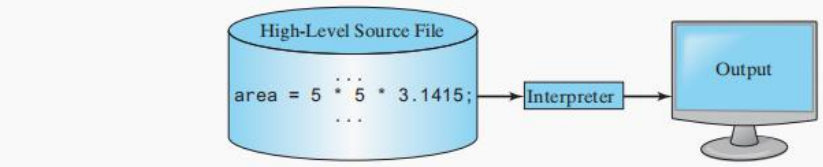
El uso de las computadoras se incrementó rápidamente con la llegada de los lenguajes ensambladores, pero los programadores aún requerían de muchas instrucciones para llevar a cabo incluso hasta las tareas más simples. Para agilizar el proceso de programación se desarrollaron los **lenguajes de alto nivel**, en los que podían escribirse instrucciones individuales para realizar tareas importantes. Los programas traductores, denominados **compiladores**, convierten programas en lenguaje de alto nivel a lenguaje máquina. Los lenguajes de alto nivel permiten a los programadores escribir instrucciones que son muy similares al inglés común, y contienen la notación matemática común. Un programa de nómina escrito en un lenguaje de alto nivel podría contener *una* instrucción como la siguiente:

```
sueldoBruto = sueldoBase + sueldoExtra
```

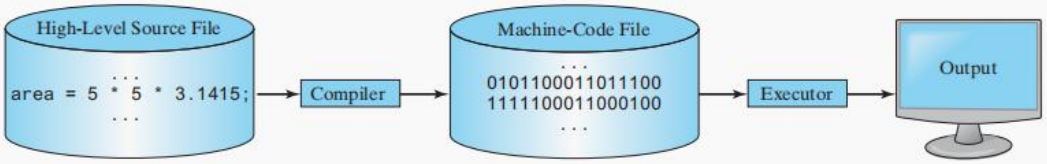
Desde el punto de vista del programador, los lenguajes de alto nivel son mucho más recomendables que los lenguajes máquina o ensamblador. Java es uno de los lenguajes de alto nivel más utilizados.

Intérpretes

El proceso de compilación de un programa extenso escrito en lenguaje de alto nivel a un lenguaje máquina, puede tardar un tiempo considerable en la computadora. Los programas *intérpretes*, que se desarrollaron para ejecutar de manera directa programas en lenguaje de alto nivel, evitan el retraso de la compilación, aunque se ejecutan con más lentitud que los programas compilados. Hablaremos más sobre la forma en que trabajan los intérpretes en la sección 1.9, en donde aprenderá que Java utiliza una astuta mezcla de compilación e interpretación, optimizada con base en el rendimiento, para ejecutar los programas.



(a)

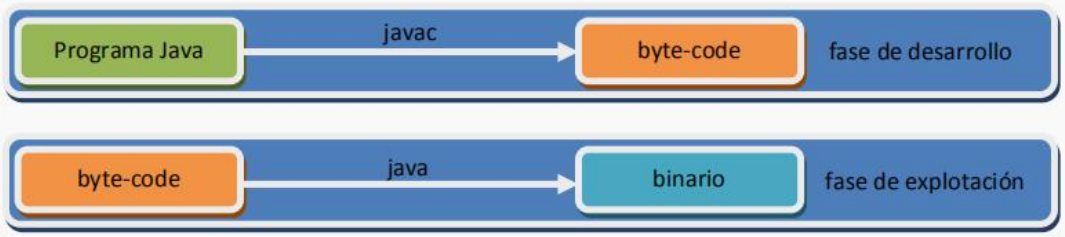


(b)

Debes conocer

Puedes entender por qué Java es un lenguaje compilado e interpretado a través del siguiente esquema.

Los dos procesos se realizan en fases distintas:



Lenguaje de programación	Descripción
Fortran	Fortran (FORmula TRANslator, traductor de fórmulas) fue desarrollado por IBM Corporation a mediados de la década de 1950 para utilizarse en aplicaciones científicas y de ingeniería que requerían cálculos matemáticos complejos. Aún se utiliza mucho y sus versiones más recientes soportan la programación orientada a objetos.
COBOL	COBOL (COmmon Business Oriented Language, lenguaje común orientado a negocios) fue desarrollado a finales de la década de 1950 por fabricantes de computadoras, el gobierno estadounidense y usuarios de computadoras de la industria, con base en un lenguaje desarrollado por Grace Hopper, un oficial de la Marina de Estados Unidos y científico informático, que también abogó por la estandarización internacional de los lenguajes de programación. COBOL aún se utiliza mucho en aplicaciones comerciales que requieren de una manipulación precisa y eficiente de grandes volúmenes de datos. Su versión más reciente soporta la programación orientada a objetos.
Pascal	Las actividades de investigación en la década de 1960 dieron como resultado la *programación estructurada*: un método disciplinado para escribir programas que sean más claros, fáciles de probar y depurar, y más fáciles de modificar que los programas extensos producidos con técnicas anteriores. Un resultado de esta investigación fue el desarrollo del lenguaje de programación Pascal en 1971, el cual se diseñó para la enseñanza de la programación estructurada y fue popular en los cursos universitarios durante varias décadas.
Ada	Ada, un lenguaje basado en Pascal, se desarrolló bajo el patrocinio del Departamento de Defensa (DOD) de Estados Unidos durante la década de 1970 y a principios de la década de 1980. El DOD quería un solo lenguaje que pudiera satisfacer la mayoría de sus necesidades. El nombre de este lenguaje es en honor de Lady Ada Lovelace, hija del poeta Lord Byron. A ella se le atribuye el haber escrito el primer programa para computadoras en el mundo, a principios de la década de 1800 (para la Máquina Analítica, un dispositivo de cómputo mecánico diseñado por Charles Babbage). Ada también soporta la programación orientada a objetos.

Lenguaje de programación	Descripción
Basic	Basic se desarrolló en la década de 1960 en el Dartmouth College, para familiarizar a los principiantes con las técnicas de programación. Muchas de sus versiones más recientes son orientadas a objetos.
C	C fue desarrollado a principios de la década de 1970 por Dennis Ritchie en los Laboratorios Bell. En un principio se hizo muy popular como el lenguaje de desarrollo del sistema operativo UNIX. En la actualidad, la mayoría del código para los sistemas operativos de propósito general se escribe en C o C++.
C++	C++, una extensión de C, fue desarrollado por Bjarne Stroustrup a principios de la década de 1980 en los Laboratorios Bell. C++ proporciona varias características que "pulen" al lenguaje C, pero lo más importante es que proporciona las capacidades de una programación orientada a objetos.
Objective-C	Objective-C es un lenguaje orientado a objetos basado en C. Se desarrolló a principios de la década de 1980 y después fue adquirido por la empresa Next, que a su vez fue adquirida por Apple. Se ha convertido en el lenguaje de programación clave para el sistema operativo OS X y todos los dispositivos operados por el iOS (como los dispositivos iPod, iPhone e iPad).
Visual Basic	El lenguaje Visual Basic de Microsoft se introdujo a principios de la década de 1990 para simplificar el desarrollo de aplicaciones para Microsoft Windows. Sus versiones más recientes soportan la programación orientada a objetos.
Visual C#	Los tres principales lenguajes de programación orientados a objetos de Microsoft son Visual Basic (basado en el Basic original), Visual C++ (basado en C++) y Visual C# (basado en C++ y Java; desarrollado para integrar Internet y Web en las aplicaciones de computadora).

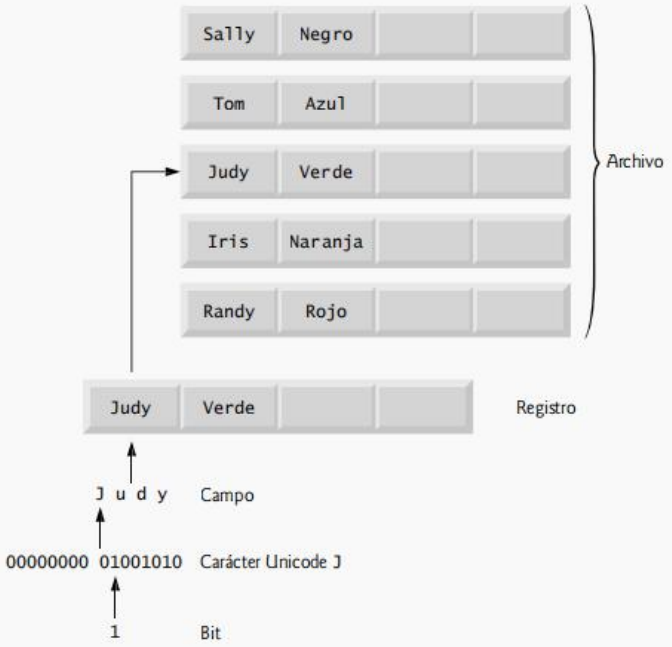
PHP	PHP es un lenguaje orientado a objetos de secuencias de comandos y código fuente abierto, el cual recibe soporte por medio de una comunidad de usuarios y desarrolladores; se utiliza en millones de sitios Web. PHP es independiente de la plataforma —existen implementaciones para todos los principales sistemas operativos UNIX, Linux, Mac y Windows. PHP también soporta muchas bases de datos, incluyendo la popular MySQL de código fuente abierto.
Perl	Perl (Lenguaje práctico de extracción y reporte), uno de los lenguajes de secuencia de comandos orientados a objetos más utilizados para la programación Web, fue desarrollado en 1987 por Larry Wall. Cuenta con capacidades complejas de procesamiento de textos.
Python	Python, otro lenguaje orientado a objetos de secuencias de comandos, se liberó al público en 1991. Fue desarrollado por Guido van Rossum del Instituto Nacional de Investigación para las Matemáticas y Ciencias Computacionales en Amsterdam (CWI); la mayor parte de Python se basa en Modula-3, un lenguaje de programación de sistemas. Python es "extensible", lo que significa que puede extenderse a través de clases e interfaces de programación.
JavaScript	JavaScript es el lenguaje de secuencias de comandos más utilizado en el mundo. Su principal uso es para agregar comportamiento dinámico a las páginas Web; por ejemplo, animaciones e interactividad mejorada con el usuario. Se incluye en todos los principales navegadores Web.

4.1 JERARQUÍA DE DATOS

Los elementos de datos que procesan las computadoras forman una **jerarquía de datos** cuya estructura se vuelve cada vez más grande y compleja, a medida que pasamos de los elementos de datos más simples (conocidos como "bits") a los más complejos, como los caracteres y los campos. La figura 1.3 ilustra una porción de la jerarquía de datos.

Bits

El elemento de datos más pequeño en una computadora puede asumir el valor `0` o el valor `1`. A dicho elemento de datos se le denomina **bit** (abreviación de "dígito binario": un dígito que puede asumir uno de *dos* valores). Es increíble que todas las impresionantes funciones que realizan las computadoras impliquen sólo las manipulaciones más simples de los dígitos `0` y `1`, como *examinar el valor de un bit, establecer el valor de un bit* e *invertir el valor de un bit* (de `1` a `0` o de `0` a `1`).



Caracteres

Es tedioso para las personas trabajar con datos en el formato de bajo nivel de los bits. En cambio, prefieren trabajar con *dígitos decimales* (0-9), *letras* (A-Z y a-z) y *símbolos especiales* (por ejemplo, $, @, %, &, *, (,), –, +, “, :, ? y /). Los dígitos, letras y símbolos especiales se conocen como **caracteres**. El **conjunto de caracteres** de la computadora es el conjunto de todos los caracteres que se utilizan para escribir programas y representar elementos de datos. Las computadoras sólo procesan unos y ceros, por lo que el conjunto de caracteres de una computadora representa a cada carácter como un patrón de unos y ceros. Java usa caracteres **Unicode®** que están compuestos de uno, dos o cuatro bytes (8, 16 o 32 bits). Unicode contiene caracteres para muchos de los idiomas que se usan en el mundo. En el apéndice H obtendrá más información sobre Unicode. En el apéndice B obtendrá más información sobre el conjunto de caracteres **ASCII (Código estándar estadounidense para el intercambio de información)**, que es el popular subconjunto de Unicode que representa las letras mayúsculas y minúsculas, los dígitos y algunos caracteres especiales comunes.

Campos

Así como los caracteres están compuestos de bits, los **campos** están compuestos de caracteres o bytes. Un campo es un grupo de caracteres o bytes que transmiten un significado. Por ejemplo, un campo compuesto de letras mayúsculas y minúsculas se puede usar para representar el nombre de una persona, y un campo compuesto de dígitos decimales podría representar la edad de esa persona.

Registros

Se pueden usar varios campos relacionados para componer un **registro** (el cual se implementa como una clase [`class`] en Java). Por ejemplo, en un sistema de nómina, el registro de un empleado podría consistir en los siguientes campos (los posibles tipos para estos campos se muestran entre paréntesis):

- Número de identificación del empleado (un número entero)
- Nombre (una cadena de caracteres)
- Dirección (una cadena de caracteres)
- Salario por horas (un número con punto decimal)
- Ingresos del año a la fecha (un número con punto decimal)
- Monto de impuestos retenidos (un número con punto decimal)

Por lo tanto, un registro es un grupo de campos relacionados. En el ejemplo anterior, todos los campos pertenecen al *mismo* empleado. Una compañía podría tener muchos empleados y un registro de nómina para cada uno.

Archivos

Un **archivo** es un grupo de registros relacionados. [*Nota:* dicho en forma más general, un archivo contiene datos arbitrarios en formatos arbitrarios. En algunos sistemas operativos, un archivo se ve tan sólo como una *secuencia de byte* y cualquier organización de esos bytes en un archivo, como cuando se organizan los datos en registros, es una vista creada por el programador de la aplicación. En el capítulo 15 verá cómo se hace eso]. Es muy común que una organización tenga muchos archivos, algunos de los cuales pueden contener miles de millones, o incluso billones de caracteres de información.

Base de datos

Una **base de datos** es una colección de datos organizados para facilitar su acceso y manipulación. El modelo más popular es la *base de datos relacional*, en la que los datos se almacenan en simples *tablas*. Una tabla incluye *registros* y *campos*. Por ejemplo, una tabla de estudiantes podría incluir los campos nombre, apellido, especialidad, año, número de identificación (ID) del estudiante y promedio de calificaciones. Los datos para cada estudiante constituyen un registro y las piezas individuales de información en cada registro son los campos. Puede *buscar*, *ordenar* y manipular de otras formas los datos con base en la relación que tienen con otras tablas o bases de datos. Por ejemplo, una universidad podría utilizar datos de la base de datos de los estudiantes en combinación con los de bases de datos de cursos, alojamiento en el campus, planes alimenticios, etc. En el capítulo 24 hablaremos sobre las bases de datos.

Big Data

La cantidad de datos que se produce a nivel mundial es enorme y aumenta con rapidez. De acuerdo con IBM, cada día se generan alrededor de 2.5 trillones (2.5 *exabytes*) de datos y el 90% de los datos en el mundo se crearon ¡tan sólo en los últimos dos años![6] De acuerdo con un estudio de Digital Universe, en 2012 el suministro de datos globales llegó a 2.8 *zettabytes* (lo que equivale a 2.8 billones de gigabytes).[7] La figura 1.4 muestra algunas mediciones comunes de bytes. Las aplicaciones de **Big Data** lidian con dichas cantidades masivas de datos y este campo crece con rapidez, lo que genera muchas oportunidades para los desarrolladores de software. De acuerdo con un estudio por parte de Gartner Group, para 2015 más de 4 millones de empleos de TI a nivel mundial darán soporte a los grandes volúmenes de datos (Big Data).[8]

Unidad	Bytes	Lo que equivale aproximadamente a
1 kilobyte (KB)	1024 bytes	10^3 (1024 bytes exactamente)
1 megabyte (MB)	1024 kilobytes	10^6 (1,000,000 bytes)
1 gigabyte (GB)	1024 megabytes	10^9 (1,000,000,000 bytes)
1 terabyte (TB)	1024 gigabytes	10^{12} (1,000,000,000,000 bytes)
1 petabyte (PB)	1024 terabytes	10^{15} (1,000,000,000,000,000 bytes)
1 exabyte (EB)	1024 petabytes	10^{18} (1,000,000,000,000,000,000 bytes)
1 zettabyte (ZB)	1024 exabytes	10^{21} (1,000,000,000,000,000,000,000 bytes)

JAVA

La contribución más importante a la fecha de la revolución del microprocesador es que hizo posible el desarrollo de las computadoras personales. Los microprocesadores han tenido un profundo impacto en los dispositivos electrónicos inteligentes para uso doméstico. Al reconocer esto, Sun Microsystems patrocinó en 1991 un proyecto interno de investigación corporativa dirigido por James Gosling, que dio como resultado un lenguaje de programación orientado a objetos y basado en C++, al que Sun llamó Java.

Un objetivo clave de Java es poder escribir programas que se ejecuten en una gran variedad de sistemas computacionales y dispositivos controlados por computadora. A esto se le conoce algunas veces como "escribir una vez, ejecutar en cualquier parte".

La popularidad del servicio Web explotó en 1993; en ese entonces Sun vio el potencial de usar Java para agregar *contenido dinámico*, como interactividad y animaciones, a las páginas Web. Java atrajo la atención de la comunidad de negocios debido al fenomenal interés en el servicio Web. En la actualidad, Java se utiliza para desarrollar aplicaciones empresariales a gran escala, para mejorar la funcionalidad de los servidores Web (las computadoras que proporcionan el contenido que vemos en nuestros navegadores Web), para proporcionar aplicaciones para los dispositivos de uso doméstico (como teléfonos celulares, teléfonos inteligentes, receptores de televisión por Internet y mucho más) y para muchos otros propósitos. Java también es el lenguaje clave para desarrollar aplicaciones para teléfonos inteligentes y tabletas de Android. En 2010, Oracle adquirió Sun Microsystems.

Bibliotecas de clases de Java

Usted puede crear cada clase y método que necesite para formar sus programas de Java. Sin embargo, la mayoría de los programadores en Java aprovechan las ricas colecciones de clases y métodos existentes en las **bibliotecas de clases de Java**, que también se conocen como **API (Interfaces de programación de aplicaciones) de Java**.

Tip de rendimiento 1.1

Si utiliza las clases y métodos de las API de Java en vez de escribir sus propias versiones, puede mejorar el rendimiento de sus programas, ya que estas clases y métodos están escritos de manera cuidadosa para funcionar con eficiencia. Esta técnica también reduce el tiempo de desarrollo de los programas.

Dispositivos		
Sistemas de aeroplanos	Cajeros automáticos (ATM)	Sistemas de infoentretenimiento de automóviles
Reproductores Blu-Ray Disc™	Decodificadores de TV por cable	Copiadoras
Tarjetas de crédito	Escáneres para TC	Computadoras de escritorio
Lectores de libros electrónicos	Consolas de juegos	Sistemas de navegación GPS
Electrodomésticos	Sistemas de seguridad para el hogar	Interruptores de luz
Terminales de lotería	Dispositivos médicos	Teléfonos móviles
MRI	Estaciones de pago de estacionamiento	Impresoras
Pases de transporte	Robots	Enrutadores
Tarjetas inteligentes	Medidores inteligentes	Plumas inteligentes
Teléfonos inteligentes	Tabletas	Televisiones
Decodificadores para televisores	Termostatos	Sistemas de diagnóstico vehicular

Fig. 1.1 | Algunos dispositivos que utilizan Java.

Estadísticas de TIOBE de lenguajes de programación:
https://www.tiobe.com/tiobe-index/

Estadísticas REDMONK:
https://redmonk.com/sogrady/2019/03/20/language-rankings-1-19/?utm_source=rss&utm_medium=rss&utm_campaign=language-rankings-1-19

Una **aplicación** en Java es un programa de computadora que se ejecuta cuando usted utiliza el **comando java** para iniciar la Máquina Virtual de Java (JVM). Más adelante en esta sección hablaremos sobre cómo compilar y ejecutar una aplicación de Java. Primero vamos a considerar una aplicación simple que muestra una línea de texto. En la figura 2.1 se muestra el programa, seguido de un cuadro que muestra su salida.

```
1  // Fig. 2.1: Bienvenido1.java
2  // Programa para imprimir texto.
3
4  public class Bienvenido1
5  {
6     // el método main empieza la ejecución de la aplicación en Java
7     public static void main(String[] args)
8     {
9        System.out.println("Bienvenido a la programacion en Java!");
10    } // fin del método main
11 } // fin de la clase Bienvenido1
```