Índice

Prólogo

INTRODUCCIÓN

En un mundo como el actual, en el que existen un gran número de profesionales en las diferentes áreas de las TI, la posesión de las certificaciones de los fabricantes constituye un hecho diferenciador que, ante igualdad de conocimientos y experiencia, permitirá a las personas que dispongan de alguna certificación, situarse en una mejor posición a la hora de conseguir un puesto de trabajo o una mejora profesional, frente a aquellos que no la posean.

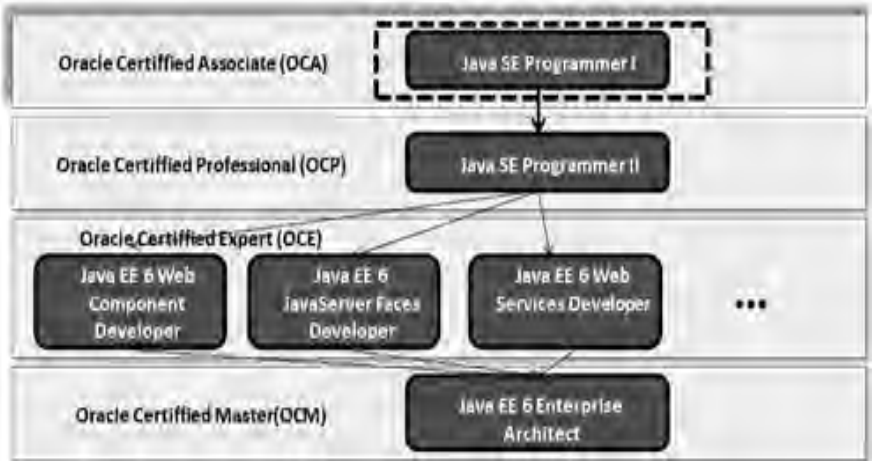
En este libro nos adentramos en el mundo de las certificaciones Java, centrándonos en la preparación del primer examen dentro del stack de certificaciones existentes sobre este lenguaje-tecnología.

OBJETIVOS

El objetivo de este libro es preparar al lector para superar el examen de certificación **Java SE Programmer I (código iz0-808),** que nos capacitaría como Programador Java Oracle Asociado.

Como se ha indicado, este examen corresponde al primer nivel dentro del conjunto de certificaciones Java y nos abriría la puerta a adquirir las certificaciones de especialización en las diferentes tecnologías que forman la plataforma Java.

En la siguiente figura podemos ver el conocido como Certification Path o pila de certificaciones Java:



Los objetivos de este examen se centran en el conocimiento detallado del lenguaje Java, así como de las clases de uso general, como las clases para el manejo de cadenas, fechas o colecciones. Concretamente, los objetivos a cubrir por el examen están organizados en nueve bloques:

- Conceptos básicos de Java
- Tipos de datos de Java
- Operadores y estructuras de decisión
- Arrays
- Estructuras repetitivas
- Métodos, constructores y encapsulación
- Estudio de la herencia
- Excepciones
- APIs de uso general

CARACTERÍSTICAS DEL EXAMEN DE CERTIFICACIÓN

El examen de certificación se realiza en centros autorizados por Oracle, repartidos a lo largo del mundo.

En la siguiente dirección, podemos encontrar información sobre los pasos que debemos seguir para realizar el examen:

https://education.oracle.com/pls/web_prod-plq-dad/db_pages.getpage?page_id=5001&get_params=p_exam_id:1Z0-808

Como datos más interesantes, comentar que el examen tiene un precio aproximado de 212 euros y consta de 70 preguntas tipo test (algunas con respuesta única y otras con respuesta múltiple) que deberán resolverse en un tiempo máximo de 150 minutos. Aunque las respuestas incorrectas no restan, para superar el examen es necesario obtener un 65% de preguntas acertadas.

A QUIÉN VA DIRIGIDO EL LIBRO

Este libro está dirigido a programadores en cualquier lenguaje de programación que quieran introducirse en Java.

Pero en general, tanto si se trata de programadores iniciados como programadores con experiencia, el libro resultará ser una herramienta muy valiosa para la preparación del examen de certificación, puesto que las preguntas del examen se centran en detalles de uso poco habitual que requieren de una preparación especial.

ESTRUCTURA Y METODOLOGÍA DEL LIBRO

El libro está organizado en nueve capítulos, que se corresponden con cada uno de los bloques que forman los objetivos de examen. En cada capítulo, además de explicarse cada concepto con su correspondiente ejemplo de código, se hace especial hincapié en aquellos aspectos relevantes que se deben de tener en cuenta de cara a las posibles preguntas de examen.

Al final de cada capítulo se incluyen un juego de preguntas tipo de examen, muy parecidas a las que podemos encontrar en los exámenes reales, incluso están enunciadas en inglés para ir acostumbrando al lector al escenario real. Después de cada batería de preguntas, se incluyen las soluciones a cada una de ellas con la explicación correspondiente.

Este no es un libro para aprender a programar en Java, está orientado a proporcionar un conocimiento profundo del lenguaje y de todos aquellos detalles más desconocidos o de uso menos habitual en el día a día de la programación, pero que pueden ser objeto de alguna pregunta de examen. Siguiendo el orden de estudio de los capítulos, prestando atención a las diferentes indicaciones que se dan en los mismos, y realizando y razonando las preguntas tipo propuestas al final de cada capítulo, las posibilidades de superar el examen de certificación Java Programmer I son muy elevadas.

EL AUTOR

Antonio Martín Sierra nació en Madrid en 1966. Es diplomado en Ingeniería Técnica de Telecomunicación por la Universidad Politécnica de Madrid y es Programador Java Certificado.

Su trayectoria profesional ha estado ligada desde el principio de su carrera a la formación. Inicialmente, impartía cursos de Electrónica para diversas Escuelas de formación de Madrid. Desde mediados de los 90, orientó su carrera al mundo del desarrollo software, especializándose en la tecnología Java. Ha impartido numerosos cursos de formación sobre Java, Java EE y frameworks en importantes compañías como Telefónica, Indra, IBM y Core Networks, y ha publicado varios libros sobre tecnologías relacionadas con estas áreas.

Además de su experiencia como formador, ha colaborado con diferentes empresas en el diseño, gestión e implementación de soluciones formativas e-learning. Ha sido uno de los responsables del programa Empleo Digital de Telefónica Educación y desarrolla contenidos formativos en las áreas de su especialidad para empresas y centros de formación.

1 Fundamentos de Java

CARACTERÍSTICAS DE JAVA

El lenguaje de programación Java es, sin lugar a dudas, el más utilizado hoy en día para la creación de aplicaciones informáticas. Ya sean aplicaciones de escritorio, para dispositivos móviles y, muy especialmente, aplicaciones para la Web, Java suele ser la opción preferida por los programadores y empresas de desarrollo.

Origen y evolución

La primera versión del lenguaje Java fue lanzada por Sun Microsystems en mayo de 1995, se trataba de la versión 1. Después vendrían la 1.1, 1.2, 1.3 y 1.4. En el año 2004 se lanzó la versión 1.5, que trajo notables mejoras respecto a su predecesora, hasta tal punto que a partir de ese momento dejó de llamarse Java 1.5 y pasó a ser Java 5. Desde entonces, han ido apareciendo Java 6, Java 7, Java 8 y la última hasta el momento, Java 9.

Desde su aparición a mediados de los 90, Java no ha hecho más que crecer y extenderse. Ya en su primera versión, incorporó una característica que comentaremos seguidamente y que hizo que tuviera gran aceptación por parte de la mayoría de las empresas de software del momento, se trata de la posibilidad de compilar una vez y ejecutar en cualquier parte, algo que sin lugar a dudas fue una auténtica novedad en aquella época.

Principales características de Java

El conocimiento de las características del lenguaje Java es uno de los objetivos del examen de certificación y, por tanto, puedes encontrar alguna pregunta sobre ello. Así pues, vamos a comentar las principales características de la tecnología Java en general y el lenguaje en particular. Pasamos a enumerarlas a continuación:

- **Lenguaje orientado a objetos**. Posiblemente, hoy en día esto no resulte una novedad, pero en la época en la que apareció Java había muy pocos lenguajes que fueran orientados a objetos. En los lenguajes orientados a objetos, el código se escribe dentro de clases, organizado en funciones (métodos), que son invocados a través de objetos o instancias de la clase. Estos lenguajes exponen una serie de características de orientación a objetos como herencia, sobrecarga y sobrescritura, polimorfismo y encapsulación, que dotan al lenguaje de una gran potencia.
- **Portabilidad.** Quizá esta sea la principal característica de Java, y es que un programa escritor en Java se **compila una vez y se ejecuta en cualquier parte**. El resultado de la compilación (bytecodes) es independiente de la plataforma, puede ejecutarse igual en un sistema operativo Windows, en Linux, etc. Esto es gracias a un elemento clave de la tecnología Java, conocido como Máquina Virtual o Java Virtual Machine (JVM), que estudiaremos a continuación.
- **Encapsulación**. Es una propiedad de la orientación a objetos y a la que se le da especial importancia en el examen de certificación. La encapsulación consiste en proteger a los atributos de la clase con modificador privado para que no sean accesibles directamente desde el exterior, permitiendo el acceso a los mismos a través de métodos. Hablaremos de la encapsulación con más detenimiento en un capítulo posterior.
- **Robusto.** La memoria es gestionada de forma automática por la JVM, de forma que no se permite el acceso a ella desde código y se evitan problemas de violación de acceso a partes de la memoria.
- **Seguro**. El código Java se ejecuta en un entorno controlado por la JVM, impidiendo operaciones dañinas sobre el equipo.

Compilación de un programa Java

Los programas Java se escriben en archivos .java, que es lo que se conoce como **código fuente del programa**. Como se ha dicho antes, este código fuente se estructura en clases, de modo que al compilarlo se generará un archivo .class, conocido como **bytecodes**, por cada clase definida en el archivo.

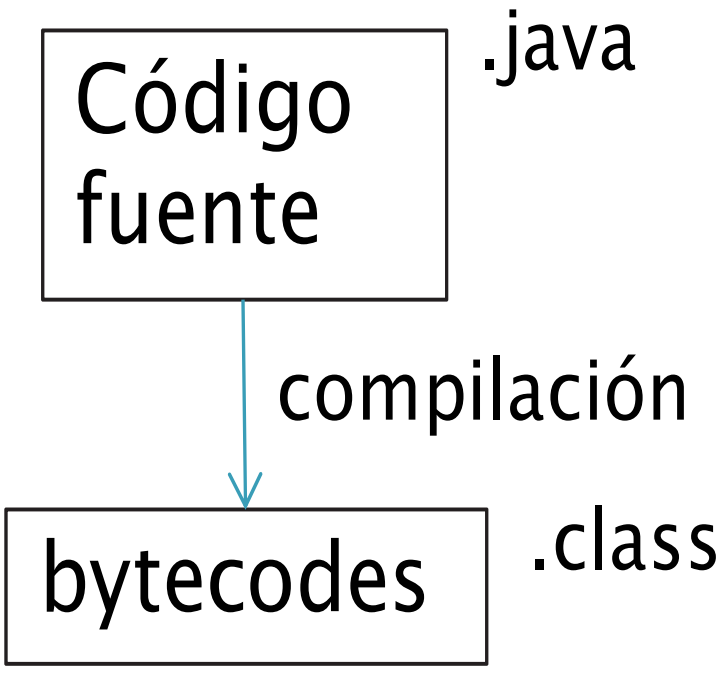


Fig. 1-1. Compilación en Java

Los bytecodes son independientes de la plataforma y se podrán ejecutar en cualquier sistema operativo que cuente con la Máquina Virtual Java (JVM).

Java Virtual Machine (JVM)

La Máquina Virtual Java, o JVM, es el software que se encarga de traducir en tiempo de ejecución los bytecodes a instrucciones comprensibles por el sistema operativo.

Existen versiones de JVM para cada todos los sistemas operativos modernos, lo que permite que un programa compilado (conjunto de archivos .class) pueda ser ejecutado en cualquier sistema operativo.

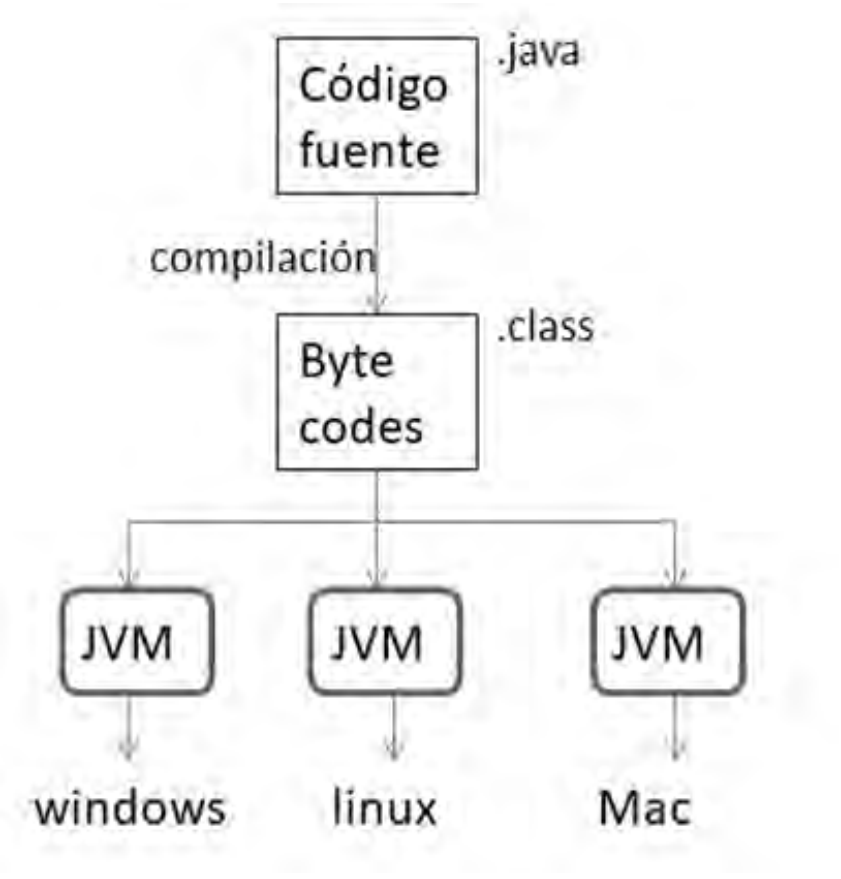


Fig. 1-2. La Máquina Virtual Java

Las primeras máquinas virtuales eran algo lentas a la hora de interpretar las instrucciones de bytecodes, lo que ya desde el principio le valió algunos enemigos a esta tecnología. Sin embargo, en las posteriores versiones se fue mejorando el rendimiento con la incorporación de los compiladores JIT, que compilan ciertas partes de código que tienen que interpretar para no repetir el proceso de interpretación cuando tienen que volver a ejecutar estos bloques.

Además del intérprete, una JVM incluye otros componentes software, como el **Gestor de multitarea**, el **Recolector de basura o Garbarge Collector** que se encarga de la gestión de memoria, el **Cargador de clases** o Class loader y el ya mencionado **Compilador JIT**.

CREANDO UN PROGRAMA JAVA: LAS CLASES

Según hemos indicado en el apartado anterior, todo programa Java se organiza en clases, pero vamos a ver cómo define una clase y cuál es su estructura.

Clase Java

Todo el código de un programa Java se escribe dentro de una clase. El objetivo de la clase es definir el comportamiento de los objetos de la misma. Este comportamiento se implementa en forma de **atributos**, que sirven para fijar las características de los objetos de la clase, y de **métodos** o funciones que definen las operaciones que se podrán realizar sobre dichos objetos.

Una vez definida la clase, se podrán crear objetos de la misma para poder hacer uso de los atributos y métodos. Dicho de otra manera, la clase es el molde y los objetos son el elemento "físico", creado a partir del molde, sobre el que se aplican las características y los comportamientos definidos.

Como ya hemos indicado, las clases se definen en archivos .java y al compilarlos, se generarán tantos .class como clases estén definidas en el código fuente.

Estructura de una clase

Una clase se define con la palabra reservada *class*, seguida del nombre de la clase. Entre llaves ({ y }) se define el contenido de la misma.

```
class NombreClase{



}
```

Un archivo .java puede contener varias clases, aunque **solo una con modificador public**, cuyo nombre debe coincidir con el del archivo .java.

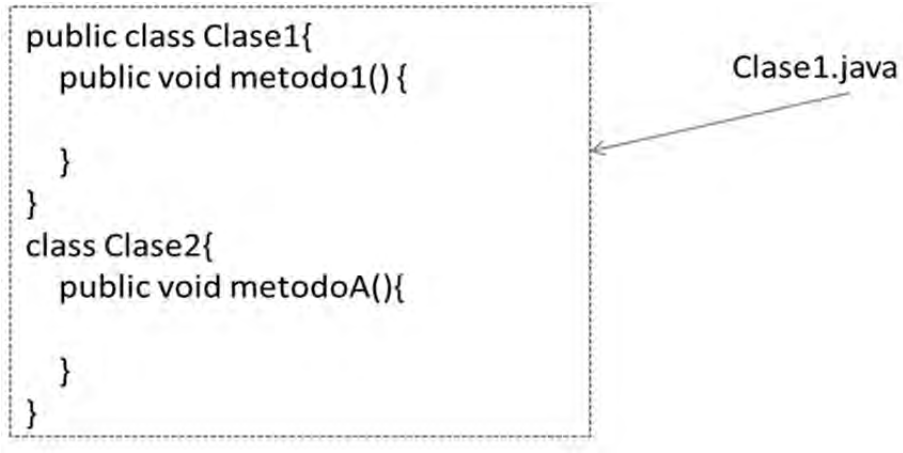


Fig. 1-3. Contenido archivo .java

Debes de tener cuidado ante una pregunta de examen en la que aparezcan dos clases públicas en un mismo archivo .java, esto supondría directamente un error de compilación, como también lo sería el hecho de que el nombre de la clase pública no coincida con el del archivo.

Como ya se ha indicado, una clase puede contener:

- **atributos**. Variables que almacenan propiedades de los objetos de la clase.
- **constructores**. Bloques de código que se ejecutan al crear objetos de la clase.
- **métodos**. Funciones que realizan tareas sobre los objetos de la clase.

Aquí tenemos un ejemplo completo de una definición de clase:

```
public class Clase1{

    int k; //atributo

    public Clase1(){ //constructor

    }


   public void metodo1() { //método

    }

}
```

A partir de una clase, se pueden crear objetos de la misma para poder hacer uso de los métodos. Aunque ya lo veremos en capítulos posteriores, la creación de un objeto se realiza utilizando el operador **new**:

```
Clase1 c=new Clase1();
```

Una vez que se tiene el objeto referenciado por la variable c, se puede utilizar el operador punto (.) para llamar a los métodos sobre el objeto:

```
c.metodo1(); //ejecuta el método
```

Empaquetado de una clase

Las clases se organizan en **paquetes** (directorios). Un paquete puede contener varios archivos .class y a su vez otros subpaquetes.

Los paquetes se definen con la sentencia ***package*** *nombrepaquete* al principio del archivo .java, de modo que todas las clases definidas en dicho archivo estarán en el mismo paquete. En el siguiente ejemplo, las clases Clase1 y Clase2 estarán en el paquete mypack:

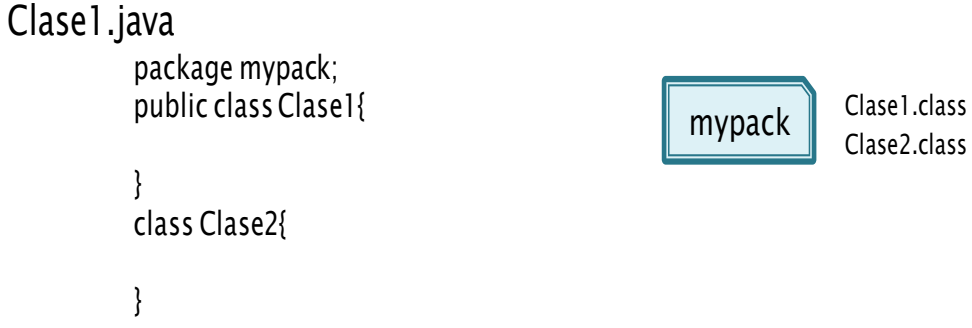


Fig. 1-4. Empaquetado de clases

Aunque insistiremos en ello más adelante, hay que destacar que la sentencia package debe ser **la primera del archivo .java**.

El método main

El método *main()* representa el punto de entrada de un programa Java. Es el método que lanza la JVM cuando se le indique que tiene que ejecutar una clase. Entre todas las clases de un programa Java, deberá haber una que incluya este método.

Es importante conocer el formato exacto del método *main()*, ya que alguna pregunta de examen puede ir en esa dirección. El formato exacto es el que se indica a continuación:

```
public class Clase1{
    public static void main(String [] args) {
        :
    }
}
```

La única variación admitida sería que, en lugar de definir un array de String como parámetro, se puede indicar un número variable de argumentos:

```
public class Clase1{
    public static void main(String ... args) {
        :
    }
}
```

Las siguientes definiciones del método main **no** serían correctas:

```
static void main(String[] args) //falta public
public void main(String[] args) //falta static
public static int main(String[] args) //tipo de devolución
                                      //incorrecto
public static void Main(String[] args) //nombre incorrecto
```

En los casos anteriores, estos métodos **no provocarían error de compilación** pues son sintácticamente correctos, lo que sucedería es que al intentar ejecutar la clase se produciría un error de ejecución ya que la JVM no encontraría el método main con el formato adecuado.

COMPILACIÓN Y EJECUCIÓN DE PROGRAMAS JAVA

Anteriormente, hemos indicado que al compilar un archivo .java se generan uno o varios .class, y que al ejecutar una clase la JVM busca el método *main()*. Pero ¿cómo se compila un código fuente Java? ¿Cómo se ejecuta una clase? Uno de los objetivos del examen de programador Java asociado es, precisamente, la utilización de los comandos para realizar estas operaciones. Vamos a ver cuáles son y cómo se utilizan.

Herramientas JDK

El comando para la compilación de un archivo .java se llama javac.exe, mientras que el que se emplea para la ejecución de una clase es el java.exe. Ambos se encuentran en el directorio de instalación del JDK, dentro de la subcarpeta bin:

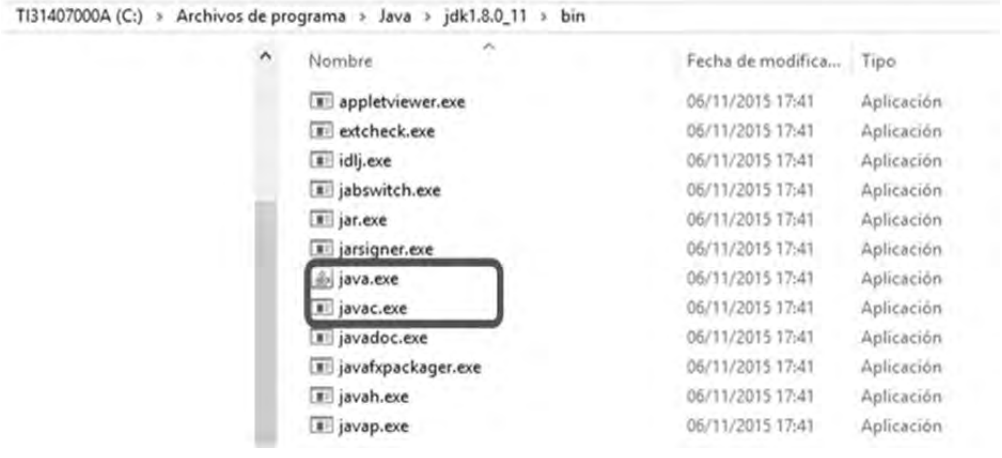


Fig. 1-5. Comandos del JDK

El JDK, o Java Development Kit, proporciona las herramientas básicas para poder compilar y ejecutar programas Java, así como las clases que forman el Java Standard Edition (Java SE).