

# **Curso de Preparación para el Examen de Certificación para la Plataforma Java, Edición Estándar 6.**

**(Sun Certified Programmer CX-310-065)**

## **Objetivo:**

Esta preparación para la certificación está dirigida a programadores con experiencia usando el lenguaje Java. A través de este curso el programador adquirirá las capacidades necesarias para obtener la certificación que otorga Sun Microsystems.

La preparación que ofrecemos permitirá al estudiante enfrentar con seguridad el examen de certificación de Sun Microsystems (CX-310-065).

Adquirir esta certificación internacional le permite a un programador avalar su conocimiento de la sintaxis y estructura del lenguaje Java y su capacidad para desarrollar aplicaciones que se ejecuten usando la versión 6 del lenguaje.

## **Requisitos:**

Experiencia en el desarrollo de aplicaciones en el lenguaje Java u otro de paradigma orientado a objetos, como C++.

## **Enfoque del Curso:**

Está estructurado en dos niveles, cada uno cubriendo un grupo distinto de objetivos presentes en el examen de certificación. Cada nivel cuenta con

material de auto-evaluación y preguntas modelo de igual nivel de dificultad a las presentes en el examen de certificación. Todo el material de apoyo (que en total suma unas 1200 páginas) está en español, facilitando al estudiante el aprendizaje de las estructuras propias del lenguaje Java.

## Módulos y Contenido

El examen de certificación internacional de Sun Microsystems tiene objetivos definidos en los temas de Control de Flujo, Contenidos del API de Java 6, Concurrencia, Conceptos de Orientación a Objetos, Colecciones, Tipos Genéricos, Fundamentos del Lenguaje, Declaraciones, Inicialización y Alcance. Estos objetivos oficiales del examen han sido divididos en dos niveles integrados de estudio, permitiendo al estudiante un aprendizaje semánticamente relevante del material.

El nivel 1 del curso abarca los módulos del 1 al 5 y el nivel 2 los módulos del 6 al 10.

## Nivel 1

### Módulo I: Declaraciones y Control de Acceso

*Objetivos del Examen de Certificación Tratados:*

#### **Declaración de Clases e Interfaces (Objetivos de Examen 1.1 y 1.2)**

*1.2 Desarrollar código que declare clases (incluyendo clases abstractas y todas las formas de clases anidadas), interfaces y enums e incluya el uso apropiado de declaraciones de package e import (incluyendo static imports).*

*1.1 Desarrollar código que declare una interface. Desarrollar código que implemente o extienda una o más interfaces. Desarrollar código que declare una clase abstracta. Desarrollar código que extienda una clase abstracta.*

## **Declaración de Miembros de Clases, Restricciones de Acceso, Identificadores y JavaBeans (Objetivos 1.3 y 1.4)**

*1.3 Desarrollar código que declare, inicialice y use primitivos, arreglos, enums y objetos como variables static, de instancia y locales. También, usar identificadores legales para los nombres de variables.*

*1.4 Desarrollar código que declare métodos static y no-static, y - si es apropiado- usar nombres de métodos que se adhieran a los estándares de nombramiento de JavaBeans. También desarrollar código que declare y use listas de argumentos de longitud variable.*

## **Módulo II: Orientación a Objetos**

*Objetivos del Examen de Certificación Tratados:*

### **Sobrescritura (Objetivos de Examen 1.4 y 1.5)**

*1.4 Dado un escenario, desarrollar código que declare y / o invoque métodos sobrescritos o sobrecargados y código que declare y / o invoque constructores de superclase, sobrescritos o sobrecargados.*

*1.5 Dado un ejemplo de código, determinar si un método está correctamente sobrescribiendo o sobrecargando otro método, e identificar los valores de retorno legales (incluyendo retornos covariantes), para el método.*

## **Encapsulación (Objetivo del Examen 5.1)**

*5.1 Desarrollar código que implemente encapsulación fuerte, bajo acoplamiento y alta cohesión en clases y describir los beneficios que este tipo de desarrollo presenta.*

## **Polimorfismo (Objetivo del Examen 5.2)**

*5.2 Dado un escenario, desarrollar código que demuestra el uso de polimorfismo. Además determinar cuando el casting es necesario y reconocer errores de compilación frente a errores de tiempo de ejecución relacionados al casting de referencias de objetos.*

## **Herencia, Relaciones Es-Un y Tiene-Un, (Objetivo de Examen 5.5)**

*5.5 Desarrollar y distinguir entre código que implemente las relaciones “es-un” y “tiene-un”.*

## **Casting de Variables de Referencia (Objetivo 5.2)**

*5.2 Dado un escenario, desarrollar código que demuestre el uso de polimorfismo. Además, determinar cuando el casting será necesario y reconocer errores de compilación relacionados al casting de referencias a objetos.*

## **Implementando una Interface (Objetivo de Examen 1.2)**

*1.2 Desarrollar código que declare una interface.*

## **Tipos de Retorno Legales (Objetivo del Examen 1.5)**

*1.5 Dado un ejemplo de código, determinar si un método está sobrecargando o sobrescribiendo de forma correcta a otro método e identificar los valores de retorno legal (incluyendo los retornos covariantes) para el método.*

## **Constructores e Instanciación (Objetivos del Examen 1.6 y 5.4)**

*1.6 Dado un conjunto de clases y superclases, desarrollar constructores para una o más de las clases. Dada una declaración de clase, determinar si un constructor por defecto será creado, y si lo es, determinar el comportamiento de ese constructor. Dado la definición de una clase anidada o no anidada, escribir código para instanciar la clase.*

*5.4 Dado un escenario, desarrollar código que declare y/o invoque métodos sobrescritos o sobrecargados y código que declare y/o invoque constructores de superclase, sobrescritos o sobrecargados.*

## **Modificador Static (Objetivo de Examen 1.3)**

*1.3 Desarrollar código que declare, inicialice y use primitivos, arreglos, enums y objetos como variables static, de instancia y locales. También, usar identificadores legales para los nombres de variable.*

## **Acoplamiento y Cohesión (Objetivo de Examen 5.1)**

*5.1 Desarrollar código que implemente fuerte encapsulación, bajo acoplamiento (coupling) y alta cohesión en las clases, y describir los beneficios.*

## **Módulo III: Asignaciones**

*Objetivos del Examen de Certificación Tratados:*

## **Literales, Asignaciones y Variables (Objetivos de Examen 1.3 y 7.6)**

*1.3 Desarrollar código que declare, inicialice, y use primitivos, arreglos, enums y objetos como variables static, de instancia y locales. También, usar identificadores legales para los nombres de variables.*

*7.6 Escribir código que aplique de forma correcta Los operadores apropiados, incluyendo Los operadores de asignación (limitados a =, +=, -=)*

## **Pasar Variables a Métodos (Objetivo del Examen 7.3)**

*7.3 Determinar el efecto sobre Las referencias a objeto y valores primitivos cuando son pasadas a métodos que realicen asignaciones u otras operaciones de modificación de Los parámetros.*

## **Declaración de Arreglos, Construcción e Inicialización (Objetivo del Examen 1.3)**

*1.3 Desarrollar código que declare, inicialice y use primitivos, arreglos, enums y objetos como variables static, de instancia y Locales. También, usar identificadores legales para Los nombres de Las variables.*

## **Usando Clases Wrapper y Boxing (Objetivo de Examen 3.1)**

*3.1 Desarrollar código que use Las clases wrapper de primitivos (tales como Boolean, Character, Double, Integer, etc.) y / o autoboxing y unboxing. Discutir Las diferencias entre Las clases String, StringBuilder y StringBuffer.*

## **Sobrecarga (Objetivos de Examen 1.5 y 5.4)**

*1.5 Dada una muestra de código, determinar si un método está sobrescribiendo o sobrecargando otro método e identificar valores de retorno legal (incluyendo retornos covariantes), para el método.*

*5.4 Dado un escenario, desarrollar código que declare y / o invoque métodos sobrescritos o sobrecargados.*

## **Recolección de Basura (Objetivo de Examen 7.4)**

*7.5 Dada una muestra de código, reconocer el punto en el cual un objeto se hace elegible para la recolección de basura y determinar que está y que no está garantizado por el sistema de recolección de basura y reconocer los comportamientos del método finalize() de Object.*

## **Módulo IV: Operadores**

*Objetivos del Examen de Certificación Tratados:*

### **Operadores de Java (Objetivo del Examen 7.6)**

*7.6 Escribir código que aplique los operadores apropiados, incluyendo los operadores de asignación (limitados a: =, +=, -=), operadores aritméticos (limitados a: <, <=, >, >=, =, !=), el operador instanceof, los operadores lógicos (limitados a: &, |, ^, !, &&, ||), y el operador condicional (? :), para producir un resultado deseado. Escribir código que determine la igualdad de dos objetos o dos primitivos.*

## **Módulo V: Control de Flujo, Excepciones y Aserciones**

*Objetivos del Examen de Certificación Tratados:*

### **Declaraciones if y switch (Objetivo del Examen 2.1)**

*2.1 Desarrollar código que implemente una declaración if o switch e identificar tipos de argumentos legales para estas declaraciones*

### **Ciclos e Iteradores (Objetivo de Examen 2.2)**

*2.2 Desarrollar código que implemente todas las formas de ciclos e iteradores, incluyendo el uso de for, el for expandido (for-each), do, while, etiquetas, y continue; y explicar los valores tomados por las variables contadoras de ciclo durante y después de la ejecución del ciclo.*

## **Manejando Excepciones (Objetivos de Examen 2.4 y 2.5)**

*2.4 Desarrollar código que haga uso de excepciones y declaraciones de manejo de excepciones (try, catch, finally), y declare métodos y métodos de sobrescritura que lancen excepciones.*

*2.5 Reconocer el efecto de una excepción surgiendo en un punto específico en un fragmento de código. Toma en cuenta que la excepción puede ser una excepción en tiempo de ejecución, una excepción chequeada, o un error.*

## **Excepciones y Errores Comunes (Objetivo del Examen 2.6)**

*2.6 Reconocer situaciones que resultarán en cualquiera de las siguientes excepciones siendo lanzada: `ArrayIndexOutOfBoundsException`, `ClassCastException`, `IllegalArgumentException`, `IllegalStateException`, `NullPointerException`, `NumberFormatException`, `AssertionError`, `ExceptionInInitializerError`, `StackOverflowError`, `NoClassDefFoundError`. Entender cuáles de estas son lanzadas por la máquina virtual y reconocer situaciones en las cuales otras deben ser lanzadas programáticamente.*

## **Trabajando con el Mecanismo de Aserciones (Objetivo de Examen 2.3)**

*2.3 Desarrollar código que haga uso de las aserciones y distinguir entre el uso apropiado e inapropiado de las aserciones.*

## **Nivel 2**

### **Módulo VI: Strings, I/O, Formateo y Parsing**

#### **String, StringBuilder y StringBuffer (Objetivo del Examen 3.1)**

*3.1 Discutir las diferencias entre las clases String, StringBuilder y StringBuffer.*

#### **Navegación de Archivos e I/O (Objetivo de Examen 3.2)**

*3.2 Dado un escenario que involucre la navegación de sistemas de archivos, leer a partir de archivos, o escribir archivos, desarrollar la solución correcta usando las siguientes clases (algunas en combinación) a partir de java.io: BufferedReader, BufferedWriter, File, FileReader, FileWriter y PrintWriter.*

#### **Serialización (Objetivo de Examen 3.3)**

*3.3 Desarrollar código que analiza y/o de-serializa objetos usando las siguientes APIs de java.io: DataInputStream, DataOutputStream, FileInputStream, FileOutputStream, ObjectInputStream, ObjectOutputStream y Serializable*

#### **Fechas, Números y Monedas (Objetivo de Examen 3.4)**

*3.4 Usar las APIs estándar J2SE en el paquete java.text para formatear o convertir correctamente fechas, números y valores de moneda para una localidad específica; y, dado un escenario, determinar los métodos apropiados a usar si quieres usar la localidad por defecto o una localidad específica. Describir el propósito y uso de la clase java.util.Locale.*

## **Conversión, Tokenizing y Formateo (Objetivo de Examen 3.5)**

*3.5 Crear código que use APIs estándar de J2SE en Los paquetes java.util y java.util.regex para formatear o convertir strings o streams. Para Los strings, escribir código que use Las clases Pattern y Matcher y el método String.split. Reconocer y usar patrones de expresiones regulares para hacer apareo (limitado a: .(punto), \*, +, ?, \d, \s, \2, [], ()). El uso de \*, +, y ? será limitado a Los greedy quantifiers, y el operador de paréntesis sólo será usado como un mecanismo de agrupación, no para capturar contenido durante el apareo. Para Los streams, escribir código usando Las clases Formatter y Scanner y Los métodos PrintWriter.format/printf. Reconocer y usar Los parámetros de formateo (limitado a: %b,%c,%d,%f,%s) en el formateo de Strings.*

## **Módulo VII: Colecciones y Generics**

*Objetivos del Examen de Certificación Tratados:*

### **Sobrescribiendo equals() y hashCode() (Objetivo del Examen 6.2)**

*6.2. Distinguir entre sobrescrituras correctas e incorrectas de métodos equals y hashCode correspondientes y explicar La diferencia entre == y el método equals.*

### **Colecciones (Objetivo del Examen 6.1)**

*6.1 Dado un escenario de diseño, determinar que clases de colecciones y / o interfaces deben ser usadas para implementar de forma correcta ese diseño, incluyendo el uso de La interfaz Comparable*

## **Usando el Framework de Collections (Objetivo del Examen 6.5)**

*6.5. Usar las capacidades del paquete `java.util` para escribir código que manipule una lista por clasificación, realizando una búsqueda binaria, o convirtiendo la lista a un arreglo. Usar capacidades del paquete `java.util` para escribir código que manipule un arreglo clasificándolo, realizando una búsqueda binaria o convirtiendo el arreglo a una lista. Usar las interfaces `java.util.Comparator` y `java.util.Comparable` para manipular la clasificación de listas y arreglos. Además, reconocer el efecto del 'ordenamiento natural' de clases wrapper de primitivos y `java.lang.String` en la clasificación de ítems.*

## **Tipos Genéricos (Objetivos del Examen 6.3 y 6.4)**

*6.3 Escribir código que use las versiones genéricas del API de Collections, en particular el de las interfaces `Set`, `List`, `Map` y las clases de implementación. Reconocer las limitaciones del API de Colecciones no genérico y como refactorizar código para que use las versiones genéricas. Escribir código que use las interfaces `NavigableSet` y `NavigableMap`.*

*6.4 Desarrollar código que haga uso apropiado de los tipos de parámetros en las declaraciones de clase/interface, variables de instancia, argumentos de métodos, y tipos de retorno; y escribir métodos genéricos o métodos que hagan uso de tipos comodín y entender las similitudes y diferencias entre ambas formas.*

## **Módulo VIII: Clases Internas**

(Este módulo no toca directamente algún objetivo del examen de certificación, trata material avanzado que estará presente en ciertas preguntas del examen de certificación para los otros objetivos).

### **8a) Introducción a las Clases Internas**

- 8b) Clases Internas Simples
- 8c) Instanciando una Clase Interna
- 8d) Referenciando una Instancia Interna o Externa a Partir de una Clase Interna
- 8e) Modificadores Aplicables a las Clases Internas
- 8f) Clases Internas Locales a Un Método
- 8g) Lo que un Objeto Local a un Método Puede Hacer
- 8h) Clases Internas Anónimas
- 8i) Clases Internas Anónimas Definidas en Argumentos
- 8j) Clases static Anidadas
- 8k) Instanciando y Usando Clases Internas static Anidadas

## **Módulo IX: Threads y Manejo de Concurrency**

*Objetivos del Examen de Certificación Tratados:*

### **Definir, Instanciar e Iniciar Threads (Objetivo 4.1)**

*4.1 Escribir código para definir, instanciar e iniciar nuevos threads usando `java.Lang.Thread` y `java.Lang.Runnable`*

### **Estados de Threads y Transiciones de Estado (Objetivo 4.2)**

4.2 Reconocer Los estados en Los que un thread puede encontrarse, e identificar Las maneras en Las que un thread puede hacer transición de un estado a otro.

### **Sincronizando Código (Objetivo 4.3)**

4.3 Dado un escenario, escribir código que haga uso apropiado de Los Locks a objetos para prevenir problemas de acceso concurrente en variables de instancia o variables static.

### **Interacción de Threads (Objetivo 4.4)**

4.4 Dado un escenario, escribir código que haga uso apropiado de Los métodos wait, notify, o notifyAll

## **Módulo X: Desarrollo**

*Objetivos del Examen de Certificación Tratados:*

### **Usando los comandos java y javac (Objetivos 7.1, 7.2 y 7.5)**

7.1 Dado un ejemplo de un código y un escenario, crear código que use apropiadamente Los modificadores de acceso, Las declaraciones de paquete y declaraciones import (a través de acceso o herencia) para interactuar con el código en el ejemplo.

7.2 Dado un ejemplo de una clase y una línea de comando, determinar el comportamiento en tiempo de ejecución apropiado.

7.5 Dado el nombre completamente calificado de una clase que sea implementada dentro y / o fuera de un archivo JAR, construir La estructura de directorio apropiada para esa clase. Dado un ejemplo de código y un classpath, determinar si el classpath permitirá al código compilarse exitosamente.

## Archivos JAR (Objetivo del Examen 7.5)

7.5 Dado el nombre completamente calificado de una clase que está implementada dentro y / o fuera de un archivo JAR, construir la estructura de directorio apropiada para esa clase. Dado un ejemplo de código y un classpath, determinar si el classpath permitirá al código compilar exitosamente.

## Usando Imports static (Objetivo del Examen 7.1)

7.1 Dado un ejemplo de código y escenario, escribir código que use los modificadores de acceso apropiados, declaraciones de paquete, y declaraciones import para interactuar (a través de acceso o herencia) con el código en el ejemplo.

### Fechas

#### Nivel 1

Inicio	Fin	Días	Horario
10/04/2010	08/05/2010	Sábados	8 am a 12 pm 1 pm a 5 pm

Precio por Participante Bs. 2.400,00

#### Nivel 2

Inicio	Fin	Días	Horario
15/05/2010	05/06/2010	Sábados	8 am a 12 pm 1 pm a 5 pm

Precio por Participante Bs. 2.400,00

Descuento del 15 % si el participante se inscribe en los dos niveles

Cupo Mínimo: 8 participantes.

## Más información e Inscripciones en CENEAC

### Producciones:

**Dirección:** UCV, Facultad de Ciencias, Escuela Computación, Edif. III, Planta Alta, Av. Los Ilustres, Los Chaguaramos (a 2 cuadras Est. Metro Los Símbolos). Horario: Lunes a Viernes, 8.30am a 12.30 y de 2 a 5.30pm

**Formas de Pago:** Efectivo • Cheque Conformable • Tarjeta Débito o Crédito

• Depósito a nombre de CENEAC PRODUCCIONES C.A., cCta. Corriente N° 0108-0019-62-0100048829 del Banco Provincial. RIF CENEAC Producciones C.A.: J-30529216-7

(0212) 693.47.38(telfax) 605.13.14    [ceneac@ciens.ucv.ve](mailto:ceneac@ciens.ucv.ve)    [www.ceneac.com.ve](http://www.ceneac.com.ve)