

ACCION FORMATIVA: SUN CERTIFIED JAVA PROGRAMER (SCJP)



TEMA 1: DECLARACIONES, INICIALIZACIÓN Y ÁMBITO

- Desarrollar código que declare clases (incluidas clases abstractas y todas las formas de clases anidadas), interfaces y enums, y que incluya el uso apropiado de instrucciones package e import (incluidas importaciones estáticas).
- Desarrollar código que declare una interfaz. Desarrollar código que implemente o amplíe una o más interfaces. Desarrollar código que declare una clase abstracta. Desarrollar código que amplíe una clase abstracta.
- Desarrollar código que declare, inicialice y use primitivas, arreglos, enums y objetos como variables estáticas, de instancia y locales. Asimismo, utilizar identificadores válidos para los nombres de variable.
- Desarrollar código que declare métodos tanto estáticos como no estáticos y, si resulta adecuado, que utilice nombres de método con nomenclatura de JavaBeans. Además, desarrollar código que declare y utilice una lista de argumentos de longitud variable.
- Con un ejemplo de código dado, determinar si un método está sobrescribiendo o sobrecargando correctamente otro método e identificar valores de retorno válidos (incluidos valores de retorno covariantes) para el método.
- A partir de una serie de clases y superclases, desarrollar constructores para una o varias clases.

TEMA 2: CONTROL DE FLUJO

- Desarrollar código que implemente una instrucción if o switch e identificar tipos de argumentos válidos para estas instrucciones.
- Desarrollar código que implemente todas las formas de ciclos e iteradores, incluidos el uso de for, el ciclo mejorado (for-each), do, while, labels, break y continue. Indicar los valores que adoptan las variables de control del ciclo durante y después de la ejecución del ciclo.
- Escribir código que utilice afirmaciones y distinga entre el uso apropiado e inapropiado de las afirmaciones.

- Desarrollar código que utilice excepciones y cláusulas de manejo de excepciones (try, catch, finally), y declarar métodos y sobrescribir métodos que generen excepciones.
- Reconocer el efecto que produce una excepción que se genera en un punto dado de un fragmento de código. Puede ser una excepción runtime, una excepción comprobada (checked) o un error.
- Reconocer las situaciones en las que se generará alguna de las siguientes excepciones:
ArrayIndexOutOfBoundsException, ClassCastException,
IllegalArgumentException,
IllegalStateException,
NullPointerException,
NumberFormatException,
AssertionError,
ExceptionInInitializerError,
StackOverflowError o NoClassDefFoundError.
Saber cuáles genera la máquina virtual y en qué situaciones deberían generarse programáticamente otras excepciones.

TEMA 3: CONTENIDO DEL API

- Desarrollar código que utilice las clases wrapper primitivas (como booleano, carácter, doble, entero, etc.) y/o autoboxing y unboxing. Describir las diferencias entre las clases String, StringBuilder y StringBuffer.
- En una situación en la que se requiera desplazarse por los file systems, leer archivos o escribir en archivos, desarrollar la solución adecuada mediante el uso de las siguientes clases (o una combinación de ellas) del paquete java.io: BufferedReader, BufferedWriter, File, FileReader, FileWriter y PrintWriter.
- Desarrollar código que serialice y/o deserialice objetos mediante el uso de las siguientes API de java.io: DataInputStream, DataOutputStream, FileInputStream, FileOutputStream, ObjectInputStream, ObjectOutputStream y Serializable.
- Utilizar API de J2SE estándar del paquete java.text para asignar formato y analizar correctamente fechas, números y valores de moneda de una configuración regional específica. En una situación dada, determinar los métodos que se deben utilizar si se desea emplear la configuración regional predeterminada u otra específica. Describir el objetivo y la utilidad de la clase java.util.Locale.

- Escribir código que utilice API de J2SE estándar de los paquetes `java.util` y `java.util.regex` para asignar formato o analizar cadenas o secuencias. Para cadenas, escribir código que utilice las clases `Pattern` y `Matcher` y el método `String.split`. Reconocer y utilizar patrones de expresión regulares para establecer la coincidencia (limitado a: `.` (punto), `*` (asterisco), `+` (signo más), `?`, `\d`, `\s`, `\w`, `[]`, `()`). El uso de `*`, `+` y `?` se limitará a cuantificadores greedy y el operador de paréntesis sólo se empleará como mecanismo de agrupación, en lugar de para capturar contenido durante la coincidencia. Para secuencias, escribir código utilizando las clases `Formatter` y `Scanner` y los métodos `PrintWriter.format/printf`. Reconocer y utilizar los parámetros de formato (limitado a: `%b`, `%c`, `%d`, `%f`, `%s`) en cadenas de formato.

TEMA 4: CONCURRENCIA

- Escribir código para definir, instanciar e iniciar nuevos threads utilizando `java.lang.Thread` y `java.lang.Runnable`.
- Reconocer los estados en los que puede existir un thread, e identificar condiciones en las que puede pasar de un estado a otro.
- En una situación dada, escribir código que utilice correctamente el bloqueo de objetos para proteger las variables estáticas o de instancia de los problemas de acceso concurrente.

TEMA 5: CONCEPTOS DE LA PROGRAMACIÓN OO

- Desarrollar código que implemente encapsulación estricta, emparejamiento ligero y gran cohesión en las clases, y describir las ventajas que ofrece.
- En una situación dada, desarrollar código que demuestre el uso del polimorfismo. También determinar cuándo se va a necesitar la conversión de tipos y distinguir los errores del compilador de los errores de tiempo de ejecución relacionados con la conversión de referencias de objeto.
- Explicar el efecto de los modificadores en la herencia en lo que se refiere a constructores, variables de instancia o estáticas y métodos de instancia o estáticos.

TEMA 6: COLECCIONES / GENÉRICOS

- En una situación de diseño dada, determinar las clases o interfaces de colección que deberían utilizarse para implementar ese diseño de forma adecuada, incluido el uso de la interfaz `Comparable`.

- Distinguir entre los valores de reemplazo correctos e incorrectos de los métodos hashCode y equals correspondientes, y explicar la diferencia entre == y el método equals.
- Escribir código que utilice las versiones genéricas de las colecciones del API, en particular las interfaces Set, List y Map y las clases de implementación. Identificar las limitaciones de las colecciones API no genéricas y cómo refactorizar código para utilizar versiones genéricas. Escribir código que use las interfaces NavigableSet y NavigableMap.
- Desarrollar código que utilice correctamente los parámetros de tipo en las declaraciones de clase/interfaz, las variables de instancia, los argumentos de métodos y los tipos de retorno. Escribir métodos genéricos o métodos que utilicen comodines y comprender las similitudes y diferencias entre estos dos métodos.

TEMA 7: ASPECTOS BÁSICOS

- Dado un ejemplo de código y una situación, escribir código que utilice los modificadores de acceso, las declaraciones package y las instrucciones import adecuadas para interactuar (a través de acceso o herencia) con el código del ejemplo.
- Con un ejemplo de una clase y una línea de comandos, determinar el comportamiento previsto del tiempo de ejecución.
- Determinar cómo afectan las referencias de objeto y valores primitivos cuando son pasados a métodos que realizan asignaciones u otras modificaciones en los parámetros.
- Con un ejemplo de código reconocer el momento en el que un objeto se convierte en elegible del garbage collection, determinar que está y que no está garantizado por el sistema del garbage collection y reconocer los comportamientos del método Object.finalize().