I – INTRODUCCIÓN A JAVA

1. ¿Cuál es la diferencia entre una variable local y un campo?

2. ¿Qué es un objeto en el lenguaje java?

3. ¿Qué es una clase?

4. Explique el propósito de un parámetro de un método. ¿Cuál es la diferencia entre un parámetro y un argumento?

5. ¿Cuál es el propósito de la palabra clave new? Explique lo que ocurre cuando se utiliza en una aplicación.

6. ¿Qué es un constructor predeterminado? ¿Cómo se inicializan las variables de instancia de un objeto, si una clase sólo tiene un constructor predeterminado?

7. Explique cómo utilizaría un programa la clase Scanner, sin importarla del paquete **java.util**.

8. Crea un programa que solicite al usuario una cantidad en segundos y la convierta en días, horas, minutos y segundos.

9. Cree una clase llamada Empleado, que incluya tres piezas de información como variables de instancia: un primer nombre (tipo String), un apellido paterno (tipo String) y un salario mensual (double). Su clase debe tener un constructor que inicialice las tres variables de instancia. Proporcione un método establecer y un método obtener para cada variable de instancia. Si el salario mensual no es positivo, establézcalo a 0.0. Escriba una aplicación de prueba llamada PruebaEmpleado, que demuestre las capacidades de cada Empleado. Cree dos objetos Empleado y muestre el salario anual de cada objeto. Después, proporcione a cada Empleado un aumento del 10% y muestre el salario anual de cada Empleado otra vez.

10. Cree una clase llamada Fecha, que incluya tres piezas de información como variables de instancia —un mes (tipo int), un día (tipo int) y un año (tipo int). Su clase debe tener un constructor que inicialice las tres variables de instancia, y debe asumir que los valores que se proporcionan son correctos. Proporcione un método establecer y un método obtener para cada variable de instancia. Proporcione un método mostrarFecha, que muestre el mes, día y año, separados por barras diagonales (/). Escriba una aplicación de prueba llamada PruebaFecha, que demuestre las capacidades de la clase Fecha.

11. (Palíndromos) Un palíndromo es una secuencia de caracteres que se lee igual al derecho y al revés. Por ejemplo, cada uno de los siguientes enteros de cinco dígitos es un palíndromo: 12321, 55555, 45554 y 11611. Escriba una aplicación que lea un entero de cinco dígitos y determine si es un palíndromo. Si el número no es de cinco dígitos, el programa debe mostrar un mensaje de error y permitir al usuario que introduzca un nuevo valor.

12. Escriba una aplicación que muestre en la ventana de comandos los múltiplos del entero 2 (es decir, 2, 4, 8, 16, 32, 64, etcétera). Su ciclo no debe terminar (es decir, debe crear un ciclo infinito). ¿Qué ocurre cuando ejecuta este programa

13. Escriba una aplicación que pida al usuario que introduzca el tamaño del lado de un cuadrado y que muestre un cuadrado hueco de ese tamaño, compuesto de asteriscos. Su programa debe funcionar con cuadrados que tengan lados de todas las longitudes entre 1 y 20.

14. Escriba una aplicación que encuentre el menor de varios enteros. Suponga que el primer valor leído especifica el número de valores que el usuario introducirá.

15. Compare y contraste las instrucciones de repetición while y for

16. Suponga que i = 1, j = 2, k = 3 y m = 2. ¿Qué es lo que imprime cada una de las siguientes instrucciones?

a) System.out.println(i == 1);

b) System.out.println(j == 3);

c) System.out.println((i >= 1) && (j < 4));

d) System.out.println((m <= 99) & (k < m));

e) System.out.println((j >= i) || (k == m));

f) System.out.println((k + m < j) | (3 – j >= k));

g) System.out.println(!(k > m));

17. (Comisión por ventas) Utilice un arreglo unidimensional para resolver el siguiente problema: una compañía paga a sus vendedores por comisión. Los vendedores reciben $200 por semana más el 9% de sus ventas totales de esa semana. Por ejemplo, un vendedor que acumule $5000 en ventas en una semana, recibirá $200 más el 9% de $5000, o un total de $650. Escriba una aplicación (utilizando un arreglo de contadores) que determine cuántos vendedores recibieron salarios en cada uno de los siguientes rangos (suponga que el salario de cada vendedor se trunca a una cantidad entera):

a) $200-299

b) $300-399

c) $400-499

d) $500-599

e) $600-699

f) $700-799

g) $800-899

h) $900-999

i) $1000 en adelante

Sintetice los resultados en formato tabular

18. (Sistema de reservaciones de una aerolínea) Una pequeña aerolínea acaba de comprar una computadora para su nuevo sistema de reservaciones automatizado. Se le ha pedido a usted que desarrolle el nuevo sistema. Usted escribirá una aplicación para asignar asientos en cada vuelo del único avión de la aerolínea (capacidad: 10 asientos). Su aplicación debe mostrar las siguientes alternativas: Por favor escriba 1 para Primera Clase y Por favor escriba 2 para Económico. Si el usuario escribe 1, su aplicación debe asignarle un asiento en la sección de primera clase (asientos 1 a 5). Si el usuario escribe 2, su aplicación debe asignarle un asiento en la sección económica (asientos 6 a 10). Su aplicación deberá entonces imprimir un pase de abordaje, indicando el número de asiento de la persona y si se encuentra en la sección de primera clase o clase económica del avión. Use un arreglo unidimensional del tipo primitivo boolean para representar la tabla de asientos del avión. Inicialice todos los elementos del arreglo con false para indicar que todos los asientos están vacíos. A medida que se asigne cada asiento, establezca los elementos correspondientes del arreglo en true para indicar que ese asiento ya no está disponible. Su aplicación nunca deberá asignar un asiento que ya haya sido asignado. Cuando esté llena la sección económica, su programa deberá preguntar a la persona si acepta ser colocada en la sección de primera clase (y viceversa). Si la persona acepta, haga la asignación de asiento apropiada. Si no acepta, imprima el mensaje "El próximo vuelo sale en 3 horas".

19. (Ventas totales) Use un arreglo bidimensional para resolver el siguiente problema: una compañía tiene cuatro vendedores (1 a 4) que venden cinco productos distintos (1 a 5). Una vez al día, cada vendedor pasa una nota por cada tipo de producto vendido. Cada nota contiene lo siguiente:

a) El número del vendedor.

b) El número del producto.

c) El valor total en dólares de ese producto vendido en ese día.

Así, cada vendedor pasa entre 0 y 5 notas de venta por día. Suponga que está disponible la información sobre todas las notas del mes pasado. Escriba una aplicación que lea toda esta información para las ventas del último mes y que resuma las ventas totales por vendedor, por producto. Todos los totales deben guardarse en el arreglo bidimensional ventas. Después de procesar toda la información del mes pasado, muestre los resultados en formato tabular, en donde cada columna represente a un vendedor específico y cada fi la

represente a un producto. Saque el total de cada fi la para obtener las ventas totales de cada producto durante el último mes. Saque el total de cada columna para obtener las ventas totales de cada vendedor durante el último mes. Su impresión tabular debe incluir estos totales cruzados a la derecha de las totalizadas, y en la parte inferior de las columnas totalizadas.

II – CLASES Y OBJETOS

1. ¿Cómo se representa una clase en JAVA?
2. ¿Cómo se declara un objeto en JAVA?
3. ¿Qué es una instancia en programación orientada objetos?
4. ¿Cuál es la función del Constructor?
5. Menciona el Tipo de Clase que tiene al menos un método abstracto.
6. Menciona el Tipo de Clase que es accesible desde otras clases bien sea directamente o por herencia.
7. ¿A través de cual clase están reguladas la entrada desde teclado y la salida a pantalla?
8. ¿A cuál package pertenece la clase System?
9. Explique la noción del acceso a nivel de paquete en Java. Explique los aspectos negativos del acceso a nivel de paquete.
10. ¿Qué ocurre cuando un tipo de valor de retorno, incluso void, se especifica para un constructor?
11. ¿Qué es una interfaz?
12. (Clase Rectangulo) Cree una clase llamada Rectangulo. La clase debe tener los atributos longitud y anchura, cada uno con un valor predeterminado de 1. Debe tener métodos para calcular el perímetro y el área del rectángulo. Debe tener métodos establecer y obtener para longitud y anchura. Los métodos establecer deben verificar que longitud y anchura sean números de punto flotante mayores de 0.0, y menores de 20.0. Escriba un programa para probar la clase Rectangulo.
13. (Clase Entero Enorme) Cree una clase llamada EnteroEnorme que utilice un arreglo de 40 elementos de dígitos, para guardar enteros de hasta 40 dígitos de longitud cada uno. Proporcione los métodos entrada, salida, sumar y restar. Para comparar objetos EnteroEnorme, proporcione los siguientes métodos: esIgualA, noEsIgualA, esMayorQue, esMenorQue, esMayorOIgualA, y esMenorOIgualA. Cada uno de estos métodos deberá ser un método predicado que devuelva true si la relación se aplica entre los dos objetos EnteroEnorme, y false si no se aplica. Proporcione un método predicado llamado esCero. Si desea hacer algo más, proporcione también los métodos multiplicar, dividir y residuo. [Nota: los valores boolean primitivos pueden imprimirse como la palabra “true” o la palabra “false”, con el especificador de formato %b].
14. (Clase Fecha) Cree la clase Fecha con las siguientes capacidades:

a) Imprimir la fecha en varios formatos, como

MM/DD/AAAA

Junio 15, 1992

DDD AAAA

b) Usar constructores sobrecargados para crear objetos Fecha inicializados con fechas de los formatos en la parte (a). En el primer caso, el constructor debe recibir tres valores

enteros. En el segundo caso, debe recibir un objeto String y dos valores enteros. En el tercer caso debe recibir dos valores enteros, el primero de los cuales representa el número de día en el año. [Sugerencia: para convertir la representación de cadena del mes a un valor numérico, compare las cadenas usando el método equals. Por ejemplo, si s1 y s2 son cadenas, la llamada al método s1.equals(s2) devuelve true si las cadenas son idénticas y devuelve false en cualquier otro caso].

15. Haz una clase llamada **Persona** que siga las siguientes condiciones:

- Sus atributos son: **nombre, edad, DNI, sexo** (H hombre, M mujer)**, peso y altura.** No queremos que se accedan directamente a ellos. Piensa que modificador de acceso es el más adecuado, también su tipo. Si quieres añadir algún atributo puedes hacerlo.
- Por defecto, todos los atributos menos el DNI serán valores por defecto según su tipo (0 números, cadena vacía para String, etc.). Sexo será hombre por defecto, usa una constante para ello.
- Se implantarán varios constructores:
    - Un constructor por defecto.
    - Un constructor con el nombre, edad y sexo, el resto por defecto.
    - Un constructor con todos los atributos como parámetro.
- Los métodos que se implementaran son:
    - **calcularIMC()**: calculara si la persona está en su peso ideal (peso en kg/(altura^2 en m)), si esta fórmula devuelve un valor menor que 20, la función devuelve un -1, si devuelve un número entre 20 y 25 (incluidos), significa que está por debajo de su peso ideal la función devuelve un 0 y si devuelve un valor mayor que 25 significa que tiene sobrepeso, la función devuelve un 1. Te recomiendo que uses constantes para devolver estos valores.
        - **esMayorDeEdad()**: indica si es mayor de edad, devuelve un booleano.
        - **comprobarSexo(char sexo)**: comprueba que el sexo introducido es correcto. Si no es correcto, será H. No será visible al exterior.
        - **toString()**: devuelve toda la información del objeto.
        - **generaDNI()**: genera un número aleatorio de 8 cifras, genera a partir de este su número su letra correspondiente. Este método será

invocado cuando se construya el objeto. Puedes dividir el método para que te sea más fácil. No será visible al exterior.

- Métodos set de cada parámetro, excepto de DNI.

Ahora, crea una clase ejecutable que haga lo siguiente:

- Pide por teclado el nombre, la edad, sexo, peso y altura.
- Crea 3 objetos de la clase anterior, el primer objeto obtendrá las anteriores variables pedidas por teclado, el segundo objeto obtendrá todos los anteriores menos el peso y la altura y el último por defecto, para este último utiliza los métodos set para darle a los atributos un valor.
- Para cada objeto, deberá comprobar si está en su peso ideal, tiene sobrepeso o por debajo de su peso ideal con un mensaje.
- Indicar para cada objeto si es mayor de edad.
- Por último, mostrar la información de cada objeto.

Puedes usar métodos en la clase ejecutable, para que sea más fácil.

III – ENCAPSULAMIENTO

1. ¿Qué es encapsulamiento?

2. Menciona ventajas del encapsulamiento.

3. ¿Qué es la abstracción en la programación orientada a objetos? También menciona un ejemplo.

4. Menciona las diferencias entre abstracción y encapsulamiento.

5. Explique la noción del acceso a nivel de paquete en Java. Explique los aspectos negativos del acceso a nivel de paquete.

6. (Números racionales) Cree una clase llamada Racional para realizar operaciones aritméticas con fracciones. Escriba un programa para probar su clase. Use variables enteras para representar las variables de instancia private de la clase: el numerador y el denominador. Proporcione un constructor que permita a un objeto de esta clase inicializarse al ser declarado. El constructor debe almacenar la fracción en forma reducida. La fracción 2/4 es equivalente a 1/2 y debe guardarse en el objeto como 1 en el numerador y 2 en el denominador. Proporcione un constructor sin argumentos con valores predeterminados, en caso de que no se proporcionen inicializadores. Proporcione métodos public que realicen cada una de las siguientes operaciones:

a) Sumar dos números Racional: el resultado de la suma debe almacenarse en forma reducida.

b) Restar dos números Racional: el resultado de la resta debe almacenarse en forma reducida.

c) Multiplicar dos números Racional: el resultado de la multiplicación debe almacenarse en forma reducida.

d) Dividir dos números Racional: el resultado de la división debe almacenarse en forma reducida.

e) Imprimir números Racional en la forma a/b, en donde a es el numerador y b es el denominador.

f) Imprimir números Racional en formato de punto flotante. (Considere proporcionar capacidades de formato, que permita seleccionar la cantidad de decimales)

7. Cree la clase Docente dentro de un paquete demo que tenga como atributos privados: nombre, apellido, tipo (ciencias y letras), horas.

Incluye:

- Un constructor público que inicialice a todos los atributos.

- Métodos de acceso: set/get para los atributos.

- Un método que retorne el nombre completo (nombre y apellido)

- Un método que retorne el suelto Bruto (“ciencia”=S/3 , “letras”= S/5 por horas)

- Un método que retorne el descuento del 0.10

- Un método que retorne el sueldo neto

8. Algunos programadores prefieren no utilizar el acceso protected, pues piensan que quebranta el encapsulamiento de la superclase. Hable sobre los méritos relativos de utilizar el acceso protected, en comparación con el acceso private en las superclases.

IV – HERENCIA

1. Diferencias entre herencia simple y múltiple.

2. Defina los modificadores de acceso public, protected y private.

3. Explique para que sirven la palabras reservada super().

4. Describa las formas en las que la herencia fomenta la reutilización de software, ahorra tiempo durante el desarrollo de los programas y ayuda a prevenir errores.

5. Dibuje una jerarquía de herencia para los estudiantes en una universidad, de manera similar a la jerarquía que se muestra en la figura 9.2. Use a Estudiante como la superclase de la jerarquía, y después extienda Estudiante con las clases EstudianteNoGraduado y EstudianteGraduado. Siga extendiendo la jerarquía con el mayor número de niveles que sea posible. Por ejemplo, EstudiantePrimerAnio, EstudianteSegundoAnio, EstudianteTercerAnio y EstudianteCuartoAnio podrían extender a EstudianteNoGraduado, y EstudianteDoctorado y EstudianteMaestria podrían ser subclases de EstudianteGraduado. Después de dibujar la jerarquía, hable sobre las relaciones que existen entre las clases. [Nota: no necesita escribir código para este ejercicio].

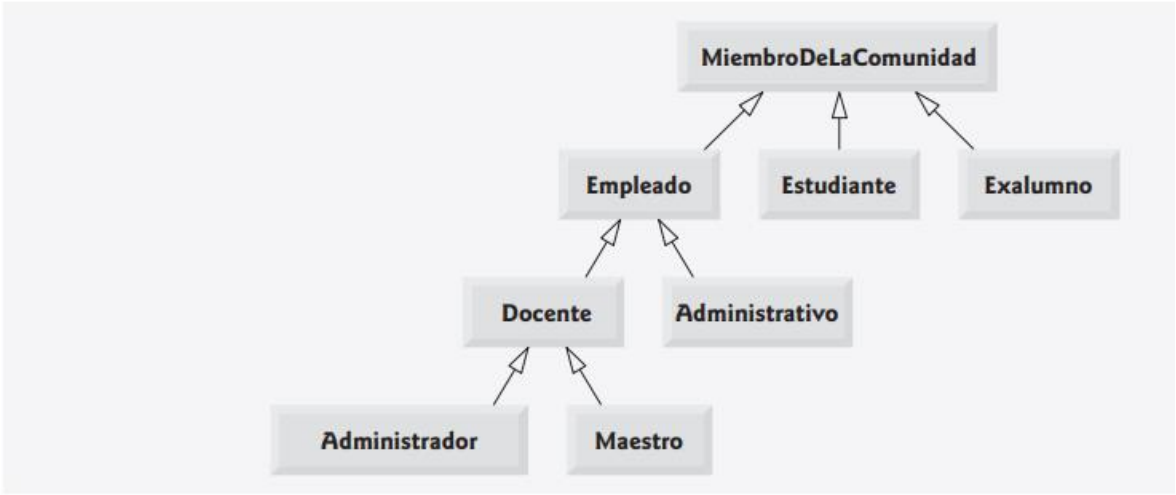
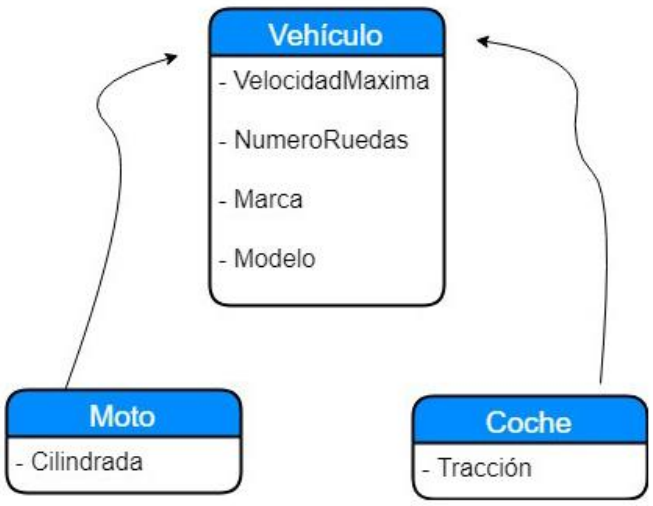


Figura 9.2 | Jerarquía de herencia para objetos `MiembroDeLaComunidad` universitaria.

6. Escriba una jerarquía de herencia para las clases Cuadrilatero, Trapezoide, Paralelogramo, Rectangulo y Cuadrado. Use Cuadrilatero como la superclase de la jerarquía. Agregue todos los niveles que sea posible a la jerarquía. Especifique las variables de

instancia y los métodos para cada clase. Las variables de instancia private de Cuadrilatero deben ser los pares de coordenadas x-y para los cuatro puntos finales del Cuadrilatero. Escriba un programa que cree instancias de objetos de sus clases, y que imprima el área de cada objeto (excepto Cuadrilatero).

7. Realice el código que representa el siguiente diagrama UML.



V – POLIMORFISMO

1. Una subclase puede heredar la “interfaz” o “implementación” de una superclase. ¿En qué difieren las jerarquías de herencia diseñadas para heredar la interfaz, de las jerarquías diseñadas para heredar la implementación?
2. ¿Qué son los métodos abstractos? Describa las circunstancias en las que un método abstracto sería apropiado.
3. (Jerarquía de figuras) Implemente la jerarquía Figura que se muestra en la figura 9.3. Cada FiguraBidimensional debe contener el método obtenerArea para calcular el área de la figura bidimensional. Cada FiguraTridimensional debe tener los métodos obtenerArea y obtenerVolumen para calcular el área superficial y el volumen, respectivamente, de la figura tridimensional. Cree un programa que utilice un arreglo de referencias Figura a objetos de cada clase concreta en la jerarquía. El programa deberá imprimir una descripción de texto del objeto al cual se refiere cada elemento del arreglo. Además, en el ciclo que procesa a todas las figuras en el arreglo, determine si cada figura es FiguraBidimensional o FiguraTridimensional. Si es FiguraBidimensional, muestre su área. Si es FiguraTridimensional, muestre su área y su volumen.

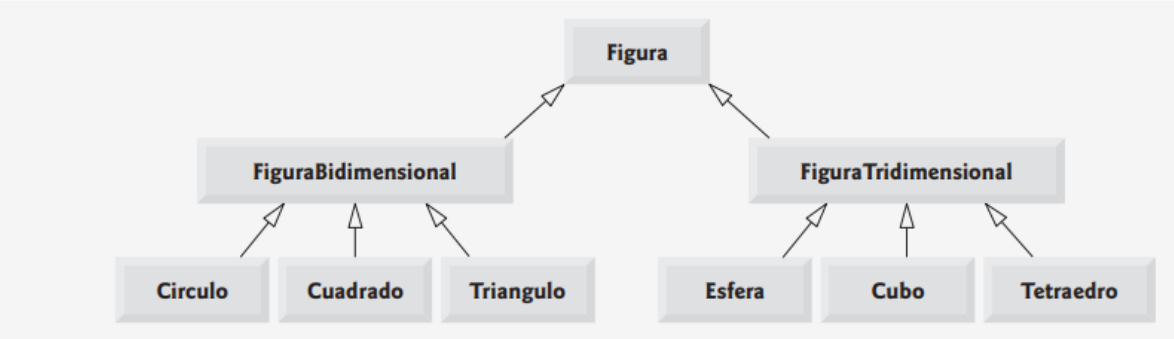


Figura 9.3 | Jerarquía de herencia para `Figuras`.

4. Realice dos clases Perro y Gato, que van a heredar de la superclase Animal. La superclase tiene un método abstracto makeSound() que se debe de implementar de forma distinta en las subclases.

VI – EXCEPCIONES

1. Hasta este capítulo, hemos visto que tratar con los errores detectados por los constructores es algo difícil. Explique por qué el manejo de excepciones es un medio efectivo para tratar con las fallas en los constructores.

2. (Atrapar excepciones con las superclases) Use la herencia para crear una superclase de excepción (llamada ExcepcionA) y las subclases de excepción ExcepcionB y ExcepcionC, en donde ExcepcionB hereda de ExcepcionA y ExcepcionC hereda de ExcepcionB. Escriba un programa para demostrar que el bloque catch para el tipo ExcepcionA atrapa excepciones de los tipos ExcepcionB y ExcepcionC.

3. (Orden de los bloques catch) Escriba un programa que demuestre que el orden de los bloques catch es importante. Si trata de atrapar un tipo de excepción de superclase antes de un tipo de subclase, el compilador debe generar errores.

4. (Falla del constructor) Escribe un programa que muestre cómo un constructor pasa información sobre la falla del constructor a un manejador de excepciones. Defina la clase unaExcepcion, que lanza una excepción Exception en el constructor. Su programa deberá tratar de crear un objeto de tipo UnaExcepcion y atrapar la excepción que se lance desde el constructor.

5. (Volver a lanzar expresiones) Escriba un programa que ilustre cómo volver a lanzar una excepción. Defina los métodos unMetodo y unMetodo2. El método unMetodo2 debe lanzar al principio una excepción. El método unMetodo debe llamar a unMetodo2, atrapar la excepción y volver a lanzarla. Llame a unMetodo desde el método main, y atrape la excepción que se volvió a lanzar. Imprima el rastreo de la pila de esta excepción.

6. Realice una excepción personalizada en el lenguaje de programación de su preferencia.

VII – ARCHIVOS GENERICOS Y COLECCIONES

1. Encuentre el error en cada uno de los siguientes bloques de código y muestre cómo corregirlo.

a) Suponga que se declaran cuenta, compania y monto.

```
ObjectOutputStream flujoSalida;

flujoSalida.writeInt( cuenta );

flujoSalida.writeChars( compania );

flujoSalida.writeDouble(monto);
```

b) Las siguientes instrucciones deben leer un registro del archivo "porpagar.txt". Se debe utilizar la variable entPorPagar de Scanner para hacer referencia a este archivo.

```
Scanner entPorPagar = new Scanner( new File( "porpagar.txt") );

RegistroPorPagar registro = ( RegistroPorPagar ) entPorPagar.readObject();
```

2. (Asociación de archivos) En un sistema de cuentas por cobrar hay generalmente un archivo maestro, el cual contiene información detallada acerca de cada cliente, como su nombre, dirección, número telefónico, saldo deudor, límite de crédito, términos de descuento, acuerdos contractuales y posiblemente un historial condensado de compras recientes y pagos en efectivo. A medida que ocurren las transacciones (es decir, a medida que se generan las ventas y llegan los pagos en el correo), la información acerca de ellas se introduce en un archivo. Al fi nal de cada periodo de negocios (un mes para algunas compañías, una semana para otras y un día en algunos casos), el archivo de transacciones (llamado "trans.txt") se aplica al archivo maestro (llamado "antmaest.txt") para actualizar el registro de compras y pagos de cada cuenta. Durante una actualización, el archivo maestro se rescribe como el archivo "nuevomaest.txt", el cual se utiliza al final del siguiente periodo de negocios para empezar de nuevo el proceso de actualización. Los programas para asociar archivos deben tratar con ciertos problemas que no existen en programas de un solo archivo.

Escriba un programa completo para asociar archivos de cuentas por cobrar. Utilice el número de cuenta en cada archivo como la clave de registro para fines de asociar los archivos. Suponga que cada archivo es un archivo de texto secuencial con registros almacenados en orden ascendente, por número de cuenta.

a) Defina la clase RegistroTransaccion. Los objetos de esta clase contienen un número de cuenta y un monto para la transacción. Proporcione métodos para modificar y obtener estos valores.

b) Modifique la clase RegistroCuenta de la figura 14.6 para incluir el método combinar, el cual recibe un objeto RegistroTransaccion y combina el saldo del objeto RegistroCuenta con el valor del monto del objeto RegistroTransaccion.

```
1  // Fig. 14.6: RegistroCuenta.java
2  // Una clase que representa un registro de información
3  package com.deitel.jhtp7.cap14; // se empaqueta para reutilizarla
4
5  public class RegistroCuenta
6  {
7     private int cuenta;
8     private String primerNombre;
9     private String apellidoPaterno;
10    private double saldo;
11
12    // el constructor sin argumentos llama a otro constructor con valores predeterminados
13    public RegistroCuenta()
14    {
15       this( 0, "", "", 0.0 ); // llama al constructor con cuatro argumentos
16    } // fin del constructor de RegistroCuenta sin argumentos
17
18    // inicializa un registro
19    public RegistroCuenta( int cta, String nombre, String apellido, double sal )
20    {
21       establecerCuenta( cta );
22       establecerPrimerNombre( nombre );
23       establecerApellidoPaterno( apellido );
24       establecerSaldo( sal );
25    } // fin del constructor de RegistroCuenta con cuatro argumentos
26
27    // establece el número de cuenta
28    public void establecerCuenta( int cta )
29    {
30       cuenta = cta;
31    } // fin del método establecerCuenta
32
33    // obtiene el número de cuenta
34    public int obtenerCuenta()
35    {
36       return cuenta;
37    } // fin del método obtenerCuenta
38
39    // establece el primer nombre
40    public void establecerPrimerNombre( String nombre )
41    {
42       primerNombre = nombre;
43    } // fin del método establecerPrimerNombre
44
45    // obtiene el primer nombre
46    public String obtenerPrimerNombre()
47    {
48       return primerNombre;
49    } // fin del método obtenerPrimerNombre
50
51    // establece el apellido paterno
52    public void establecerApellidoPaterno( String apellido )
53    {
54       apellidoPaterno = apellido;
55    } // fin del método establecerApellidoPaterno
56
57    // obtiene el apellido paterno
58    public String obtenerApellidoPaterno()
59    {
```

Figura 14.6 | RegistroCuenta mantiene la información para una cuenta. (Parte 1 de 2).

```
60          return apellidoPaterno;
61       } // fin del método obtenerApellidoPaterno
62
63       // establece el saldo
64       public void establecerSaldo( double sal )
65       {
66          saldo = sal;
67       } // fin del método establecerSaldo
68
69       // obtiene el saldo
70       public double obtenerSaldo()
71       {
72          return saldo;
73       } // fin del método obtenerSaldo
74    } // fin de la clase RegistroCuenta
```

Figura 14.6 | `RegistroCuenta` mantiene la información para una cuenta. (Parte 2 de 2).

c) Escriba un programa para crear datos de prueba para el programa. Use los datos de la cuenta de ejemplo de las figuras 14.24 y 14.25. Ejecute el programa para crear los archivos trans.txt y antmaest.txt, para que los utilice su programa de asociación de archivos.

Número de cuenta	Nombre	Saldo
100	Alan Jones	348.17
300	Mary Smith	27.19
500	Sam Sharp	0.00
700	Susy Green	-14.22

Figura 14.24 | Datos de ejemplo para el archivo maestro.

Archivo de transacciones Número de cuenta	Monto de la transacción
100	27.14
300	62.11
400	100.56
900	82.17

Figura 14.25 | Datos de ejemplo para el archivo de transacciones.

d) Cree la clase AsociarArchivos para llevar a cabo la funcionalidad de asociación de archivos. La clase debe contener métodos para leer antmaest.txt y trans.txt. Cuando ocurra una coincidencia (es decir, que aparezcan registros con el mismo número de cuenta en el archivo maestro y en el archivo de transacciones), sume el monto en dólares del registro de transacciones al saldo actual en el registro maestro, y escriba el registro "nuevomaest.txt". (Suponga que las compras se indican mediante montos positivos en el archivo de transacciones, y los pagos mediante montos negativos). Cuando haya un registro maestro para una cuenta específica, pero no haya un registro de transacciones correspondiente, simplemente escriba el registro maestro en "nuevomaest.txt". Cuando haya un registro de transacciones, pero no un registro maestro correspondiente, imprima en un archivo de registro el mensaje "Hay un registro de transacciones no asociado para ese número de cliente..." (utilice el número de cuenta del registro de transacciones). El archivo de registro debe ser un archivo de texto llamado "registro.txt".

3. Explique el uso de la siguiente notación en un programa en Java:

public class Array< T > { }

4. Escriba una clase genérica llamada Par, que tenga dos parámetros de tipo: F y S, cada uno de los cuales representa el tipo del primer y segundo elementos del par, respectivamente. Agregue métodos obtener y establecer para los elementos primero y segundo del par. [Sugerencia: el encabezado de la clase debe ser public class Par< F, S >].

5. Sobrecargue el método genérico imprimirArreglo de la figura 18.3 con una versión no genérica que imprima en forma específica un arreglo de cadenas en un formato tabular impecable, como se muestra en los resultados de ejemplo a continuación:

El arreglo arregloCadena contiene:

uno dos tres cuatro

cinco seis siete ocho

```java
// Fig. 18.3: PruebaMetodoGenerico.java
// Uso de métodos genéricos para imprimir arreglos de distintos tipos.

public class PruebaMetodoGenerico
{
   // método genérico imprimirArreglo
   public static < E > void imprimirArreglo( E[] arregloEntrada )
   {
      // muestra los elementos del arreglo
      for ( E elemento : arregloEntrada )
         System.out.printf( "%s ", elemento );

      System.out.println();
   } // fin del método imprimirArreglo

   public static void main( String args[] )
   {
      // crea arreglos de objetos Integer, Double y Character
      Integer[] arregloInteger = { 1, 2, 3, 4, 5, 6 };
      Double[] arregloDouble = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7 };
      Character[] arregloCharacter = { 'H', 'O', 'L', 'A' };

      System.out.println( "El arreglo arregloInteger contiene:" );
      imprimirArreglo( arregloInteger ); // pasa un arreglo Integer
      System.out.println( "\nEl arreglo arregloDouble contiene:" );
      imprimirArreglo( arregloDouble ); // pasa un arreglo Double
      System.out.println( "\nEl arreglo arregloCharacter contiene:" );
      imprimirArreglo( arregloCharacter ); // pasa un arreglo Character
   } // fin de main
} // fin de la clase PruebaMetodoGenerico
```

```
El arreglo arregloInteger contiene:
1 2 3 4 5 6

El arreglo arregloDouble contiene:
1.1 2.2 3.3 4.4 5.5 6.6 7.7

El arreglo arregloCharacter contiene:
H O L A
```

Figura 18.3 | Impresión de los elementos de un arreglo, usando el método genérico `imprimirArreglo`.

6. ¿Cómo pueden sobrecargarse los métodos genéricos?

7. Explique por qué un programa en Java podría utilizar la instrucción

ArrayList< Empleado > listaTrabajadores = new ArrayList< Empleado >();

8. Defina cada uno de los siguientes términos:

a) Collection

b) Collections

c) Comparator

d) List

e) factor de carga

f) colisión

g) concesión entre espacio y tiempo en hashing

h) HashMap

9. Determine si cada uno de los siguientes enunciados es verdadero o falso. Si es falso, explique por qué.

a) Los elementos en un objeto Collection deben almacenarse en orden ascendente, antes de poder realizar una búsqueda binaria mediante binarySearch.

b) El método first obtiene el primer elemento en un objeto TreeSet.

c) Un objeto List creado con el método asList de Arrays puede cambiar su tamaño.

d) La clase Arrays proporciona el método static llamado sort para ordenar los elementos de un arreglo.

10. Explique la operación de cada uno de los siguientes métodos de la clase Properties:

a) load

b) store

c) getProperty

d) list

11. Escriba un programa que lea una serie de nombres de pila y los almacene en un objeto LinkedList. No almacene nombres duplicados. Permita al usuario buscar un nombre de pila

12. Escriba un programa que determine e imprima el número de palabras duplicadas en un enunciado. Trate a las letras mayúsculas y minúsculas de igual forma. Ignore los signos de puntuación.