Práctica Dirigida

Pregunta 1.

```
import java.util.Scanner;

public class NumerosAmigos {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Introduce el primer número:");
        int num1 = scanner.nextInt();
        System.out.println("Introduce el segundo número:");
        int num2 = scanner.nextInt();
        if (sonAmigos(num1, num2)) {
            System.out.println("Son amigos");
        } else {
            System.out.println("No son amigos");
        }
    }

    public static boolean sonAmigos(int num1, int num2) {
        int suma1 = 0;
        for (int i = 1; i < num1; i++) {
            if (num1 % i == 0) {
                suma1 += i;
            }
        }
        int suma2 = 0;
        for (int i = 1; i < num2; i++) {
            if (num2 % i == 0) {
                suma2 += i;
            }
        }
        return suma1 == num2 && suma2 == num1;
    }
}
```

Implemente las siguientes pruebas unitarias:

a) caso de prueba que verifica el funcionando correctamente cuando los números son amigos.

b) caso de prueba que verifica cuando los números no son amigos

c) caso de prueba que verifica que esté funcionando correctamente cuando los números son iguales.

d) caso de prueba, que verifica cuando los números son primos.

e) caso de prueba, que verifica cuando los números tienen repetidos.

Pregunta 2.

```java
public static void burbuja(int[] A) {
        int i, j, aux;
        for (i = 0; i < A.length - 1; i++) {
            for (j = 0; j < A.length - i - 1; j++) {
                if (A[j + 1] < A[j]) {
                    aux = A[j + 1];
                    A[j + 1] = A[j];
                    A[j] = aux;
                }
            }
        }
}
```

a) Muestre el grafo asociado y calcule la complejidad
b) Determine los caminos independientes
c) Generar los casos de prueba para cada camino independiente

Pregunta 3.

Se tiene una aplicación bancaria, donde el usuario puede conectarse al banco por Internet y realizar una serie de operaciones bancarias. Una vez accedido al banco con las consiguientes medidas de seguridad (clave de acceso y demás), la información de entrada del procedimiento que gestiona las operaciones concretas a realizar por el usuario requiere la siguiente entrada:

- **Código del banco.** En blanco o número de tres dígitos. En este último caso, el primero de los dígitos tiene que ser mayor que 1.
- **Código de sucursal**. Un número de cuatro dígitos. El primero de ellos mayor de 0.
- **Número de cuenta.** Número de cinco dígitos.
- **Clave personal**. Valor alfanumérico de cinco posiciones.
- **Orden**. Este valor se introducirá según la orden que se desee realizar. Puede estar en blanco o ser una de las dos cadenas siguientes,◦ “Talonario” o “Movimientos ”. En el primer caso el usuario recibirá un talonario de cheques, mientras que en el segundo recibirá los movimientos del mes en curso. Si este código está en blanco, el usuario recibirá los dos documentos.

Pregunta 4 .

El siguiente código Java proporcionado calcula el precio total de un pedido en función de la cantidad, el precio del artículo, el descuento y el envío.

```
public class CalculadoraPrecio {

 public static void main(String[] args) {

  double precio = precio(10, 5.0);
  System.out.println("Precio por 10 artículos a $5.00 cada uno: $" + precio);
 }

 public static double precio(int cantidad, double itemPrecio) {

  return cantidad * itemPrecio -
       Math.max(0, cantidad - 500) * itemPrecio * 0.05 +
       Math.min(cantidad * itemPrecio * 0.1, 100.0);
 }
}
```

Se solicita:

a) Justificación de la técnica de refactorización aplicada para eliminar el Bad Smell
b) Refactorice el código y verifique que pasa las pruebas unitarias