# BOLETÍN DE EJERCICIOS - PRUEBAS DE CAJA BLANCA

Para cada uno de los siguientes ejercicios, realiza:

1. Dibujo del grafo de flujo
2. Cálculo de la complejidad ciclomática usando los tres métodos revisados en los apuntes
3. Identificación de caminos independientes
4. Diseño de casos de prueba para conseguir una cobertura del 100%

## 1. EJERCICIO 1: Validador de Contraseñas

```
public static boolean validarPassword(String password) {
    if (password == null || password.length() < 8) {
        return false;
    }

    boolean tieneMayuscula = false;
    boolean tieneNumero = false;
    boolean tieneEspecial = false;
    boolean tieneMinuscula = false;

    for (char c : password.toCharArray()) {
        if (Character.isUpperCase(c)) {
            tieneMayuscula = true;
        } else if (Character.isDigit(c)) {
            tieneNumero = true;
        } else if (!Character.isLetterOrDigit(c)) {
            tieneEspecial = true;
        } else {
            tieneMinuscula = true;
        }

        if (tieneMayuscula && tieneNumero && tieneEspecial &&
tieneMinuscula) {
            return true;
        }
    }

    return false;
}
```

## 2. EJERCICIO 2: Calculador de Descuentos

```
public static double calcularDescuento(double precio, int unidades,
String tipoCliente) {
    if (precio <= 0 || unidades <= 0) {
        throw new IllegalArgumentException("Precio y unidades deben
ser positivos");
    }

    double descuento = 0;

    // Descuento por volumen
    if (unidades >= 10) {
        descuento += 0.10;
    }

    // Descuento por tipo de cliente
    if (tipoCliente != null) {
        if (tipoCliente.equals("VIP")) {
            descuento += 0.20;
        } else if (tipoCliente.equals("FRECUENTE")) {
            descuento += 0.10;
        }
    }

    return precio * unidades * (1 - descuento);
}
```

## 3. EJERCICIO 3: Validador de Fechas

```java
public static boolean esFechaValida(int dia, int mes, int año) {
    if (año < 1900 || año > 2100) {
        return false;
    }

    if (mes < 1 || mes > 12) {
        return false;
    }

    int[] diasPorMes = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
31};

    if (mes == 2 && (año % 4 == 0 && (año % 100 != 0 || año % 400 ==
0))) {
        return dia >= 1 && dia <= 29;
    }

    return dia >= 1 && dia <= diasPorMes[mes - 1];
}
```

## 4. EJERCICIO 4: Evaluador de Calificaciones

```java
public static String evaluarCalificacion(int[] notas) {
    if (notas == null || notas.length == 0) {
        throw new IllegalArgumentException("El array de notas no
puede estar vacío");
    }

    double suma = 0;
    for (int nota : notas) {
        if (nota < 0 || nota > 10) {
            throw new IllegalArgumentException("Las notas deben estar
entre 0 y 10");
        }
        suma += nota;
    }

    double promedio = suma / notas.length;

    if (promedio >= 9) {
        return "Sobresaliente";
    } else if (promedio >= 7) {
        return "Notable";
    } else if (promedio >= 5) {
        return "Aprobado";
    } else {
        return "Suspenso";
    }
}
```

## 5. EJERCICIO 5: Conversor de Números Romanos

```java
import java.util.HashMap;
import java.util.Map;

public static int romanoADecimal(String romano) {
    if (romano == null || romano.isEmpty()) {
        throw new IllegalArgumentException("El número romano no puede
estar vacío");
    }

    Map<Character, Integer> valores = new HashMap<>();
    valores.put('I', 1);
    valores.put('V', 5);
    valores.put('X', 10);
    valores.put('L', 50);

    int resultado = 0;
    int anterior = 0;

    for (int i = romano.length() - 1; i >= 0; i--) {
        char c = romano.charAt(i);
        if (!valores.containsKey(c)) {
            throw new IllegalArgumentException("Carácter romano
inválido: " + c);
        }

        int actual = valores.get(c);
        if (actual >= anterior) {
            resultado += actual;
        } else {
            resultado -= actual;
        }
        anterior = actual;
    }

    return resultado;
}
```

## 6. EJERCICIO 6: Calculadora de cuotas mensuales

```java
import java.text.DecimalFormat;

public static double calcularCuotaMensual(double capital, double
tipoInteres, int plazoMeses) {
    if (capital <= 0) {
        throw new IllegalArgumentException("El capital del préstamo
debe ser positivo");
    }
    if (tipoInteres < 0 || tipoInteres > 100) {
        throw new IllegalArgumentException("El tipo de interés debe
estar entre 0 y 100");
    }
    if (plazoMeses <= 0) {
        throw new IllegalArgumentException("El plazo debe ser
positivo");
    }

    // Convertir tipo de interés anual a mensual
    double tipoInteresMensual = (tipoInteres / 100) / 12;

    // Fórmula de cuota mensual
    double cuota;
    if (tipoInteresMensual == 0) {
        cuota = capital / plazoMeses;
    } else {
        cuota = (capital * tipoInteresMensual * Math.pow(1 +
tipoInteresMensual, plazoMeses))
                / (Math.pow(1 + tipoInteresMensual, plazoMeses) - 1);
    }

    // Redondear a 2 decimales
    DecimalFormat df = new DecimalFormat("#.##");
    return Double.parseDouble(df.format(cuota).replace(',', '.'));
}
```