

1.Desarrolla una tabla comparativa de las ventajas y desventajas de la reutilización de código. Incluye al menos 3 de cada una.

Ventajas	Desventajas
<ul style="list-style-type: none">• Mayor productividad. Debido a la facilidad con la que se puede reutilizar el mismo bloque de código, los desarrolladores son capaces de terminar proyectos más rápidamente sin comprometer la calidad del producto final	<ul style="list-style-type: none">• Ambigüedad del método a reutilizar.
<ul style="list-style-type: none">• Facilitar la compartición de productos del ciclo de vida.	<ul style="list-style-type: none">• Falta de documentación o mal escrito
<ul style="list-style-type: none">• Menor riesgo. Reutilizar el mismo bloque de código significa menor riesgo de errores o fallas en los programas, ya que se ha comprobado muchas veces antes y se conocen sus resultados.	<ul style="list-style-type: none">• El código es específico para un entorno o plataforma en particular
<ul style="list-style-type: none">• Dependencia reducida: La creación de su propio código reutilizable también puede reducir la dependencia de desarrolladores, <u>bibliotecas</u> o plataformas específicas	

2. Describe las características de los depuradores y cómo se utilizan para encontrar errores en el software.

Un depurador es un programa que permite detectar y diagnosticar fallos en programas informáticos. El objetivo de estas herramientas es garantizar, a largo plazo, que el software funcione en todos los dispositivos y plataformas para los que está pensado.

Características:

-Recorrido del Código: Un depurador permite a los desarrolladores ejecutar un programa línea por línea, permitiéndoles observar cambios en variables y en el flujo de control.

-Inspección de Variables: Los desarrolladores pueden inspeccionar los valores de variables y objetos en cualquier punto durante la ejecución del programa, lo que les permite identificar valores incorrectos o comportamientos no deseados.

-Establecimiento de Puntos de Interrupción: Los puntos de interrupción son ubicaciones específicas en el código donde el depurador pausa la ejecución, permitiendo a los desarrolladores analizar el estado del programa y el flujo de ejecución hasta ese punto.

-Examinación de la Pila de Llamadas: La pila de llamadas realiza un seguimiento de la secuencia de llamadas de función que llevaron al punto actual de la ejecución del

programa. Un depurador permite a los desarrolladores examinar la pila de llamadas y trazar la secuencia de estas, lo cual puede ayudar a identificar la causa raíz de un error.

Paso a paso:

1. Establecer Puntos de Control (Breakpoints):
 - Identifica los puntos críticos en el código y establece puntos de control (breakpoints) en esos lugares para detener la ejecución y examinar el estado del programa.
2. Inspeccionar Variables:
 - Utiliza la capacidad de inspección de variables para revisar los valores de las variables en diferentes puntos del código, especialmente cerca de donde se produce el error.
3. Ejecutar Paso a Paso:
 - Ejecuta el programa paso a paso para observar cómo fluye la ejecución y para identificar el punto exacto donde ocurre el error.
4. Rastrear la Pila de Llamadas:
 - Sigue la pila de llamadas para entender la secuencia de funciones o métodos que llevaron al programa al punto actual. Esto puede ayudar a identificar la causa subyacente.
5. Utilizar Consola o Ventana de Depuración:
 - Emplea la consola de depuración o ventana de salida para obtener información adicional y ejecutar comandos durante la depuración.
6. Identificar Excepciones:
 - Busca excepciones o errores y comprende la naturaleza de los problemas que se están produciendo.
7. Corregir el Código:
 - Una vez identificado el problema, realiza las correcciones necesarias en el código.
8. Pruebas Adicionales:
 - Después de hacer cambios, vuelve a ejecutar el programa y realiza pruebas adicionales para asegurarte de que el error se haya corregido sin introducir nuevos problemas.

Apartado A

```
public class ParImpar {  
    public static void main(String[] args) {  
        int numero = 5;  
  
        if (numero % 2 = 0) {  
            System.out.println("El número es impar");  
        } else {  
            System.out.println("El número es par");  
        }  
    }  
}
```

Primer error:

% 2 = 0 , en lugar de % 2 == 0 , hay que poner == no =

Error 2:

If (numero % 2== 0){

System.out.println("El número es par") no impar como está escrito

Error 3:

Else {

System.out.println("El número es impar") no par como está escrito

Código bien escrito:

```
public class ParImpar {  
    public static void main(String[] args) {  
        int numero = 5;  
        if (numero % 2 == 0) {  
            System.out.println("El número es par");  
        } else {  
            System.out.println("El número es impar");  
        }  
    }  
}
```

Explicación:

Este código sirve para ver si un número es par o impar, para ello se divide entre 2, si el resto es 0 es par en caso contrario es impar. En el ejemplo pone el número 5.

Apartado B

```
public class Saludo {  
    public static void main(String[] args) {  
        String mensaje = "Hola mundo!"  
        System.out.println()  
    }  
}
```

Error:

System.out.println() , el campo a imprimir () está vacío, por tanto no imprime nada, hay que poner (mensaje)

Código corregido:

```
public class Saludo {  
    public static void main(String[] args) {  
        String mensaje = "Hola mundo!";  
        System.out.println(mensaje);  
    }  
}
```

Explicación:

El código sirve para que en la pantalla aparezca Hola mundo!

Apartado C

```
public class Edad {  
    public static void main(String[] args) {  
        int edad == 18;  
  
        if (edad < 18) {
```

```
        System.out.println("Eres mayor de edad")  
    } else {  
        System.out.println("Eres menor de edad")  
    }  
}
```

Primer error:

int edad == 18 en lugar de int edad = 18 ; el operador == se utiliza para comparar si dos valores son iguales, mientras que el operador = se utiliza para asignar un valor a una variable.

Segundo error:

```
if (edad < 18) {
```

```
    System.out.println("Eres mayor de edad")
```

En lugar de: if (edad < 18) {

```
    System.out.println("Eres menor de edad")
```

Tercer error:

```
} else {
```

```
    System.out.println("Eres menor de edad")
```

En lugar de : } else {

```
    System.out.println("Eres mayor de edad");
```

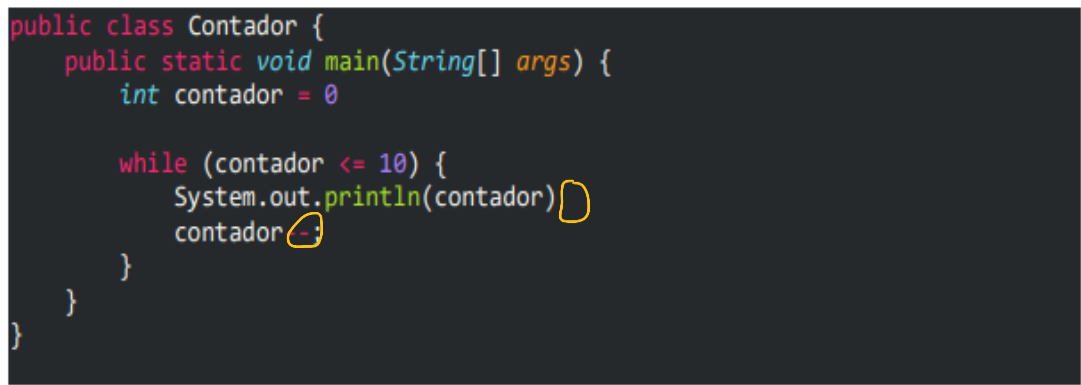
Código Corregido:

```
public class Edad {  
  
    public static void main(String[] args) {  
  
        int edad = 18;  
  
        if (edad < 18) {  
  
            System.out.println("Eres menor de edad");  
  
        } else {  
  
            System.out.println("Eres mayor de edad");  
  
        }  
  
    }  
  
}
```

Explicación:

Se introduce una edad, si la edad es menos de 18 sale que es menor de edad, si la edad es igual o mayor a 18 sale que es mayor de edad, es este caso edad=18 por lo que saldrá que es mayor de edad.

Apartado D



```
public class Contador {  
    public static void main(String[] args) {  
        int contador = 0  
  
        while (contador <= 10) {  
            System.out.println(contador) D  
            contador --;  
        }  
    }  
}
```

Primer error:

System.out.println(contador) en lugar de System.out.println(contador); falta el ;

Segundo error:

Contador--; en lugar de contador++; si ponemos – en lugar de + el contador nunca se incrementa.

Código Corregido:

```
public class Contador {  
    public static void main(String[] args) {  
        int contador = 0;  
        while (contador <= 18) {  
            System.out.println(contador);  
            contador++;  
        }  
    }  
}
```

Explicación:

Este código mostrará en la pantalla los números del 0 al 18, uno en cada línea, cada vez que muestre un número el contador sumará 1, así hasta que llegue a 19 y se pare.

Apartado E

```
public class Calculadora {  
    public static void main(String[] args) {  
        int resultado = sumar(5);  
        System.out.println("El resultado es: " + resultado);  
    }  
  
    public static int sumar(int num1, int num2) {  
        int suma = num1 - num2;  
        return suma;  
    }  
}
```

Primer error:

Int resultado = sumar(5) en lugar de poner 2 valores como por ejemplo

Int resultado=sumar(5,1)

Segundo error:

Int suma = num1 – num2 en lugar de Int suma = num1 + num2 porque en lugar de sumar se estaba restando

Código Corregido:

```
public class Calculadora {  
    public static void main(String[] args) {  
        int resultado = sumar(5, 1);  
        System.out.println("El resultado es: " + resultado);  
    }  
  
    public static int sumar(int num1, int num2) {  
        int suma = num1 + num2;  
        return suma;  
    }  
}
```

Explicación:

Se introducen 2 valores que se van a sumar y se muestra el resultado.

Apartado F

```
public class Calculadora {  
    public static void main(String[] args) {  
        int num1 = 10;  
        int num2 = 5;  
  
        String suma = num1 + num2  
  
        System.out.println("La suma es: " + suma)  
    }  
}
```

Primer error:

Convertir los números a cadenas usando String.valueOf()

Esto String suma = String.valueOf(num1) + String.valueOf(num2);

En lugar de lo que pone String suma = num1 + num2

Segundo error:

Hay que concatenar directamente con una cadena aquí System.out.println("La suma es:" + suma) porque no se pueden poner 2 valores, por lo que hay que poner esto

String suma = "La suma es: " + (num1 + num2);

System.out.println(suma)

Para mostrar la suma

Código Corregido:

```
public class Calculadora {  
    public static void main(String[] args) {  
        int num1 = 10;  
        int num2 = 5;  
  
        String suma = String.valueOf(num1) + String.valueOf(num2);  
  
        String suma = "La suma es: " + (num1 + num2);  
  
        System.out.println(suma);  
    }  
}
```

Explicación:

Con este código se introducen 2 números enteros y se hace su suma

Apartado G

```
public class Calculadora {  
    public static void main(String[] args) {  
        int num1 = 10;  
        int num2 = 0;  
  
        int suma = num1 - num2;  
        int resta = num1 - num2;  
        int multiplicacion = num1 * num2;  
        int division = num1 / num2;  
  
        System.out.println("La suma es: " + suma);  
        System.out.println("La resta es: " + resta);  
        System.out.println("La multiplicación es: " + multi);  
        System.out.println("La división es: " + division)  
    }  
}
```

Primer error:

Int suma = num1 – num2 ; en lugar de Int suma = num1 + num2 cambiar el signo – por el + porque es una suma

Segundo error:

Cambiar int división = num1/num2 por

int division;

if (num2 != 0)

Para verificar si el divisor es cero.

Evitar la división por cero: Si num2 es cero, se asigna un valor predeterminado a la variable división.

division = num1 / num2;

} else {

division = 0;

Código Corregido:

```
public class Calculadora {  
    public static void main(String[] args) {  
        int num1 = 10;  
        int num2 = 0;
```

```
int suma = num1 + num2;

int resta = num1 - num2;

int multiplicacion = num1 * num2;


int division;

if (num2 != 0) {
    division = num1 / num2;
} else {
    division = 0;
}


System.out.println("La suma es: " + suma);
System.out.println("La resta es: " + resta);
System.out.println("La multiplicación es: " + multiplicacion);
System.out.println("La división es: " + division);
}
}
```

Explicación:

Este código crea una calculadora que suma, resta, multiplica y divide.