Explicacion del codigo de la solucion

```java
import java.util.Scanner;

public class AzulVerde {

	public static void main(String[] args) {

		Scanner teclado = new Scanner( System.in );
		int iEntUno = 0;
		int iEntDos = 0;
		int iEntTres = 0;
		String sEnt= "";
		String sSalida = "";
		String [] sDat;

		System.out.print("Introduce el numero de las tres entradas: ");
		sEnt = teclado.nextLine();
		sDat = sEnt.split(" ");

		iEntUno = Integer.parseInt(sDat[0]);
		iEntDos = Integer.parseInt(sDat[1]);
		iEntTres = Integer.parseInt(sDat[2]);

		sSalida = (iEntUno % 2) == 0 ?  sSalida + "VERDE" : sSalida + "AZUL";
		sSalida = (iEntDos % 2) == 0 ?  sSalida + " VERDE" : sSalida + " AZUL";
		sSalida = (iEntTres % 2) == 0 ?  sSalida + " VERDE" : sSalida + " AZUL";

		System.out.println(sSalida);

		teclado.close();

	}	// Fin de la funcion main
}
```

Explicación cada línea del código de java detalladamente:

Línea 1:
Importa la clase `Scanner` del paquete `java.util`, que se utiliza para obtener la entrada del usuario desde la consola.

```java
import java.util.Scanner;
```

Línea 3:
Declara una clase pública llamada `AzulVerde`. En Java, todo el código debe estar contenido dentro de una clase.

```java
public class AzulVerde {
```

Línea 5:
Define el método `main`, que es el punto de entrada de cualquier aplicación Java. Es `public` para que sea accesible desde fuera de la clase, `static` para que pueda ser ejecutado sin crear una instancia de la clase, y acepta un arreglo de `String` como argumentos de línea de comandos.

```java
	public static void main(String[] args) {
```

Línea 7:
Crea una instancia de `Scanner` llamada `teclado` que lee la entrada del usuario desde la entrada estándar (generalmente el teclado).

```java
		Scanner teclado = new Scanner( System.in );
```

Líneas 8-13:
Declara y, en su caso, inicializa varias variables:

- `iEntUno`, `iEntDos`, `iEntTres`: variables *enteras* que almacenarán los tres números ingresados por el usuario.
- `sEnt`: una *cadena* que almacenará la entrada completa del usuario.
- `sSalida`: una *cadena* que se utilizará para construir la salida final.
- `sDat`: un *arreglo de cadenas* que contendrá los números separados.

```java
		int iEntUno = 0;
		int iEntDos = 0;
		int iEntTres = 0;
		String sEnt= ""; //string
		String sSalida = ""; //string
		String [] sDat; //array de strings
```

Línea 15:
Muestra un mensaje en la consola solicitando al usuario que introduzca tres números separados por espacios.

```
                System.out.print("Introduce el numero de las tres entradas: ");
```

Línea 16:
Lee la línea completa de entrada del usuario y la almacena en la variable `sEnt`.

```
                // sEnt es una cadena que almacena la entrada completa del usuario
                sEnt = teclado.nextLine(); // Por ejemplo: "2 3 4"
```

Línea 17:
Divide la cadena `sEnt` en un arreglo de cadenas `sDat` utilizando el espacio como delimitador. Esto separa los tres números ingresados.

```
                // sDat es un arreglo de cadenas que contiene ["2", "3", "4"]
                sDat = sEnt.split(" ");
```

Nota:

> - **División de la Cadena:**

```
- `sDat = sEnt.split(" ");`
    El método `.split(" ")` divide la cadena `sEnt` en un arreglo de subcadenas (`sDat`) utilizando el espacio `" "` como delimitador.
```

> - **Resultado de la División:**

```
- Después de la división, `sDat` será un arreglo de cadenas: `["2", "3", "4"]`.
```

Líneas 19-21:
Convierte cada elemento del arreglo `sDat` de tipo `String` a `int` y los asigna a las variables correspondientes:

- `sDat[0]` → `iEntUno`
- `sDat[1]` → `iEntDos`
- `sDat[2]` → `iEntTres`

```
                iEntUno = Integer.parseInt(sDat[0]);
                iEntDos = Integer.parseInt(sDat[1]);
                iEntTres = Integer.parseInt(sDat[2]);
```

Líneas 23-25:
Para cada número ingresado (`iEntUno`, `iEntDos`, `iEntTres`), se verifica si es par o impar utilizando el operador módulo `%`:

- Si el número es par (`% 2 == 0`), se concatena `"VERDE"` a la cadena `sSalida`.
- Si el número es impar, se concatena `"AZUL"` a `sSalida`.
  El uso de `sSalida + "VERDE"` o `sSalida + " AZUL"` asegura que los resultados se separen por espacios después del primero.

```
                sSalida = (iEntUno % 2) == 0 ?  sSalida + "VERDE" : sSalida + "AZUL";
                sSalida = (iEntDos % 2) == 0 ?  sSalida + " VERDE" : sSalida + " AZUL";
                sSalida = (iEntTres % 2) == 0 ?  sSalida + " VERDE" : sSalida + " AZUL";
```

Línea 27:
Imprime la cadena `sSalida` resultante en la consola. Esta cadena contendrá una combinación de las palabras "VERDE" y "AZUL" según la paridad de los números ingresados.

```
                System.out.println(sSalida);
```

Línea 29:
Cierra el objeto `Scanner` `teclado` para liberar los recursos asociados con él. Es una buena práctica cerrar los flujos de entrada una vez que ya no se necesitan.

```
                teclado.close();

        }       // Fin de la funcion main
}
```

Resumen del Funcionamiento del Programa

1. **Entrada del Usuario:**
   El programa solicita al usuario que ingrese tres números separados por espacios.
2. **Procesamiento de Datos:**
   - Lee la línea de entrada y la divide en tres partes.
   - Convierte cada parte a un número entero.
   - Determina si cada número es par o impar.
   - Construye una cadena de salida donde cada número par se representa como "VERDE" y cada impar como "AZUL".
3. **Salida:**
   Muestra la cadena resultante que indica "VERDE" o "AZUL" para cada uno de los tres números ingresados, separados por espacios.

Ejemplo de Ejecución

Entrada del Usuario:

```
2 3 4
```

Salida del Programa:

```
VERDE AZUL VERDE
```

- **2** es par → "VERDE"
- **3** es impar → "AZUL"
- **4** es par → "VERDE"

Notas Adicionales

- **Validación de Entrada:**
  El programa asume que el usuario ingresará exactamente tres números separados por espacios. No incluye manejo de errores para casos donde la entrada no cumpla con este formato.
- **Espacios en la Salida:**
  La primera concatenación no agrega un espacio antes de "VERDE" o "AZUL", mientras que las siguientes sí, lo que garantiza que la salida esté correctamente separada.

Si tienes alguna pregunta adicional o necesitas más detalles sobre alguna parte del código, ¡no dudes en preguntar!