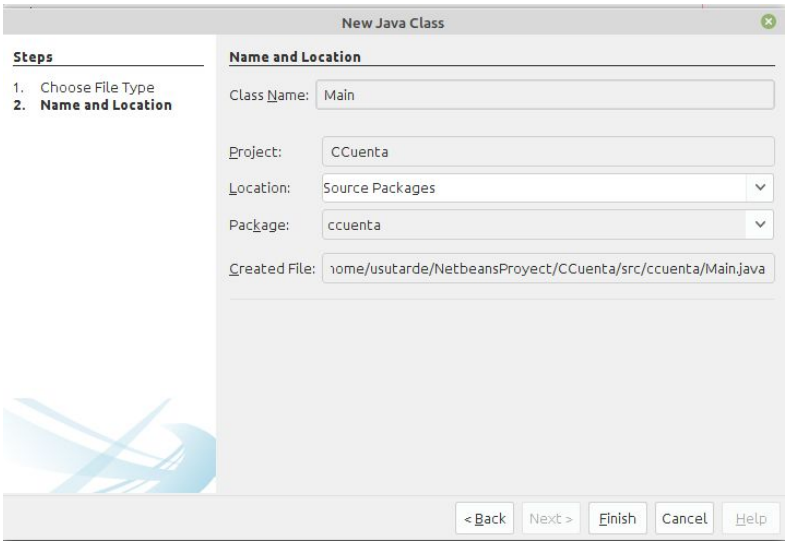
OPTIMIZACIÓN Y DOCUMENTACIÓN

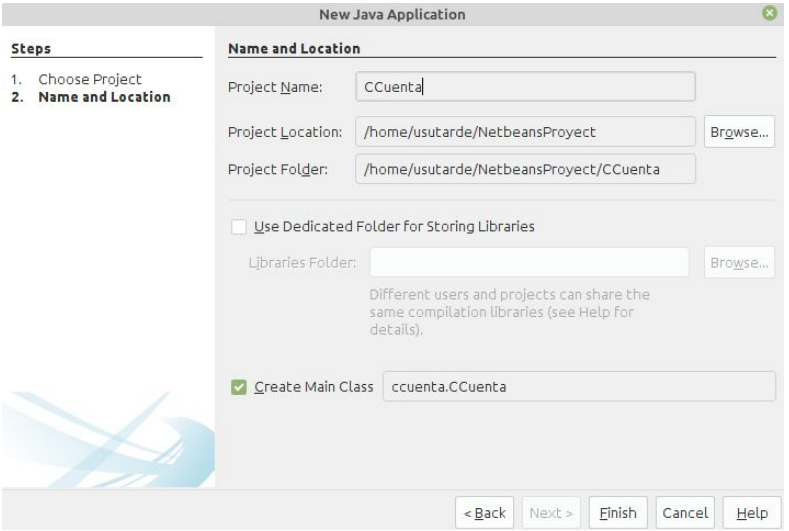
RA 4. Optimiza código empleando las herramientas disponibles en el entorno de desarrollo.

1. **En el proyecto Java "Depósito", hay definida una Clase llamada CCuenta, que tiene una serie de atributos y métodos. El proyecto cuenta asimismo con una Clase Main, donde se hace uso de la clase descrita.**

- Creamos el proyecto nuevo, con la clase llamada “Main”.



- Dicho proyecto, cuenta con otra clase denominada “CCuentas”.



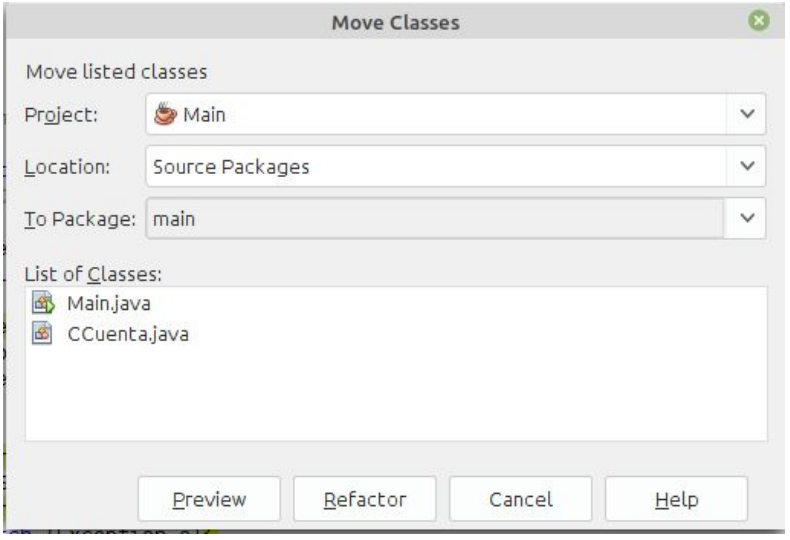
- Código Main.

```
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

        CCuenta cuenta1;
        double saldoActual;

        cuenta1 = new CCuenta ("Antonio Lopez","1000-2365-1230456789",2500,0);
        saldoActual = cuenta1.estado();
        System.out.println("El saldo actual es"+ saldoActual);

        try{
            cuenta1.retirar(2300);
        }catch (Exception e){
            System.out.println("Fallo al retirar");
        }
        try{
            System.out.println("Ingreso en cuenta");
            cuenta1.ingresar(695);
        }catch (Exception e){
            System.out.println("Fallo al ingresar");
```

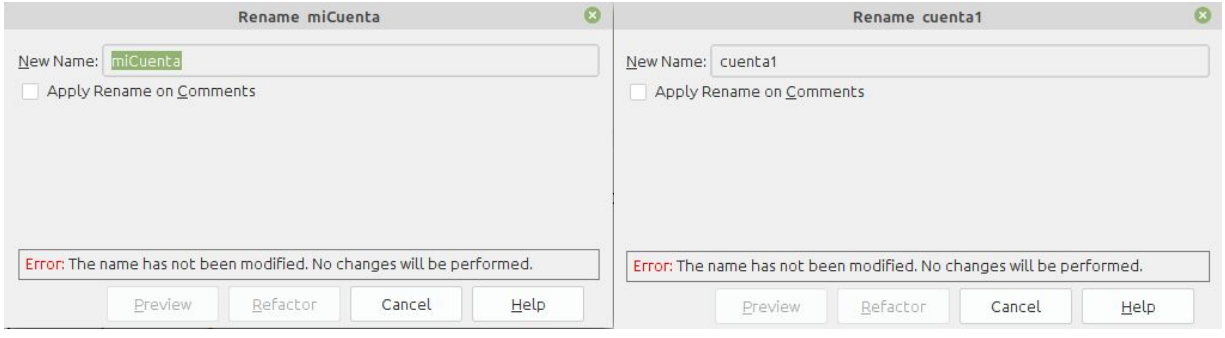
- Código CCuenta.

```
class CCuenta {

private String nombre;
private String cuenta;
private double saldo;
private double tipoInteres;

    public CCuenta() {
    }

    public CCuenta(String nombre, String cuenta, double saldo, double tipoInteres) {

        this.nombre = nombre;
        this.cuenta = cuenta;
        this.saldo = saldo;

    }

    public double estado (){

        return saldo;}

    public void ingresar (double cantidad) throws Exception{

        if (cantidad<0)
            throw new Exception ("No se puede ingresar una cantidad negativa");
        saldo = saldo - cantidad; }

    public void retirar (double cantidad) throws Exception{

        if (cantidad<= 0)
            throw new Exception ("No se puede ingresar una cantidad negativa");
        if (estado ()< cantidad)
            throw new Exception ("No se hay suficiente saldo");
        saldo = saldo - cantidad; }


}
```

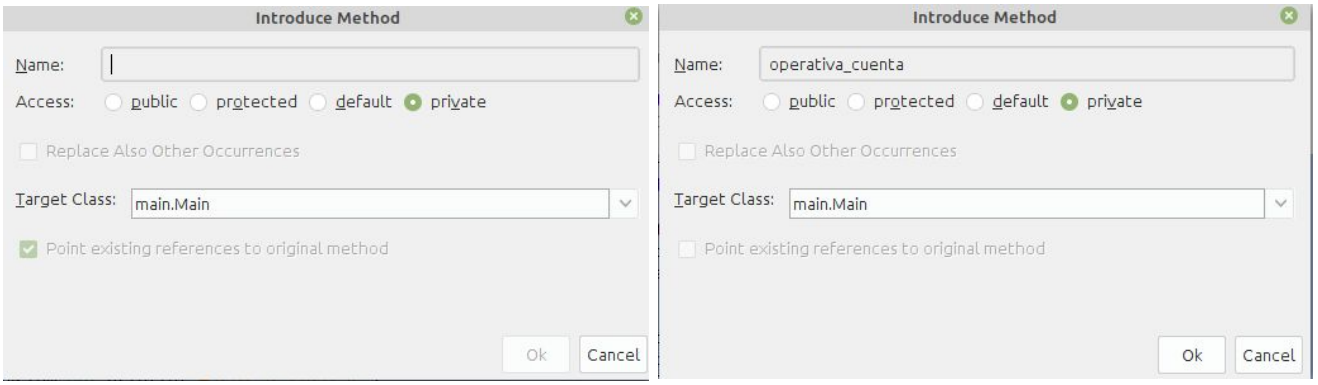
- Englobamos las clases.



- Cambiamos el nombre “miCuenta” por “cuenta1”.

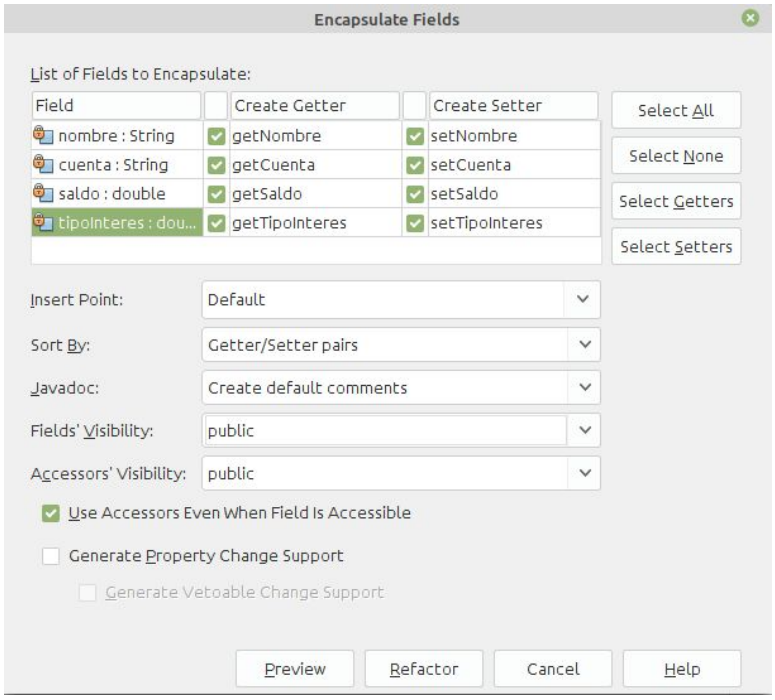


- Introducimos el método “operativa_cuenta”, donde engloba las sentencias de la clase “Main” que operan en conjunto con el objeto “Cuenta1”.



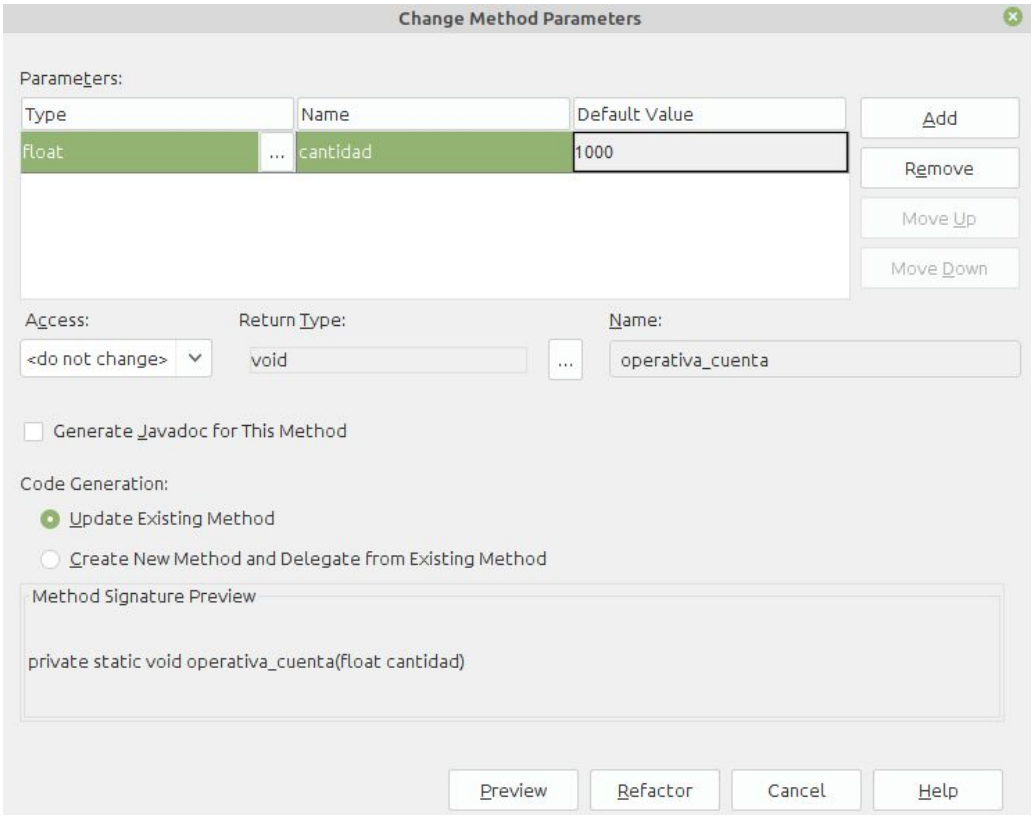

```java
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        operativa_cuenta();
    }

    private static void operativa_cuenta() {
        // TODO code application logic here

        CCuenta cuenta1;
        double saldoActual;

        cuenta1 = new CCuenta ("Antonio Lopez","1000-2365-1230456789",2500,0);
        saldoActual = cuenta1.estado();
        System.out.println("El saldo actual es"+ saldoActual);

        try{
            cuenta1.retirar(2300);
        }catch (Exception e){
            System.out.println("Fallo al retirar");
        }
        try{
            System.out.println("Ingreso en cuenta");
            cuenta1.ingresar(695);
        }catch (Exception e){
            System.out.println("Fallo al ingresar");
        }
    }

}
```

- Encapsulamos los atributos de la clase “CCuenta”.




```
class CCuenta {

    /**
     * @return the nombre
     */
    public String getNombre() {
        return nombre;
    }

    /**
     * @param nombre the nombre to set
     */
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    /**
     * @return the cuenta
     */
    public String getCuenta() {
        return cuenta;
    }

    /**
     * @param cuenta the cuenta to set
     */
    public void setCuenta(String cuenta) {
        this.cuenta = cuenta;
    }

    /**
     * @return the saldo
     */
    public double getSaldo() {
        return saldo;
    }

    /**
     * @param saldo the saldo to set
     */
    public void setSaldo(double saldo) {
        this.saldo = saldo;
    }

    /**
     * @return the tipoInteres
     */
    public double getTipoInteres() {
        return tipoInteres;
    }

    /**
     * @param tipoInteres the tipoInteres to set
     */
    public void setTipoInteres(double tipoInteres) {
        this.tipoInteres = tipoInteres;
    }
```

- Añadimos un nuevo párametros al método “operativa_cuenta”, de nombre “cantidad” y de tipo float.




```
public class main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        operativa_cuenta(1000);
    }

    private static void operativa_cuenta(float cantidad) {
        // TODO code application logic here

        CCuenta cuenta1;
        double saldoActual;

        cuenta1 = new CCuenta ("Antonio Lopez","1000-2365-1230456789",2500,0);
        saldoActual = cuenta1.estado();
        System.out.println("El saldo actual es"+ saldoActual);

        try{
            cuenta1.retirar(2300);
        }catch (Exception e){
            System.out.println("Fallo al retirar");
        }
        try{
            System.out.println("Ingreso en cuenta");
            cuenta1.ingresar(695);
        }catch (Exception e){
            System.out.println("Fallo al ingresar");
        }
    }
```