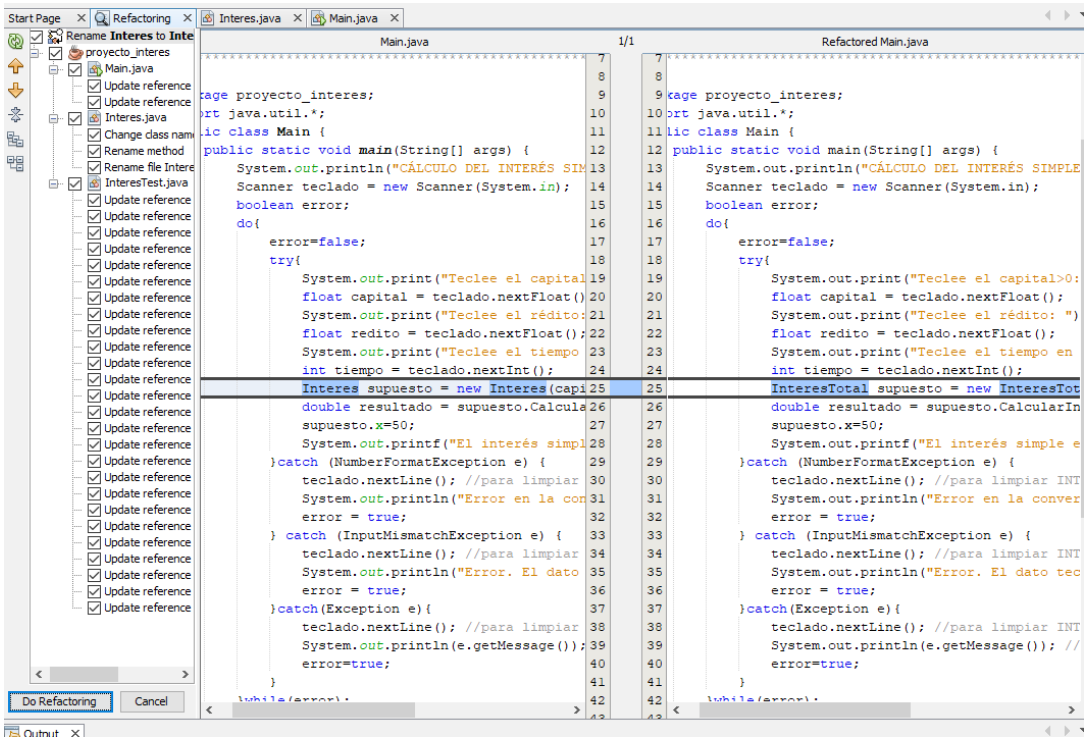
Examen Contornos de desarrollo. Estefanía Penide Casanova

1. Refactorizar código empregando o proxecto Java NetBeans proyecto_interes. Recomendación: (2,5 puntos)
• duplica o proxecto, traballa sobre a copia e mantén o orixinal sen variación para recorrer a el en caso necesario.
• Entregar capturas de cada apartado (ten que quedar claro que se está refactorizando)
• Entregar o ficheiro java cá refactorización realizada
a) Renomear a clase Interes por InteresTotal.



b) Borrar de forma segura o campo x da clase InteresTotal.

```java
 * autor: profesor
 */
package proyecto_interes;
public class InteresTotal {
    public int x=2;
    private float capital;
    private float redito;
    private int tiempo;
    public InteresTotal(float capital, float redito, int tiempo) {
        this.capital=capital;
        this.redito=redito;
        this.tiempo=tiempo;
    }
    public double CalcularInteres()throws Exception{
        if (capital<=0)
            throw (new Exception ("Error. El capital tiene que ser >=0"));
        if (tiempo<=0)
            throw (new Exception ("Error. El interés tiene que ser >=0"));
        x=25;
        return (double)capital*redito*tiempo/100;
    }
```

proyecto_interes.InteresTotal > x

Output | Usages

Usages of x [2 occurrences]
- proyecto_interes
  - Main.java
    - 27: supuesto.x=50;
  - InteresTotal.java
    - 23: x=25;

Start Page | Refactoring | InteresTotal.java | Main.java

Delete element(s) and ke
- proyecto_interes
  - InteresTotal.java
  - Update

Do Refactoring | Cancel

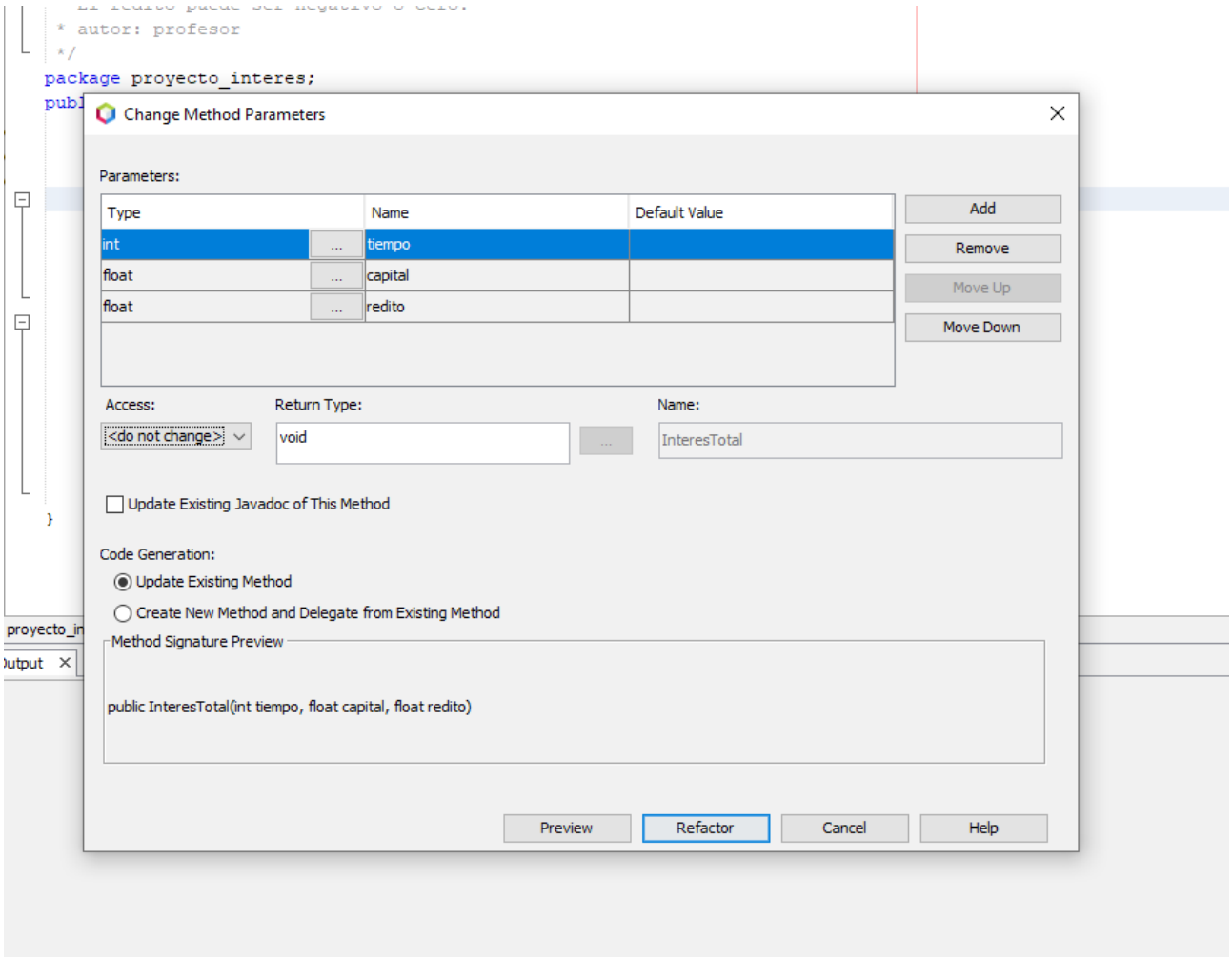
InteresTotal.java — 1/1 — Refactored InteresTotal.java

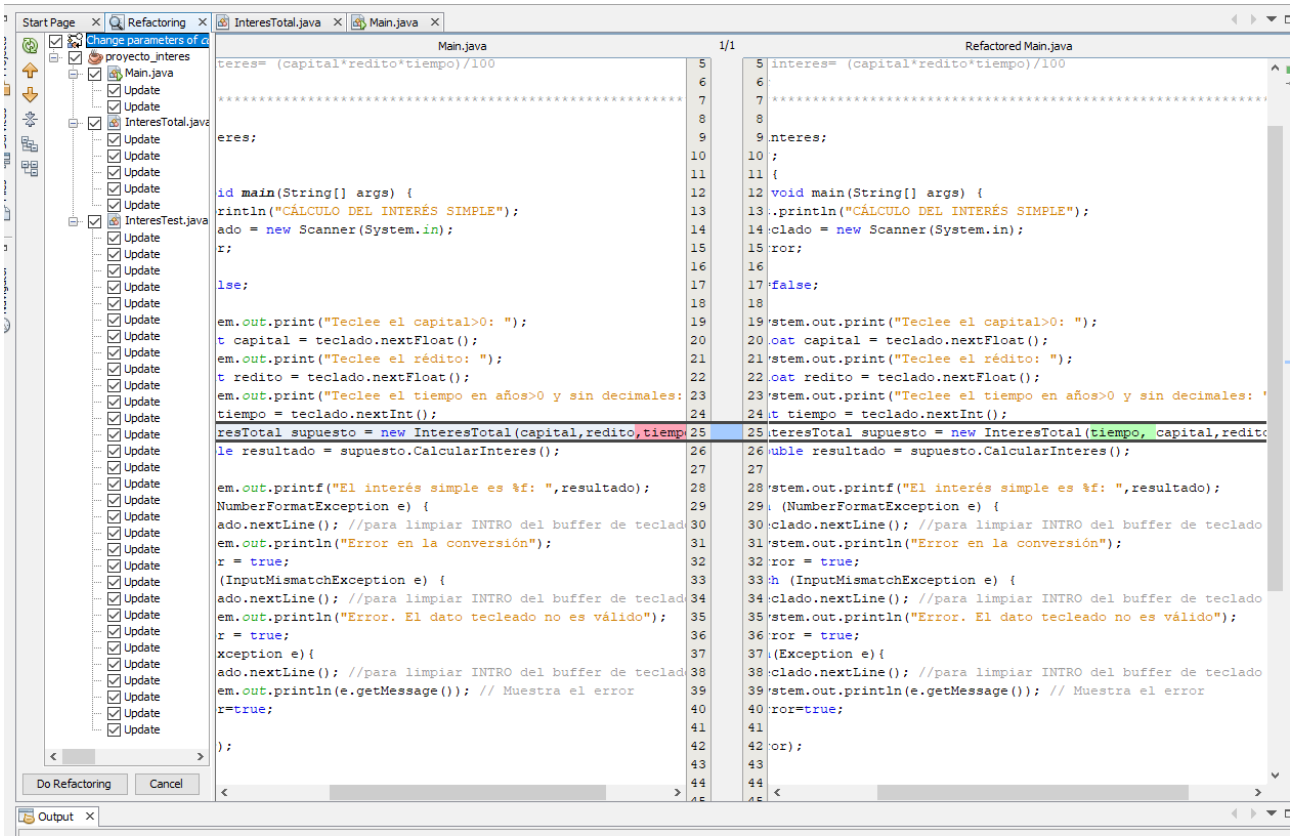
```java
/*
 * Cálculo de interés simple = capital * redito
 * El capital y el tiempo son >0.
 * El redito puede ser negativo o cero.
 * autor: profesor
 */
package proyecto_interes;
public class InteresTotal {
    public int x=2;
    private float capital;
    private float redito;
    private int tiempo;
    public InteresTotal(float capital, float re
        this.capital=capital;
        this.redito=redito;
        this.tiempo=tiempo;
    }
    public double CalcularInteres()throws Excep
        if (capital<=0)
            throw (new Exception ("Error. El ca
        if (tiempo<=0)
            throw (new Exception ("Error. El in

        return (double)capital*redito*tiempo/10
    }
}
```

```java
/*
 * Cálculo de interés simple = capital * redito * t
 * El capital y el tiempo son >0.
 * El redito puede ser negativo o cero.
 * autor: profesor
 */
package proyecto_interes;
public class InteresTotal {
    private float capital;
    private float redito;
    private int tiempo;
    public InteresTotal(float capital, float redito
        this.capital=capital;
        this.redito=redito;
        this.tiempo=tiempo;
    }
    public double CalcularInteres()throws Exceptior
        if (capital<=0)
            throw (new Exception ("Error. El capita
        if (tiempo<=0)
            throw (new Exception ("Error. El interé

        return (double)capital*redito*tiempo/100;
    }
}
```

```java
 */
package proyecto_interes;
public class InteresTotal {
    private float capital;
    private float redito;
    private int tiempo;
    public InteresTotal(float capital, float redito, int tiempo) {
        this.capital=capital;
        this.redito=redito;
        this.tiempo=tiempo;
    }
    public double CalcularInteres()throws Exception{
        if (capital<=0)
            throw (new Exception ("Error. El capital tiene que ser >=0"));
        if (tiempo<=0)
            throw (new Exception ("Error. El interés tiene que ser >=0"));

        return (double)capital*redito*tiempo/100;
    }
}
```

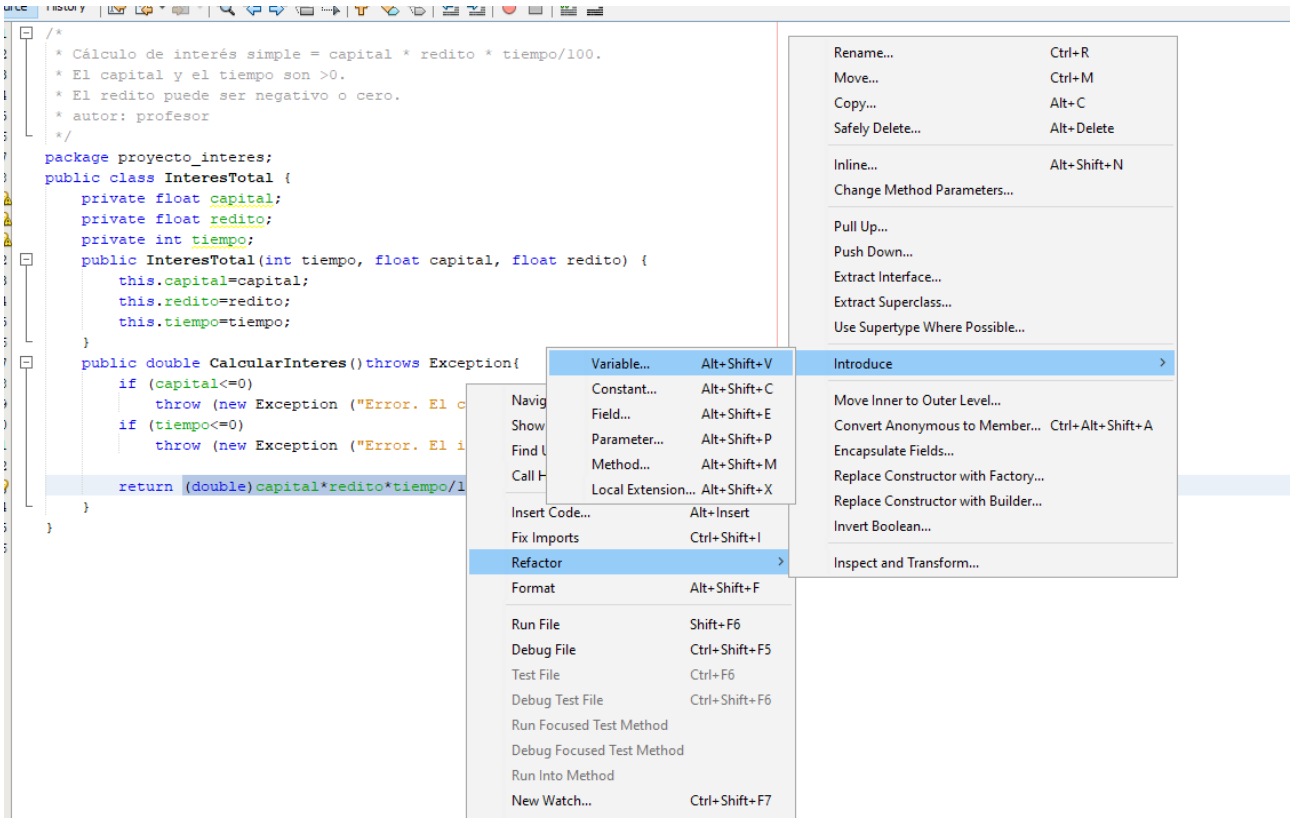
c) Cambiar o método construtor InteresTotal para que o tempo apareza como primeiro parámetro.

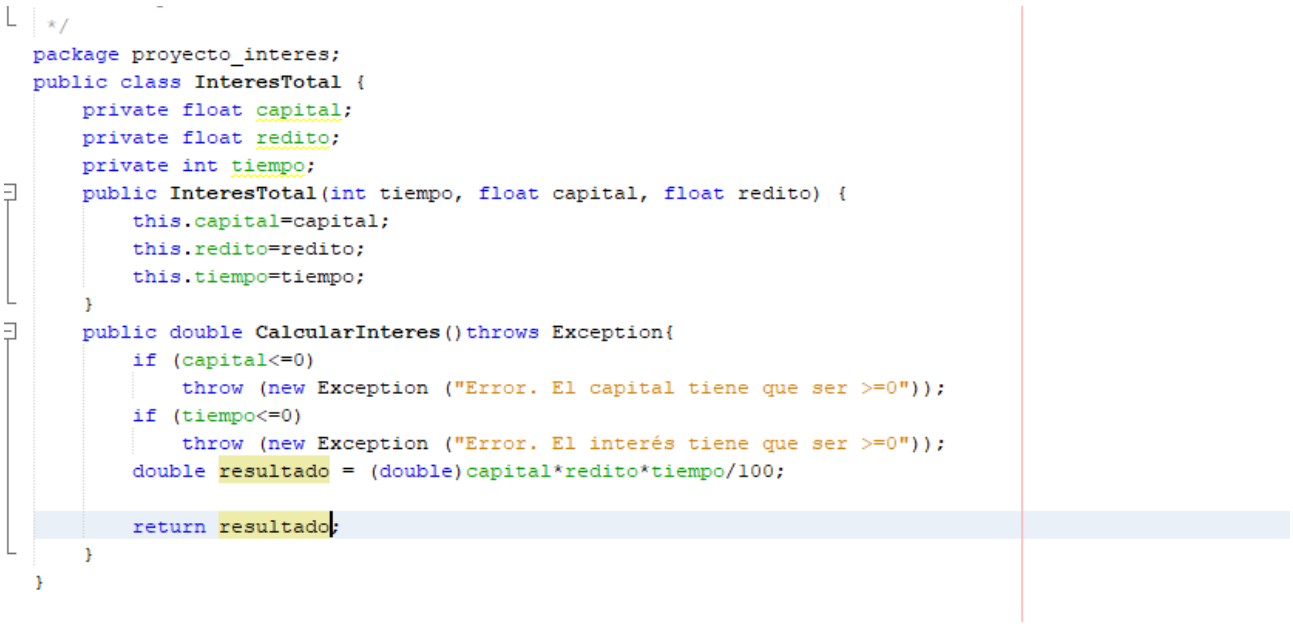

* autor: profesor
*/
package proyecto_interes;
Change Method Parameters
Parameters:
Type
Name
Default Value
int
tiempo
float
capital
float
redito
Add
Remove
Move Up
Move Down
Access:
<do not change>
Return Type:
void
Name:
InteresTotal
Update Existing Javadoc of This Method
Code Generation:
Update Existing Method
Create New Method and Delegate from Existing Method
Method Signature Preview
public InteresTotal(int tiempo, float capital, float redito)
Preview
Refactor
Cancel
Help



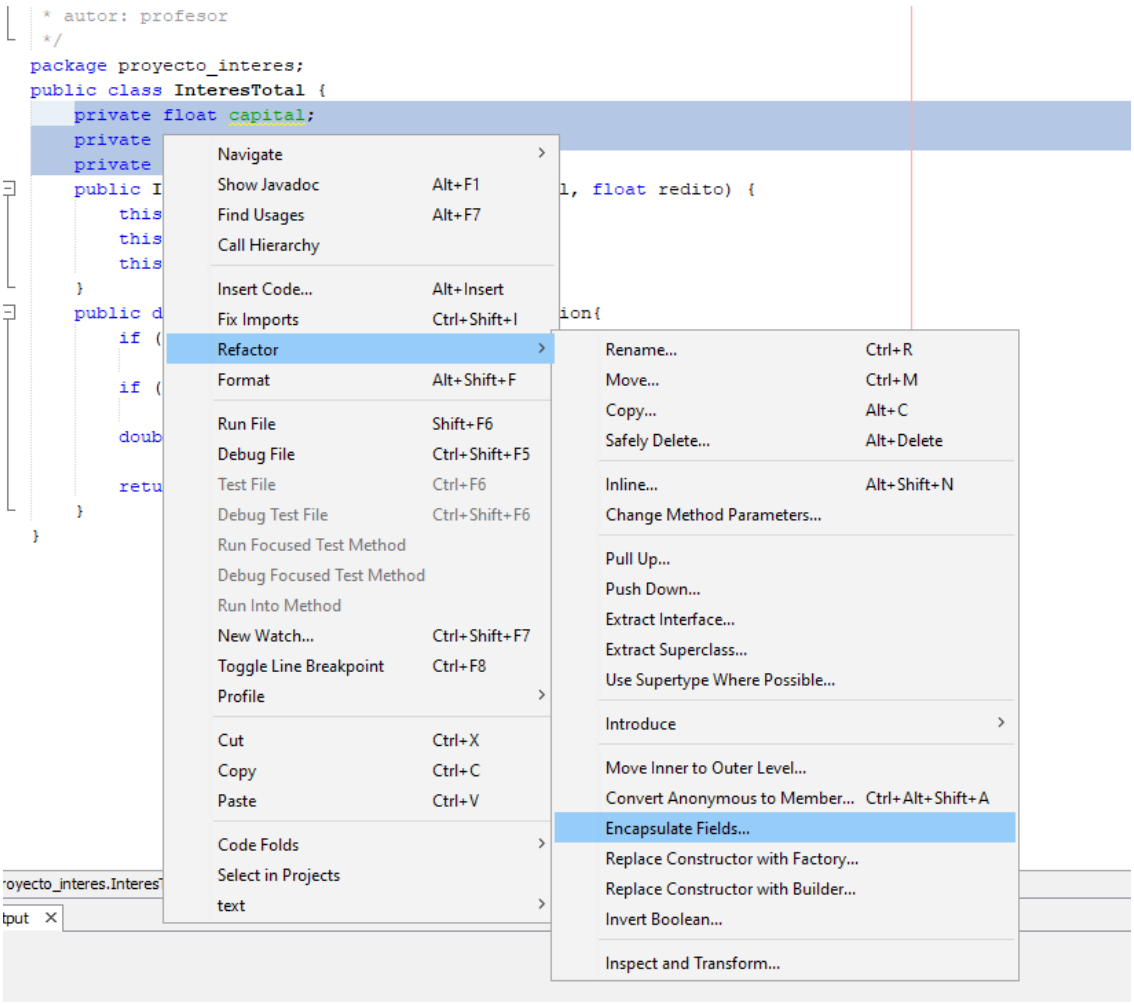

```java
 * autor: profesor
 */
package proyecto_interes;
public class InteresTotal {
    private float capital;
    private float redito;
    private int tiempo;
    public InteresTotal(int tiempo, float capital, float redito) {
        this.capital=capital;
        this.redito=redito;
        this.tiempo=tiempo;
    }
    public double CalcularInteres()throws Exception{
        if (capital<=0)
            throw (new Exception ("Error. El capital tiene que ser >=0"));
        if (tiempo<=0)
            throw (new Exception ("Error. El interés tiene que ser >=0"));

        return (double)capital*redito*tiempo/100;
    }
}
```

d) Definir a fórmula (double)capital*redito*tiempo/100 como o campo privado double resultado.

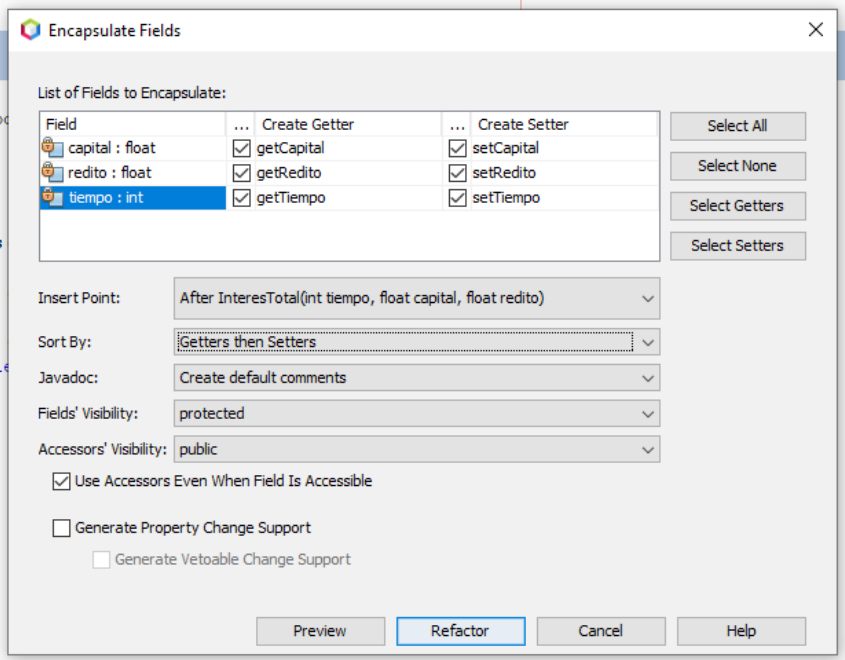


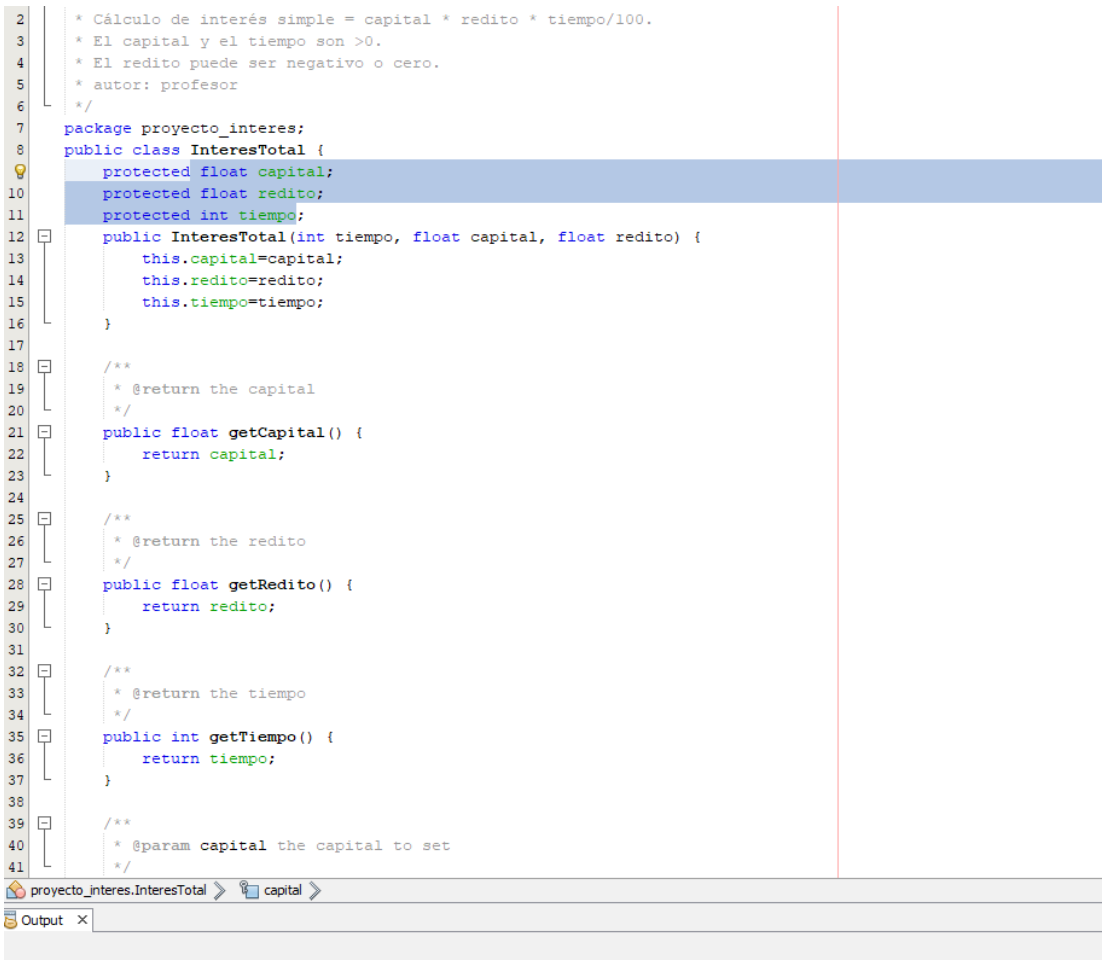


e) Crear os métodos públicos get e set para os campos capital, redito e tiempo da clase InteresTotal e poñer eses campos como protexidos.

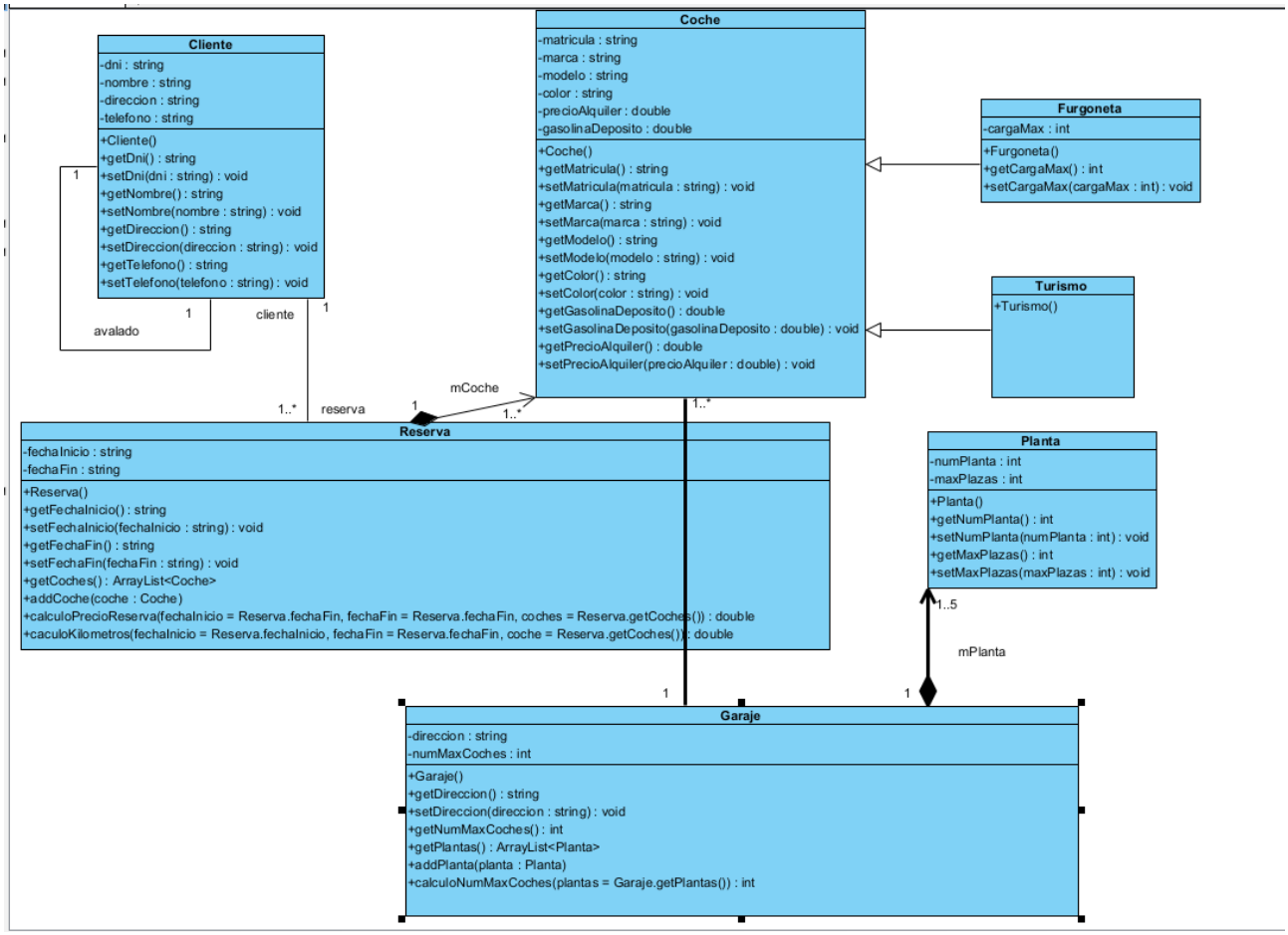

* autor: profesor
*/
package proyecto_interes;
public class InteresTotal {
private float capital;
private
private
public I
this
this
this
}
public d
if (
if (
doub
retu
}
}
l, float redito) {
ion{
Navigate
Show Javadoc Alt+F1
Find Usages Alt+F7
Call Hierarchy
Insert Code... Alt+Insert
Fix Imports Ctrl+Shift+I
Refactor
Format Alt+Shift+F
Run File Shift+F6
Debug File Ctrl+Shift+F5
Test File Ctrl+F6
Debug Test File Ctrl+Shift+F6
Run Focused Test Method
Debug Focused Test Method
Run Into Method
New Watch... Ctrl+Shift+F7
Toggle Line Breakpoint Ctrl+F8
Profile
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Code Folds
Select in Projects
text
Rename... Ctrl+R
Move... Ctrl+M
Copy... Alt+C
Safely Delete... Alt+Delete
Inline... Alt+Shift+N
Change Method Parameters...
Pull Up...
Push Down...
Extract Interface...
Extract Superclass...
Use Supertype Where Possible...
Introduce
Move Inner to Outer Level...
Convert Anonymous to Member... Ctrl+Alt+Shift+A
Encapsulate Fields...
Replace Constructor with Factory...
Replace Constructor with Builder...
Invert Boolean...
Inspect and Transform...
oyecto_interes.Interes
tput

```
 * El redito puede ser negativo o cero.
 * autor: profesor
 */
package proyecto_interes;
public class InteresTotal {
    private float capital;
    private float redito;
    private int tiempo;
    public InteresTotal(int tiempo
        this.capital=capital;
        this.redito=redito;
        this.tiempo=tiempo;
    }
    public double CalcularInteres
        if (capital<=0)
            throw (new Exception
        if (tiempo<=0)
            throw (new Exception
        double resultado = (double

        return resultado;
    }
}
```




* Cálculo de interés simple = capital * redito * tiempo/100.
* El capital y el tiempo son >0.
* El redito puede ser negativo o cero.
* autor: profesor
*/
package proyecto_interes;
public class InteresTotal {
protected float capital;
protected float redito;
protected int tiempo;
public InteresTotal(int tiempo, float capital, float redito) {
this.capital=capital;
this.redito=redito;
this.tiempo=tiempo;
}
/**
* @return the capital
*/
public float getCapital() {
return capital;
}
/**
* @return the redito
*/
public float getRedito() {
return redito;
}
/**
* @return the tiempo
*/
public int getTiempo() {
return tiempo;
}
/**
* @param capital the capital to set
*/
proyecto_interes.InteresTotal
capital
Output

3. Realiza un diagrama de clases para o seguinte escenario: (1,5 puntos)
Deséxase deseñar unha aplicación informática para unha empresa dedicada ao aluguer de automóbiles, tendo en conta que:
– Un determinado cliente pode ter nun momento dado varias reservas.
– De cada cliente interesa o seu DNI, nome, dirección e telefono.
– Cada cliente puiden ser avalado por outro cliente da empresa.
– Todo coche ten asignado un garaxe que non pode cambiar. Do garaxe interésanos a súa dirección e o número máximo de coches que pode albergar. A aplicación debe poder informar do número de coches que hai asignados a un garaxe nun momento dado. Los garaxes poden ter ata 5 plantas e interésanos almacenar o número máximo de prazas en cada unha delas. Identifícanse polo número de planta.
– De cada coche requírese a matrícula, marca, modelo e cor. Interésanos diferenciar as furgonetas dos turismos, porque das primeiras teremos que almacenar a carga máxima que soportan.
– Unha reserva realízaa un único cliente sobre un coche concreto.
– É importante rexistrar a data de inicio e fin da reserva, o prezo de aluguer de cada un dos coches, os litros de gasolina no deposito no momento de realizar a reserva, e débese de poder calcular o prezo total da reserva e unha estimación dos quilómetros que pode realizar o cliente sen necesidade de repoñer.



La relación entre cliente y coche también la podía haber planteado usando la clase Reserva como una clase de asociación entre ambas.

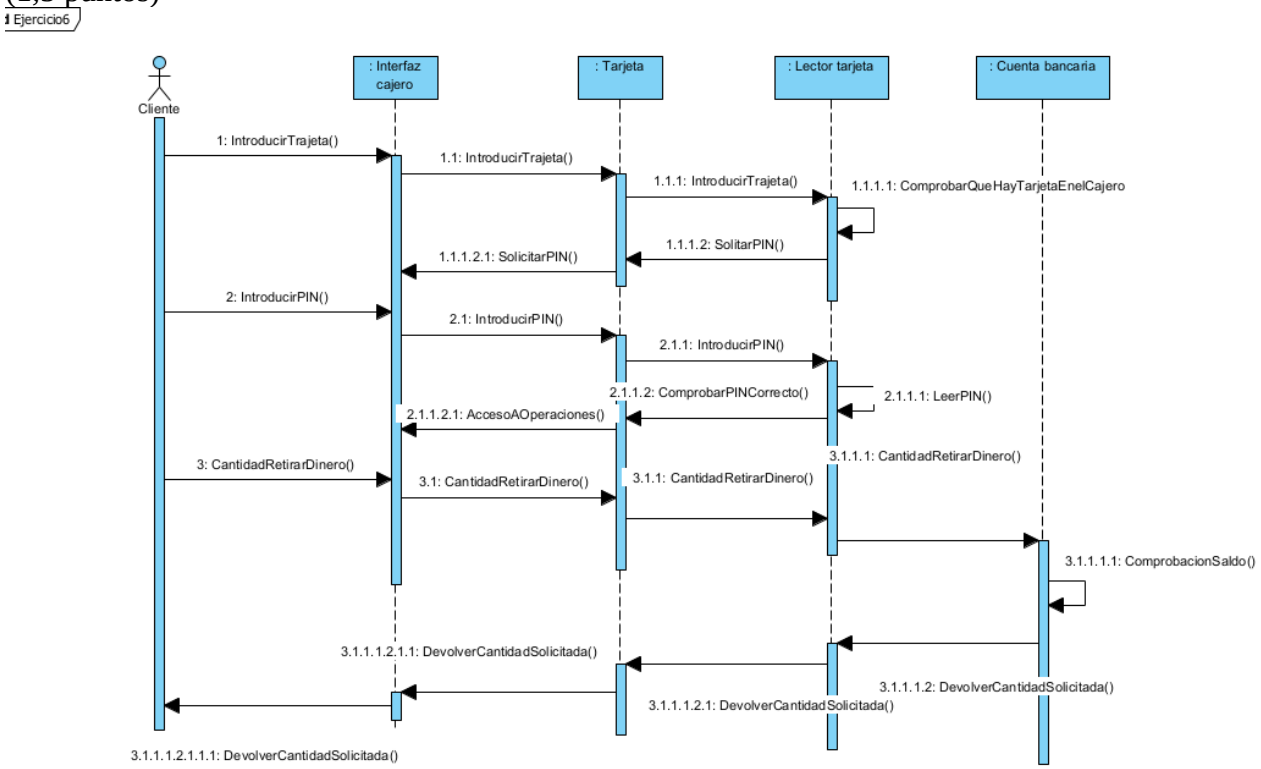
4. Dado o seguinte diagrama de clases, redacta un posible escenario que se adapte á información recollida no diagrama: (1,5 puntos)

Unos cliente hacen pedidos a una determinada empresa. Cada cliente se identifica por un número de cliente. Además de los mismos se registra su dirección, el saldo del que disponen para gastar, cual es el límite de su crédito y si disponen de algún descuento personal. Serán avisados cuando llegan al límite de su saldo y esto será cuando esté sea mayor o igual del 90% de su límite de crédito.
Cada cliente pude hacer múltiples pedidos. De cada pedido se registra su coste total (previamente claculado). Cada pedido consta de una cabecera donde se indica su código, la dirección de envío y la fecha, también tiene una línea de pedido donde se indica la cantidad de artículos que se envían. De un mismo artículo se pueden enviar múltiples unidades. Cada artículo es identificado con su número de artículo y consta de una descripción y de un precio.
Los artículos vienen de distintas fábricas, las cuales tienen un número identificativo y un número de teléfono propio. Un stock de artículos sirve para determinar de qué fabrica viene cada uno, así como tener un histórico de artículos y conocer las existencias actuales de cada uno de ellos. Se debe ir actualizando el histórico de artículos con el tiempo y poder avisar cuando no quedan existencias de un determinado artículo.

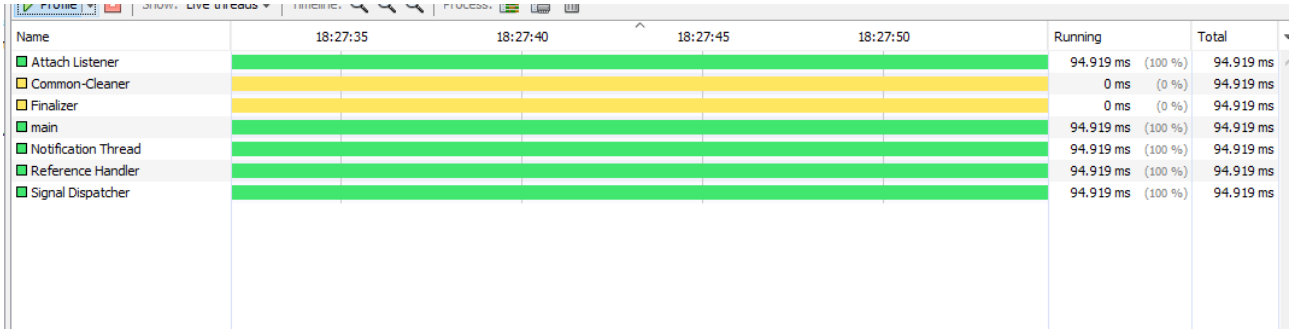
5. Dado o seguinte diagrama de casos de uso, redacta un posible escenario que se adapte á información recollida no diagrama: (1,5 puntos)

Un cajero de un banco permite realizar una serie de operaciones a los clientes con tarjeta. Introduciendo la tarjeta en el cajero, tras identificarse, los usuarios pordrán consultar su saldo y despositar dinero (en efectivo o mediante un cheque). En caso de que los clientes dispusieran del tipo de tarjeta inteligente podrían, además de las operaciones ya descritas, y también tras su identificación, retirar dinero.
El mantenimento de los cajeros se lleva a cabo por medio de los empleados, cuyas funcionen son las de reponer el dinero en el cajero y retirar las tarjetas que se hayan quedado retenidas.

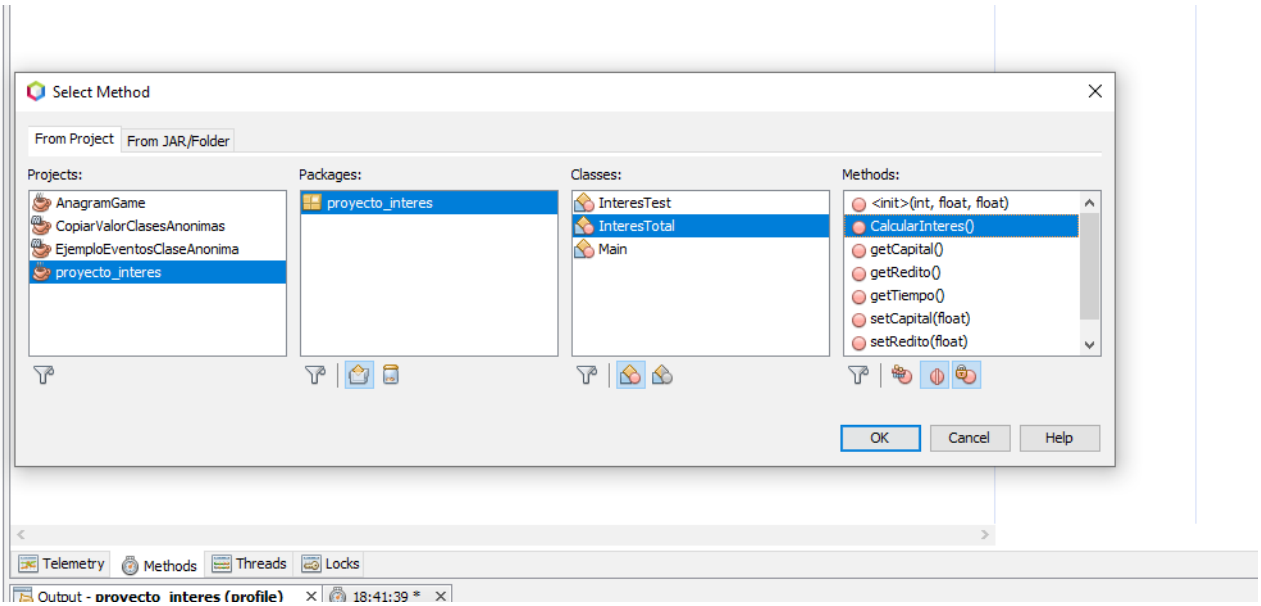
6. Realiza un diagrama de secuencia que mostre un caso de uso para retirar diñeiro dun caixeiro. Constará dos seguintes elementos: cliente, interface caixeiro, tarxeta, lector tarxeta e conta bancaria. (1,5 puntos)



2. Profiling (1,5 puntos)
a) Abre o proxecto Java proyecto_interes. Executa o Profiling en modo monitor, con monitoreo de fios, fai unha copia da pantalla.



b) Executa o Profiling analizando parte da aplicación, fixando o método raíz (Interes.java): public float CalcularInteres() ¿Cantos milisegundos levou a execución do método CalcularInteres? Fai unha copia da pantalla onde se mostren estos resultados.



c) Abre o proxecto Java estadísticos. Analizar o rendemento dos métodos factorial, combinaciones e variaciones da clase Estadisticos. Visualiza os resultados do análise en vivo despois de teclear m e n e ver o valor das súas combinacións e as variacións.