

UF4 ENTORNOS DE DESARROLLO
Actividad 2. Tarea en equipo
CONTROL DE VERSIONES

MARIA CARMEN CORREA HERAS
DANIEL DEL RÍO PÉREZ
JUAN RAMON VARÓ NÚÑEZ
VERÓNICA BONIS MARTÍN

Los integrantes del equipo son:

MARIA CARMEN CORREA HERAS
DANIEL DEL RÍO PÉREZ

JUAN RAMON VARÓ NÚÑEZ
VERÓNICA BONIS MARTÍN

URL repositorio remoto: <https://github.com/carwenblue/EntornosDesarrolloTelefonica>



A continuación se presentan las tareas a realizar y el método de trabajo elegido:

Requerimiento 1

Para resolver el primer requerimiento mantenemos una corta reunión (a través de Discord) en la que definimos la estructura que utilizaremos para implementar el código de la calculadora. Se decide finalmente crear una clase *Main* que nos permitirá probar el funcionamiento de las clases suma, resta, producto y multiplicación mediante un menú (*switch*) en el que podemos elegir la clase (Suma “s”, Resta “r”, Producto “p” y Cociente “d”) y dentro de cada clase otro menú (*switch*) que nos permite probar cada método (1, 2, 3, 4).

Cada integrante instala GIT y crea su usuario en GitHub, compartimos estos usuarios para darnos permiso para trabajar en el repositorio remoto:

- [María del Carmen Correa Heras](#)
- [Juan Ramón Varó](#)
- [Daniel del Río Pérez](#)
- [Verónica Bonis Martín](#)

Hacemos algunas pruebas con el repositorio remoto y finalmente el flujo de trabajo seguido es el siguiente:

1) Clase principal (Main) y primera clase Producto:



Para empezar a trabajar todos sobre la misma base hacemos el primer commit de la clase principal (*Main*) sobre la que debemos trabajar todos los miembros del equipo.

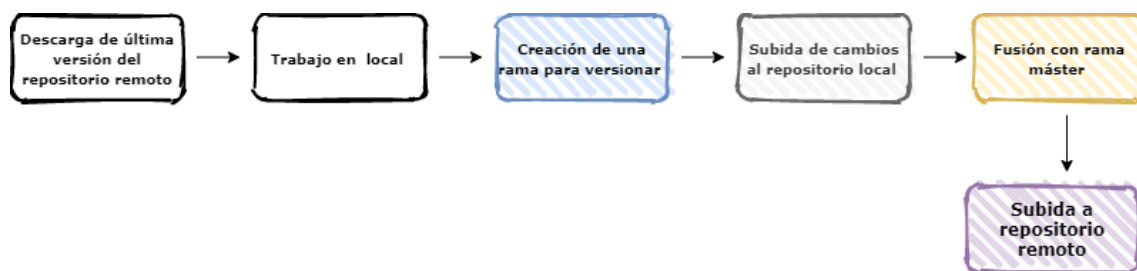
Commits on Feb 11, 2021

modificacion Main dan committed 6 days ago	2dbae18	<>
modificacion Main dan committed 6 days ago	5785669	<>
Merge branch 'master' of https://github.com/carwenblue/EntornosDesarr... dan committed 6 days ago	836dca8	<>
Prueba pull Danubio80 committed 6 days ago	Verified 0832f5b	<>
añado un Main: dan committed 6 days ago	6748723	<>

- Primera versión Main y Producto: Daniel del Río Pérez

Requerimiento 2







Continuamos trabajando sobre la versión ya subida del Main, para ello es necesario otro flujo de trabajo:



1. `git remote add origin https://github.com/carwenblue/EntornosDesarrolloTelefonica`
2. `git pull origin master` → trabajo y cambios en local
3. `git branch nombredelarama`
4. `git checkout nombredelarama`
5. `git add -A`
6. `git commit -m "Etiqueta del commit"`
7. `git checkout master`
8. `git branch merge nombredelarama`
9. `git push origin master`

En este punto tenemos algún problema de sincronización, ya que necesitamos trabajar todos sobre la clase principal Main, para ello establecemos un orden de trabajo y el siguiente miembro del equipo actualiza la clase principal y sube su clase utilizando para ello su rama y fusionando con la rama máster:

Commits on Feb 12, 2021







Merge branch 'carmen'		carwenblue committed 4 days ago		d9a43b1	
Suma y Principal con try-catch		carwenblue committed 4 days ago		9e1bcae	

- Segunda versión Main y Suma: María del Carmen Correa Heras

Commits on Feb 17, 2021

Main y clase Cociente		verobm committed 9 minutes ago		7f9ba83	
-----------------------	---	--------------------------------	---	---------	---

- Tercera versión Main y Cociente: Verónica Bonis Martín

Update Main.java		JRVaro committed yesterday	Verified		ecfe123	
Update resta.java		JRVaro committed yesterday	Verified		b37b3b5	

- Cuarta versión Main y Resta : Juan Ramón Varó

Requerimiento 3

Para cada método se han realizado algunas modificaciones sobre el código original documentado en el Javadoc:

Clase Suma

- Método 1, 2, 3 y 4 – Suma

Se implementa un caso especial (si n1 y/o n2 y/o n3 es menor de 0) en cada uno de los métodos, ya que queremos implementar una calculadora que no acepte los números negativos. Por otra parte, se eliminan los casos especiales de los números NaN por dificultades en la implementación.

```
/**
 * Este método suma dos numeros reales.<br>Casos especiales:
 * <br>Si n1 o n2 es un número negativo devuelve que no se permiten negativos.
 * @param n1 es el primer número real que se va a sumar
 * @param n2 es el segundo número real que se va a sumar
 * @return
 */
public static double sumar(double n1,double n2){
    if (n1 < 0 || n2 < 0)
        System.out.println("No se permiten números negativos");
    else {
        return sumaR= n1 + n2;
    }
    return sumaR;
}
```

```
/**
 * Este método suma dos numeros enteros.<br>Casos especiales:
 * <br>Si n1 o n2 es un número negativo devuelve que no se permiten negativos.
 * @param n1 represente el primero número entero a sumar
 * @param n2 representa el segundo número entero a sumar
 * @return la suma de ambos numeros, puede ser <u>negativo</u> si n1 mayor n2
 */
public static double Suma(int n1,int n2){
    if (n1 < 0 || n2 < 0)
        System.out.println("No se permiten números negativos");
    else {
        return sumaE= n1 + n2;
    }
    return sumaE;
}
```

```
/**
 * Este método suma tres numeros Reales.<br>Casos especiales:
 * <br>Si n1 o n2 es un número negativo devuelve que no se permiten negativos.
 * @param n1 represente el primero número a sumar
 * @param n2 representa el segundo número a sumar
 * @param n3 representa el tercer número a sumar
 * @return la suma de los tres números reales
 */
public static double sumar(double n1,double n2,double n3){
    if (n1 < 0 || n2 < 0)
        System.out.println("No se permiten números negativos");
    else
        return sumaR = n1 + n2 + n3;
    return sumaR;
}
```

```
/**
 * Este método acumula un resultado.<br>Casos especiales:
 * <br>Si n1 es un número negativo devuelve que no se permiten negativos.
 * @param n1 represente un acumulado, pero no retorna nada.
 * @return
 */
public static double sumarAcumulado(double numAcum){

    if (numAcum < 0)
        System.out.println("No se permiten números negativos");
    else
        return numAcum;
    return numAcum;
}
```

Clase Resta

- Método 1, 2, 3 y 4 – Resta

Se implementan los mismos casos que en la suma.

Clase Producto

- Método 1, 2, 3 y 4 – Producto

Se implementa un caso especial (si n1 y/o n2 y/o n3 es menor de 0) en cada uno de los métodos, ya que queremos implementar una calculadora que no acepte los números negativos. Por otra parte, se eliminan los casos especiales de los números NaN por dificultades en la implementación.

Se observa que cuando los números con los que se opera son excesivamente grandes la propia implementación del algoritmo por parte Java, dé como resultado *Infinity*.

```
/**
 * Este metodo multiplica dos números reales.<br>Casos especiales:
 *
 * <br>Si n1 o n2 es un número negativo devuelve que no se permiten negativos.
 *
 * @param n1 representa el primer número real a multiplicar.
 * @param n2 representa el segundo número real a multiplicar.
 *
 * @return el resultado de la multiplicación de ambos números reales.
 */

public static double producto(double n1, double n2) {

    // casos especiales

    if (n1 < 0 || n2 < 0)
        System.out.println("No se permiten números negativos");

    return n1 * n2;
```

```
/**
 * Este método multiplica dos números enteros.<br>Casos especiales:
 *
 * <br>Si n1 o n2 es un número negativo devuelve que no se permiten negativos.
 *
 * @param n1 representa el primer número entero a multiplicar.
 *
 * @param n2 representa el segundo número entero a multiplicar.
 *
 * @return la multiplicación de ambos números enteros.
 */

public static int producto(int n1, int n2) {

    // casos especiales

    if (n1 < 0 || n2 < 0)
        System.out.println("No se permiten números negativos");

    return n1 * n2;
```

```
/**
 * Este método multiplica tres números reales.<br>Casos especiales:
 *
 * <br>Si n1 o n2 es un número negativo devuelve que no se permiten negativos
 *
 * @param n1 representa el primer número real a multiplicar.
 * @param n2 representa el segundo número real a multiplicar.
 * @param n3 representa el tercer número real a multiplicar.
 *
 * @return el resultado de la multiplicación de los tres números reales.
 */

public static double producto(double n1, double n2, double n3) {

    // casos especiales

    if (n1 < 0 || n2 < 0 || n3 < 0)
        System.out.println("No se permiten números negativos");

    return n1 * n2 * n3;
```

```
/**
 *
 * Este método devuelve la potencia del n1 elevado al n2.<br>Casos especiales:
 *
 * <br>Si el segundo argumento es cero o negativo, entonces el resultado es 1.0.
 * <br>Si el segundo argumento es 1.0, entonces el resultado es el mismo que el del
 * primer argumento.
 *
 *
 * @param n1 representa la base entera.
 * @param n2 representa la potencia.
 *
 *
 * @return el resultado de elevar n1 a n2.
 *
 */
public static int potencia(int n1, int n2) {

    // casos especiales
    if (n2 <= 0 || n2 == 0 )
        return (int)1.0;
    if (n2 == 1 )
        return n1;

    return (int) Math.pow(n1, n2); // se hace un casting para convertir a entero el resultado.
```


Clase Cociente

- *Método 1 – Cociente*

Se implementan dos casos especiales (si n1 es 0 y si n2 es 0) y se elimina el caso en el que cualquiera de los número es NaN.

```
/**
 * Este método divide dos números reales.<br>Casos especiales:
 * <br>Si n1 es 0, el resultado será igual a 0.
 * <br>Si n2 es 0, el resultado no se puede mostrar porque tiende a infinito.
 * @param n1 representa el dividendo
 * @param n2 representa el divisor
 * @return la división de ambos números, puede ser decimal
 */
public static double cociente(double n1, double n2) {
    //casos especiales
    if (n1==0)
        return 0;
    if (n2==0)
        System.out.println("El resultado no se puede mostrar porque tiende a infinito");

    return (n1 / n2); // divide el primer número real entre el segundo real
}
```

- *Método 2 – Cociente*

Se implementan dos casos especiales (si n1 es 0 y si el resultado no da un número exacto) y se elimina el caso en el que cualquiera de los números es NaN.

```
/**
 * Este método divide dos números enteros.<br>Casos especiales:
 * <br>Si n1 es 0, el resultado sería igual a 0.
 * <br>Si el resultado no da un número exacto, solo muestra la parte entera al ser división entera.
 * @param n1 representa el dividendo
 * @param n2 representa el divisor
 * @return la división de ambos números
 */
public static int cociente(int n1, int n2) {
    //casos especiales
    if (n1==0)
        return 0;
    if (n1%n2!=0)
        System.out.println("Cociente entero sin decimales");
    return n1 / n2; // divide el primer número entero entre el entero
}
```

- *Método 3 – Inverso*

Se implementan un caso especial (si n1 es 0) y se elimina el caso en el que el número es NaN.

```
/**
 * Este método calcula el inverso de un número real.<br>Casos especiales:
 * <br>Si n1 es 0, el resultado no se puede mostrar porque tiende a infinito
 *
 * @param n1 representa el número real
 * @return el inverso del número real<br>
 *
 *
 */
public static double inverso(double n1) {

    //casos especiales
    if (n1==0)
        System.out.println("El resultado no se puede mostrar porque tiende a infinito");

    return 1 / n1; // calcula el inverso del número real n1
}
```

- *Método 4 - Raíz*

Se implementan un caso especial (si el argumento es 0) y se elimina el caso en el que el argumento es NaN.

```
/**
 * Este método calcula la raíz cuadrada de un número.<br>Casos especiales:
 * <br>Si el argumento es 0 (positivo o negativo), el resultado es el mismo que el argumento
 * @param n1 representa el número
 * @return la raíz cuadrada del número
 *
 */
public static double raiz(double n1) {

    //casos especiales
    if (n1==0)
        return n1;

    return Math.sqrt(n1); // calcula la raíz cuadrada positiva del número real n1
}
```

Una vez tenemos la versión final (v 2.2) del Main y de cada una de las clases generamos una nueva versión del JavaDoc que también se incluye en el repositorio remoto.