Ejercicio 1.

- **Abstracción,** se refiere a que se pueden representar los datos y características de un objeto pero ocultando las acciones y procesos que se realizan al usuario.
- **Modularidad,** consiste en dividir el código en diferentes partes que pueden usarse de manera individual o conjunta.
- **Encapsulamiento,** se refiere a que los atributos e información de un objeto está guardada dentro de el mismo y solo se permite ver desde fuera los atributos e información que se quiera
- **Herencia,** Significa que una clase u objeto recibe los atributos y rasgos de su clase padre, así se pueden extender los atributos y comportamiento desde las clases padres a las clases secundarias.
- **Polimorfismo,** consiste en tener varios objetos diferentes, que se usan de la misma manera pero que dan resultados distintos

Ejercicio 2.

- El resultado si N es 5 es 24.
- En la línea 15.
- En la línea 19.

Ejercicio 3.

1. está compuesto de dos clases, una es la clase Persona y otra es la clase Ejercicio1

2.1 Los atributo que vemos con de tipo prívate y son String

2.2 hay un constructor por parámetros desde la línea 12 a la 18 y tiene los parámetros nombre, apellido1, apellido2 y dni, no hay un constructor por defecto

2.3 hay 2 metodos.

2.4 este método tiene 2 argumentos, uno de tipo string que es clave y otro de tipo char que es letra, este método lo que hace es coger la primera letra del nombre y la añade al String clave, si el nombre estaría vacio pone una “a”.

Después si el primer apellido tiene más de tres letras , coge las tres primeras letras y las añade al string clave, si el primer apellido tiene menos de 3 letras añade las que tenga.

Después si el segundo apellido tiene por lo menos una letra, coge la ultima letra del apellido y la añade al string clave , si el segundo apellido estaría vacio añade una “a”.

Es decir que lo que hace es coger la primera letra del nombre, las 3 primeras letras el primer apellido y la ultima letra del segundo apellido y las junta en una string.

Con mi nombre y mis apellidos (David prado mejuto) la clave seria “**dprao**”.

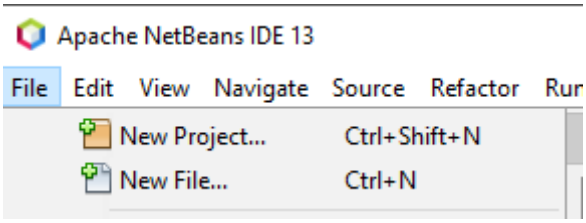
2.5 este método tiene 6 argumentos, que son resul**,** que es booleano , clave y dniNumeros que son String**,** letraDni y letraValida que son char, num que es long y resto que es int.

Este método lo que hace es extraer los números del DNI sin la letra y los guarda en un String, después extrae la letra del DNI y la guarada en un char , después convierte los números sacados del DNI que están en formato String a formato int para poder hacer una división de esos números entre 23, y recoge el resto de esa division.

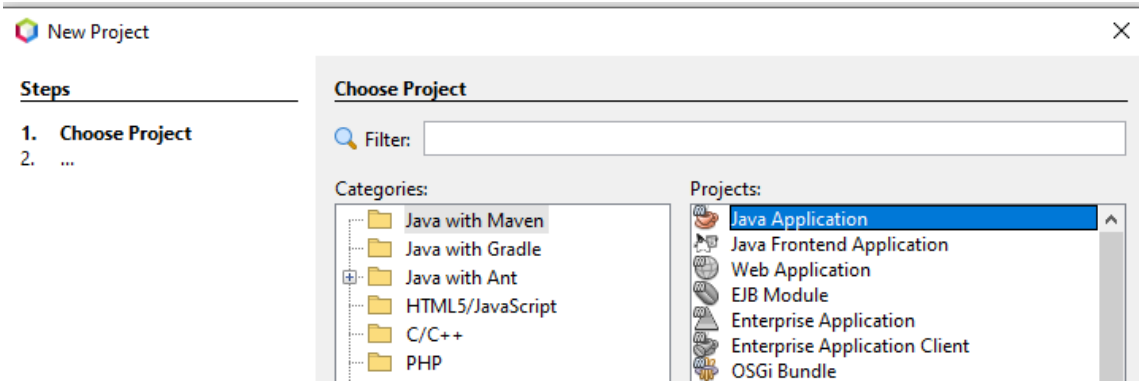
Una vez tiene el resto de esa división y tiene también la letra del DNI dado por el usuario , crea un String con un conjunto de letras, después mete en un char la letra que está en la numero de posición que ha dado el resto y compara le letra que está en la posición del resto con la letra que tiene el DNI dado por el usuario, si las dos son iguales revuelve un true, si no lo son devuelve un false.

Ejercicio 4.

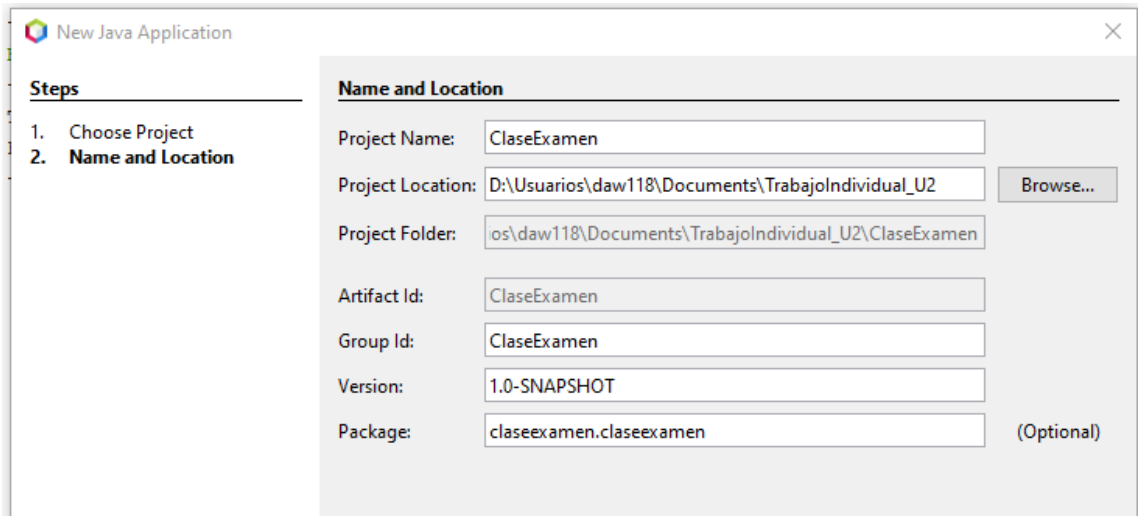
Para crear el proyecto le doy a "**NEW"** y a **"NEW PROJECT"**.



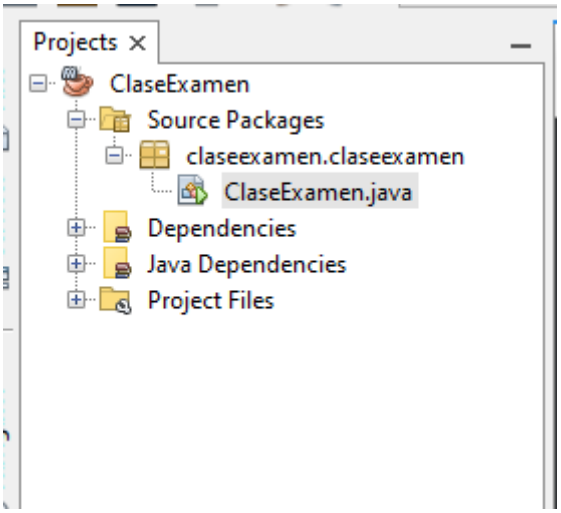
Despues a **"JAVA with Maven"** y a **"Java Application"**



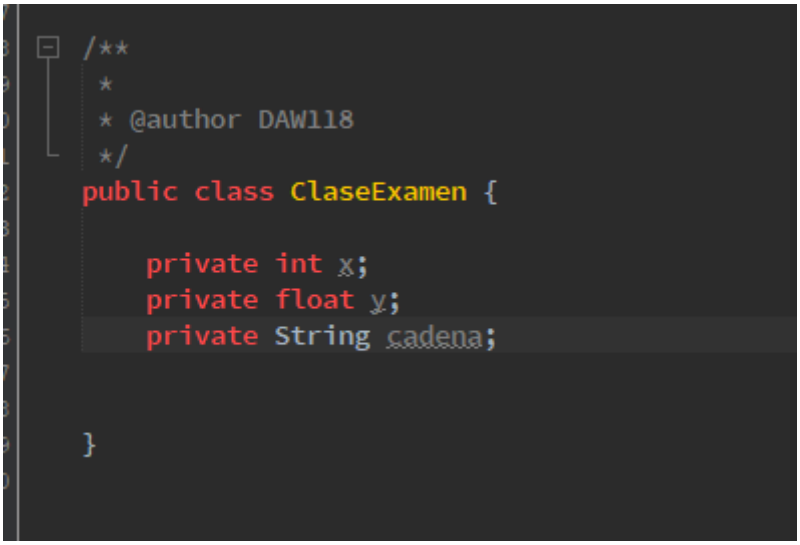
Despues le doy un nombre al proyecto y le he creado una carpeta donde guardarse.



Una vez dado el nombre se crea esta estructura.

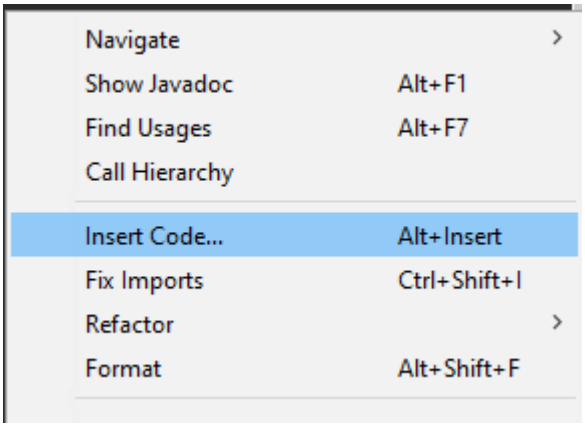


Ahora voy a modificar el código, primero creo los atributos de la clase que vamos a usar

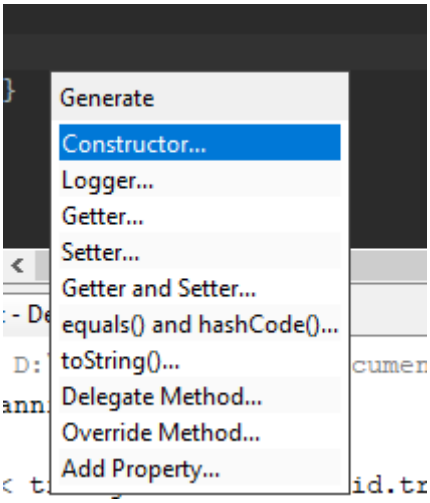


```
/**
 *
 * @author DAW118
 */
public class ClaseExamen {

    private int x;
    private float y;
    private String cadena;


}
```

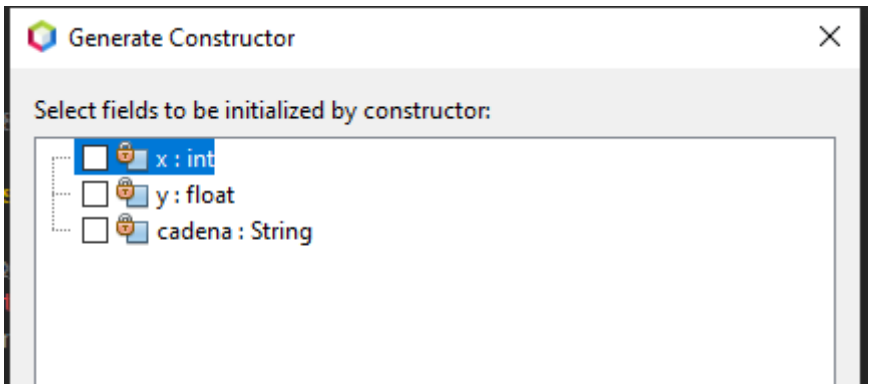
Después usando el creador de código automático voy a insertar un constructor sin parámetros, así que doy click derecho en el editor y voy a "**insert code"**,



Elijo el Constructor



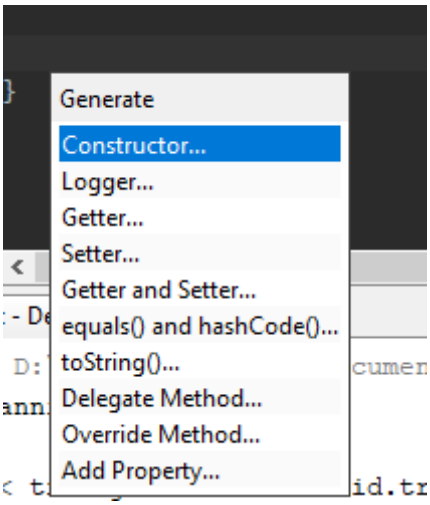
Le digo que no tiene ningún parámetro



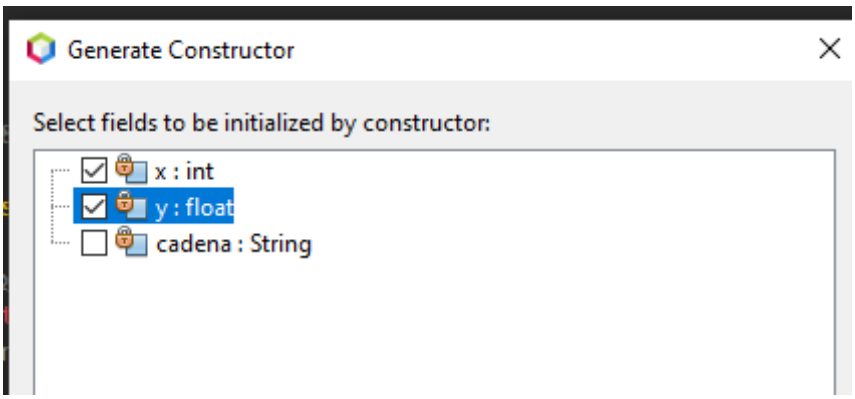
Y se crea el constructor sin parámetros

```
5
6   package claseexamen.claseexamen;
7
8   /**
9    *
10   * @author DAW118
11   */
12  public class ClaseExamen {
13
14      private int x;
15      private float y;
16      private String cadena;
17
18      public ClaseExamen() {
19      }
20
```

Ahora voy a crear el constructor con parámetros, voy a insert code y eligo "**constructor**"



Y le digo que me añada los parámetros X e Y.



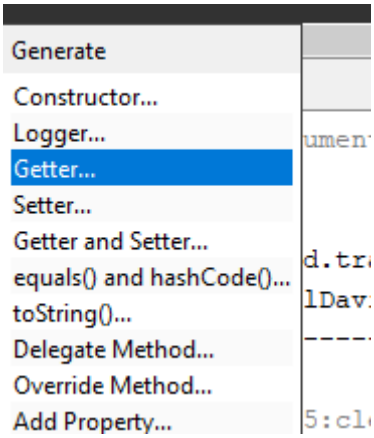
Cuando lo genera ya nos aparecen los parámetros y las asignaciones.

```
public ClaseExamen(int x, float y) {
    this.x = x;
    this.y = y;
}
```

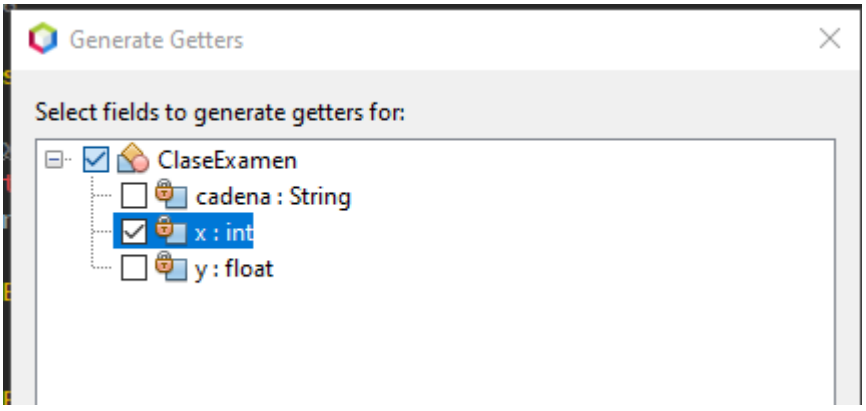
Aparte añado a mano la inicialización de cadena y la pongo vacia

```
21      public ClaseExamen(int x, float y) {
22          this.x = x;
23          this.y = y;
24          cadena = "";
25      }
```

Ahora voy a crear un método Get, para ello voy a insertar un código tipo **"Getter"**

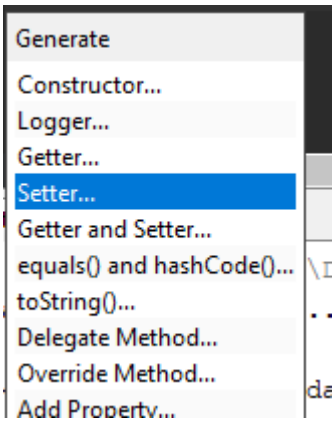


Y le digo que use el parámetro X.

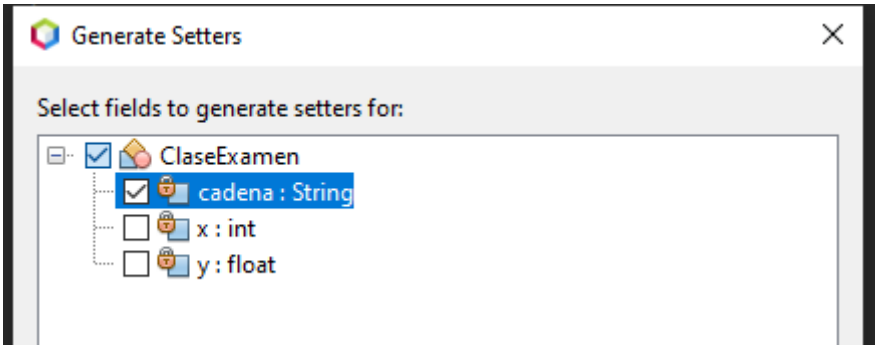


Cuando lo crea veo que devuelve en forma de int el parámetro que le he dicho que es X

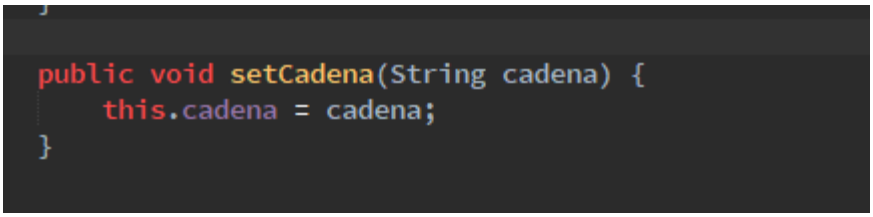
Creo un método tipo Set, asi que voy a insertar un código tipo **“Setter”**



Y le digo que use el parámetro cadena.



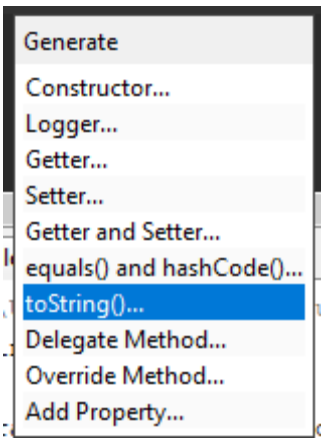
Cuando se crea, veo que ha creado un método VOID y que recoge el parámetros cadena y se lo asigna a this.cadena.



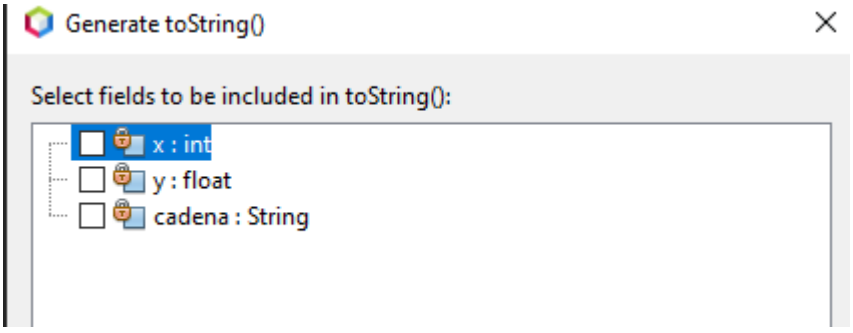
Creo este método a mano, lo que hace este método es comparar la cadena que se introduce por parámetro al invocar este método con la cadena que tiene le constructor guardada. Si es la misma devuelve **true**

```
36      public boolean funcion(String cadena) {
37          boolean aux = false;
38          if (this.cadena.equalsIgnoreCase(cadena))
39              aux = true;
40          return aux;
41      }
```

Ahora creo el método toString, para ello voy al generador de código e inserto un código toString().



No pongo ningún parámetro.



Y cuando lo creo veo que ha creado la estructura del toString().

```
42
43      @Override
45      public String toString() {
46          return "ClaseExamen{" + '}';
47      }
48
```

Añado a mano las líneas de código que faltan.

```
43          @Override
            public String toString() {
                return "ClaseExamen{" + "x=" + x +", y=" + y + ", cadena="+ cadena + '}';
46          }
47
```

El resultado de todo el código es este:

```
3   package claseexamen.claseexamen;
4
5   /**
6    *
7    * @author DAW118
8    */
9   public class ClaseExamen {
10
11      private int x;
12      private float y;
13      private String cadena;
14
15      public ClaseExamen() {
16      }
17
18      public ClaseExamen(int x, float y) {
19          this.x = x;
20          this.y = y;
21          cadena = "";
22      }
23
24      public int getX() {
25          return x;
26      }
27
28      public void setCadena(String cadena) {
29          this.cadena = cadena;
30      }
31
32
33      public boolean funcion(String cadena) {
34          boolean aux = false;
35          if (this.cadena.equalsIgnoreCase(cadena))
36              aux = true;
37          return aux;
38      }
39
40      @Override
41      public String toString() {
42          return "ClaseExamen{" + "x=" + x +", y=" + y + ", cadena="+ cadena + '}';
43      }
44  }
45
```