Clases e obxectos

As **clases** e os **obxectos** son os compoñentes fundamentais da **Programación Orientada a Obxectos** (POO). A miúdo hai confusión entre os conceptos de clase e obxecto.

Imaxinemos que temos unha tenda-obradoiro de galletas artesás. Na tenda vendemos diferentes tipos de galletas con diferentes formas, sabores e cores. Evidentemente, ao ter diferentes formas para as nosas galletas, necesitaremos varios moldes para as galletas que producimos (un molde para cada forma). Necesitaremos tamén, diferentes receitas para o sabor de cada unha e, por último, diferentes maneiras de darlle cor ás galletas.

Clases

Unha **clase** é unha entidade que determina como se comportará un obxecto e o que conterá o obxecto. Noutras palabras, é un **modelo** ou conxunto de instrucións para construír un **tipo específico** de **obxecto**.

No noso exemplo da tenda-obradoiro de galletas, a clase sería un dos moldes para galletas, xunto cos métodos para colorealas e a receita (método para preparalas). Polo tanto, no noso programa Java teremos unha maneira de especificar as características dunha galleta posto que hai diferentes moldes de galletas.

Sintaxe

```
class <nome_clase>{
    campo;
    metodo;
  }
```

En Java, primeiro deberemos crear unha clase antes de poder crear exemplares desa clase.

Obxectos

Un **obxecto** non é máis que un compoñente autónomo que consiste en **métodos** e **propiedades** que fan útil un tipo particular de datos.

Para o exemplo da tenda-obradoiro de galletas, os obxectos virían sendo canda unha das diferentes galletas obtidas dos moldes definidos (clases) e creadas por medio dun proceso ou "**construtor**" de galletas.

Sintaxe

```
NomeClase variableReferencia = new NomeClase();
```

A **crear un obxecto** chámaselle **instanciar** unha **clase**. Cando se crea un obxecto dunha clase, dise que a clase está instanciada. Todas as instancias comparten os atributos e o comportamento da

clase. Pero os **valores** deses **atributos**, é dicir, o **estado**, son únicos para cada obxecto. Unha clase pode ter calquera cantidade de instancias.

Diferenza entre clases e obxectos

Unha clase é un plan ou prototipo que define as variables e os métodos (funcións) comúns a todos os obxectos dun certo tipo.

Un obxecto é un espécime dunha clase. Os obxectos de software utilízanse a miúdo para modelar obxectos do mundo real que podemos atopar na nosa vida cotiá.

Creación de clases en Java

Unha clase contén os atributos e os métodos que conformarán ao exemplar. No momento de crear unha clase en Java, debemos especificar (no caso de que existan) os métodos ou funcións, o tipo de dato que retornan, o nome e os parámetros que reciben eses métodos.

A **estrutura básica** dunha **clase** en **Java** é a seguinte:

```
//Dámoslle o nome "UnhaClase" á clase
public class UnhaClase
{
    //Atributos da clase
    private String atributo1;
    private int atributo2;
    private float atributo3;

    //Construtor co mesmo nome da clase
    public UnhaClase(){}

    //Métodos da clase
    public void metodo1()
    {
        //Método baleiro
    }

    public String metodo2()
    {
        return "metodo2";
    }
}
```

O código anterior crea unha clase en Java chamada “UnhaClase” que ten un total de tres atributos (todos eles privados) e son de tipo String, int e float respectivamente. Ademais, esta clase ten un construtor (que sempre, por norma, debe ter o mesmo nome da clase) que non recibe ningún parámetro aínda que pode recibir todos os parámetros que desexemos. A clase ten tamén un método chamado metodo1 que non retorna valor algún (é de tipo void) e outro método chamado metodo2 que retorna unha cadea de caracteres (é de tipo String) co valor “metodo2”.

Unha clase Java pode ter ou non métodos e atributos, aínda que o máis normal é que teña tanto métodos como atributos que a caractericen.

Un construtor é un método especial dunha clase que se chama sempre que se crea un obxecto desa clase. A súa función é inicializar o obxecto e serve para asegurarnos de que os obxectos sempre conteñen valoers válidos.

Cando se crea un obxecto en Java realízanse as seguintes operacións de forma automática:

Asígnaselle memoria ao obxecto.

Inicialízanse os atributos dese obxecto cos valores predeterminados polo sistema.

Chámase a un dos construtores da clase.

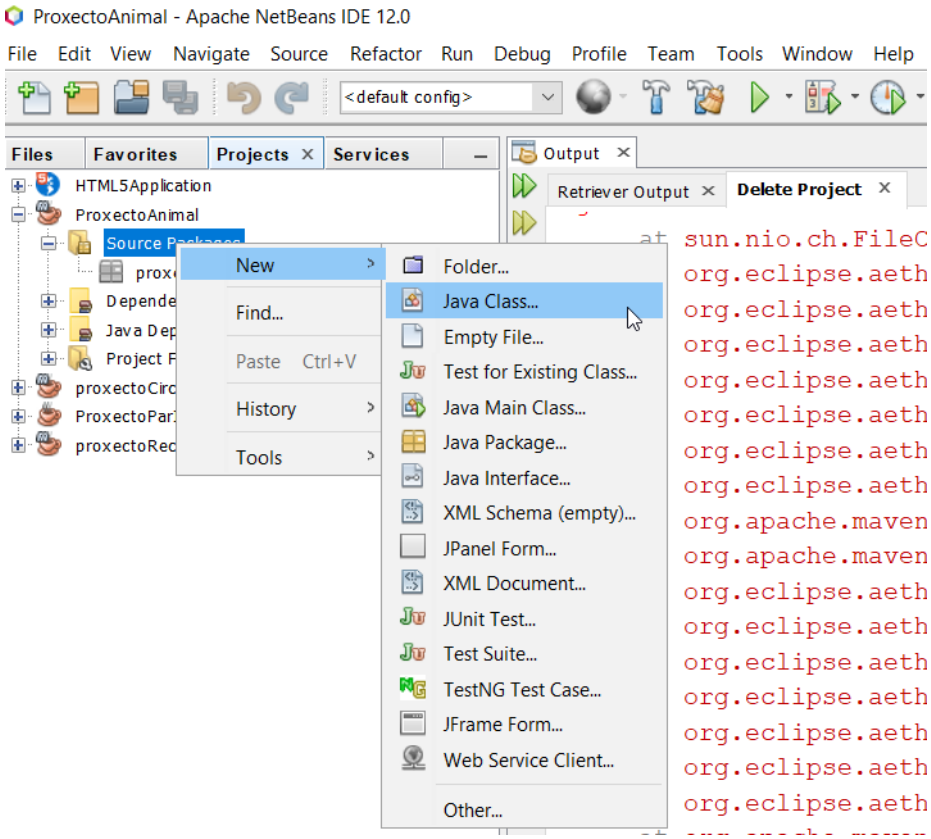
Unha clase pode ter varios construtores que teñen o mesmo nome que a clase á que pertencen pero que se diferencian polo número de argumentos. Os construtores non poden devolver ningún valor (incluíndo void).

Creación dunha clase Java en NetBeans

Imos crear unha clase Animal en NetBeans nun proxecto Maven.

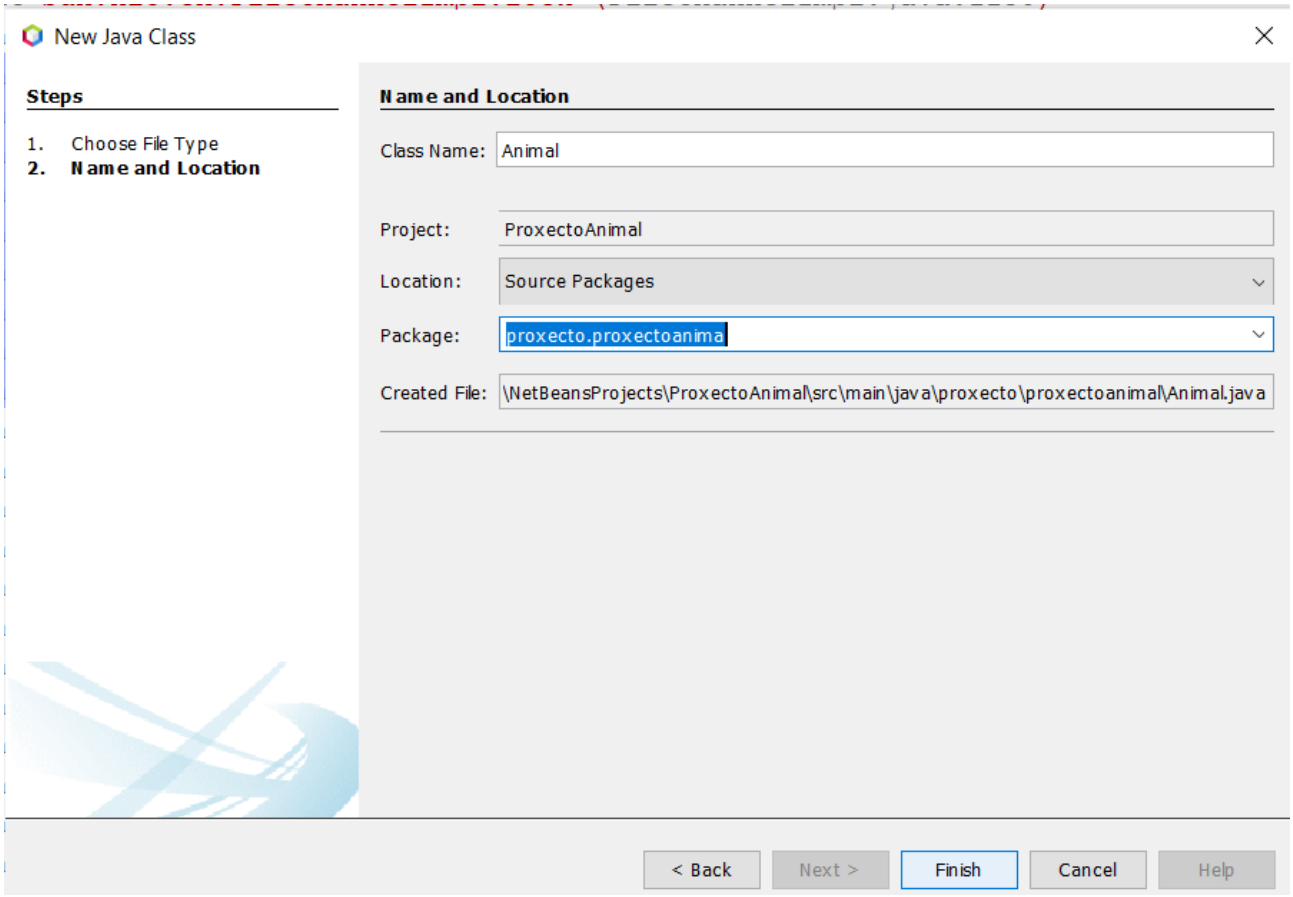
Creamos un proxecto **Java con Maven** de nome ProxectoAnimal.

Unha vez creado, imos crear a clase Animal. Podemos facelo indo ao menú Files e, estando no ProxectoAnimal, abrimos o menú contextual co botón dereito do rato e seleccionamos **New > Java Class...** tal e como se ve na seguinte imaxe:



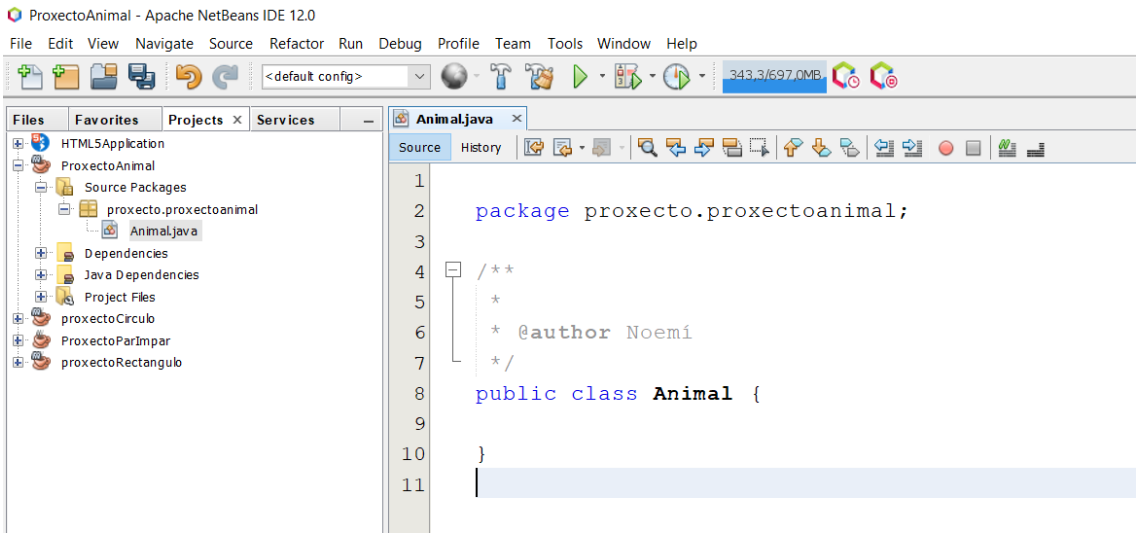
A continuación aparécenos o asistente de creación de nova clase en NetBeans e indicamos o nome que lle queremos poñer á clase (neste caso Animal) e seleccionamos o paquete no que queremos crear esa clase. Usaremos neste caso o paquete proxecto.proxectoanimal.

Recoméndase, sobre todo se imos traballar nun proxecto en equipo, que utilicemos sempre paquetes distintos do paquete por defecto para as clases. Desta maneira, aínda que haxa clases co mesmo nome, ao non estar no mesmo paquete, non haberá problemas de interferencias entre elas.



Finalmente, pinchamos no botón Finish para rematar.

Unha vez creada a clase, aparece no editor de NetBeans o esqueleto para a clase seguinte:



Podemos observar que o arquivo que se xera no NetBeans para a clase Animal chámase Animal.java.

O nome das clases en Java segue, por convención, a notación **UpperCamelCase** (máis coñecida como **PascalCase**) na que toda letra inicial de palabra debe estar en maiúsculas (Upper) e o resto da palabra en minúsculas e, non caso de que se utilicen máis palabras, a letra inicial de cada unha delas debe estar en maiúsculas e, o resto da palabra en minúsculas. Os nomes das clases non poden conter espazos en branco. Exemplo: OlaMundo.

A clase **Animal** vai ter **3 atributos**: especie, nome e idade. A clase tamén vai ter **un construtor** Animal(String nome) e tamén vai ter **tres métodos**: getIdade, setIdade e getNome.

O nome dos **atributos** e **métodos** segue, por convención, a notación **camelCase** (tamén chamada lowerCamelCase) idéntica á PascalCase excepto na primeira palabra que debe ter a primeira letra en minúsculas.

Para representar esquematicamente as clases soen usarse os **diagramas de clases UML**. Nestes diagramas unha clase represéntase utilizando unha caixa que se divide en tres zonas utilizando para facelo liñas horizontais.

No diagrama de clases a **primeira** das zonas utilízase para o **nome da clase**.

A **segunda** das zonas utilízase para escribir os atributos ou campos **da clase**.

A **terceira** das zonas úsase para escribir os **métodos** ou operacións que ofrece a **clase**.

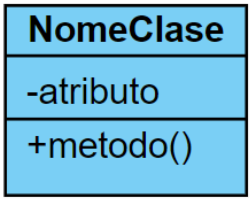
Tanto os atributos como os métodos inclúen ao principio da súa descrición a visibilidade que terá. Esta **visibilidade** identifícase escribindo un símbolo e poderá ser:

(+) **Pública**. Representa que se pode acceder ao atributo ou función desde calquera lugar da aplicación.

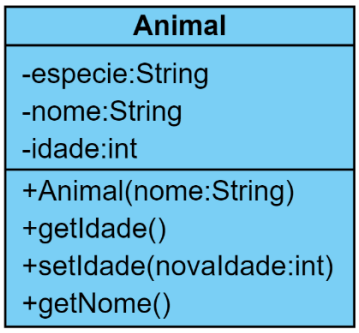
(-) **Privada**. Representa que se pode acceder ao atributo ou función unicamente desde a mesma clase.

O tipo de dato do atributo, en caso de poñerse, separase do nome do atributo con dous puntos.

Na seguinte imaxe pode verse un **diagrama de clases** xenérico dunha clase **NomeClase** cun atributo privado e un método público:

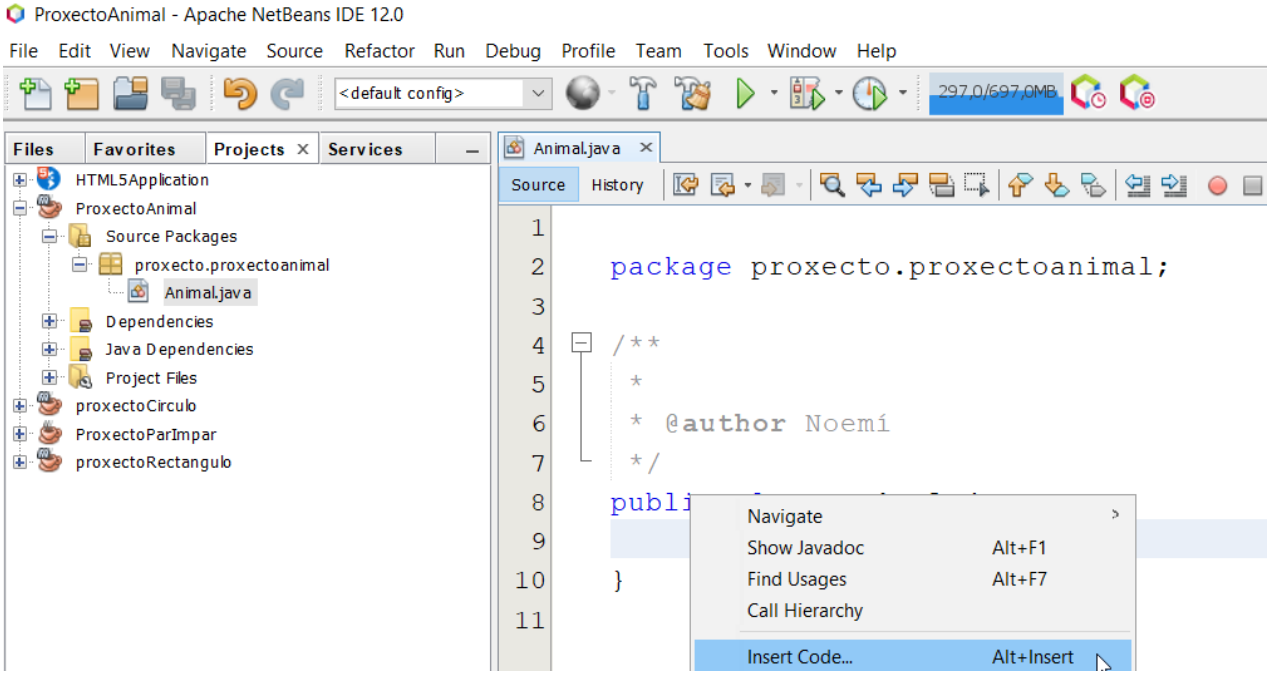


Para o caso da clase **Animal** o diagrama de clases sería o seguinte:

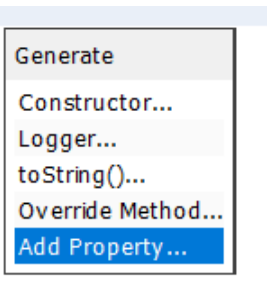


Imos crear o **atributo idade** na clase Animal usando as facilidades que nos ofrece o editor de NetBeans.

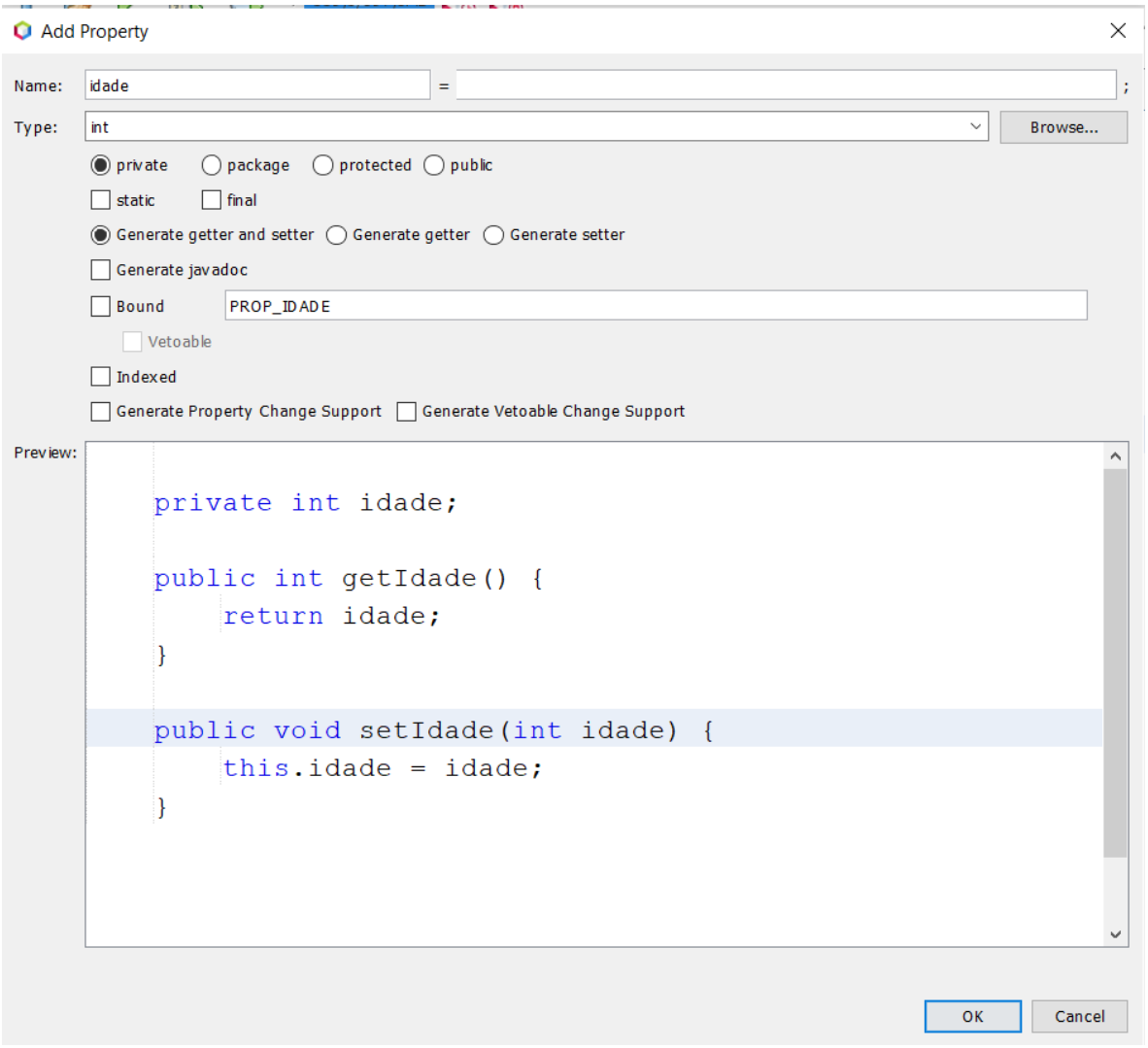
Estando dentro do código da clase facemos clic no botón dereito do rato e aparécenos o menú contextual e eliximos **Insert code...**



A continuación, aparece o menú **Generate** no que eliximos **Add Property…**



Poñemos os datos da propiedade idade, o nome, o tipo de dato (neste caso int) e deixamos que se xeren o setter e o getter tal e como se mostra na seguinte imaxe:



Tras pinchar no botón OK debe aparecer o código que se vía en Preview na clase Animal.

Facemos o mesmo para o atributo nome, fixándonos en só xerar o getter.

Finalmente, creamos de maneira manual o atributo especie.

Fixámonos en que os atributos queden ao todos seguidos na parte superior do código e os métodos todos seguidos na parte inferior do código. Isto último recoméndase para ter o código ben organizado e doado de ler para calquera.

Os setters e getters son métodos de acceso que sempre se declaran como públicos.

A convención de nomenclatura di de utilizar set+NomeAtributo para o nome do setter.

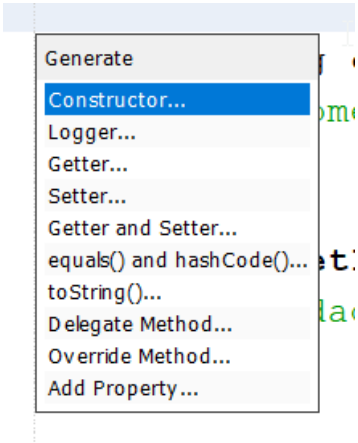
Setter (do inglés set que significa establecer, serve para asignar un valor inicial a un atributo de forma explícita. O setter nunca retorna nada (sempre é void) e só nos permite dar acceso público a certos atributos que desexemos que o usuario poda modificar.

A convención de nomenclatura di de utilizar get+NomeAtributo para o nome do getter podendo usar is+NomeAtributo no caso de que o atributo sexa de tipo boolean.

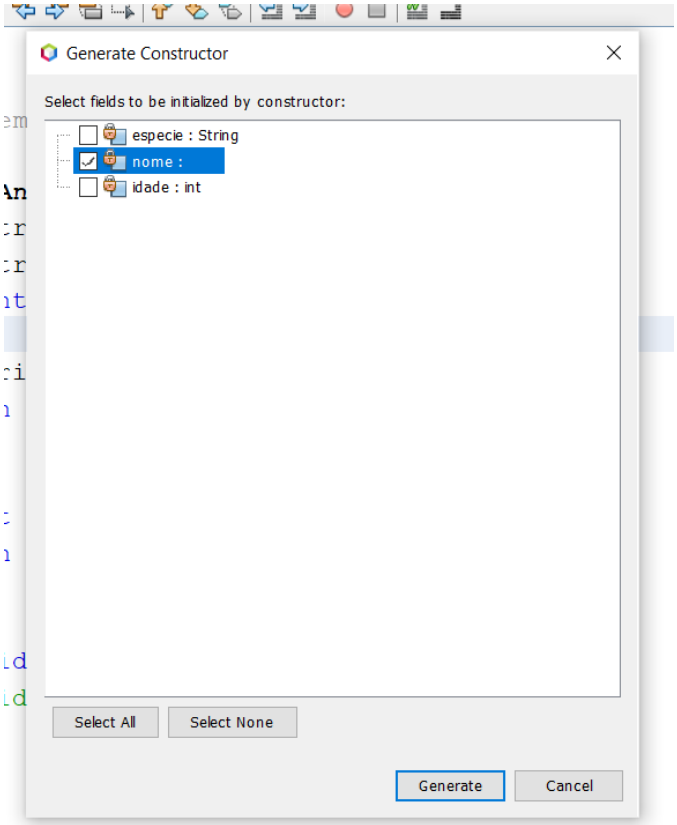
Os getters (do inglés get que significa obter) serven para obter (recuperar ou acceder) ao valor xa asignado a un atributo e utilizalo para certo método.

Para crear o **construtor** da clase co parámetro nome de tipo String en NetBeans facemos como para o atributo idade e dentro do código da clase facemos clic no botón dereito do rato e aparécenos o menú contextual e eliximos **Insert code...**

A continuación, aparece o menú **Generate** no que eliximos **Constructor…**



Aparece entón unha pantalla na que indicamos o parámetro ou parámetros que se van inicializar co construtor. Neste caso nome.



Finalmente pulsamos no botón **Generate** para xerar o código do construtor.

A clase Animal debe quedar da seguinte maneira:

```
package proxecto.proxectoanimal;//declárase o paquete

public class Animal {
    private String especie;
    private String nome;
    private int idade;

    public Animal(String nome) {
        this.nome = nome;//dáselle un nome ao animal
    }
    //método para obter o nome do animal
    public String getNome() {
        return nome;
    }
    //método para obter a idade do animal
    public int getIdade() {
        return idade;
    }
    //método para establecer a idade do animal
    public void setIdade(int idade) {
        this.idade = idade;
    }
}
```

A clase Java que temos creado neste exemplo ten por nome Animal, pertence ao paquete proxecto.proxectoanimal e ten os atributos especie nome e idade. Ademais, ten un construtor que recibe un nome que lle asigna ao animal e tres métodos encargados de obter e establecer a idade do animal e un método para obter o nome do animal.

Creación de obxectos en Java

Cando creamos obxectos en Java debemos saber o nome da clase da que imos crear o obxecto e o construtor que ten esa clase, é dicir, se o construtor recibe ou non parámetros.

Para crear obxectos en Java, a linguaxe proporciónanos o comando new. Con este comando dicímoslle a Java que imos crear un novo obxecto dunha clase específica e enviámoslle os parámetros (en caso de ser necesario) dependendo do construtor.

Imos crear un obxecto ou instancia en Java da clase UnhaClase. Esta clase ten un construtor que non recibe parámetros polo que non é necesario enviar ningún valor no momento de crear o obxecto.

```
UnhaClase unObxecto; //Declaramos unha variable do tipo da clase
unObxecto = new UnhaClase(); //Aquí xa temos creado un obxecto de UnhaClase
```

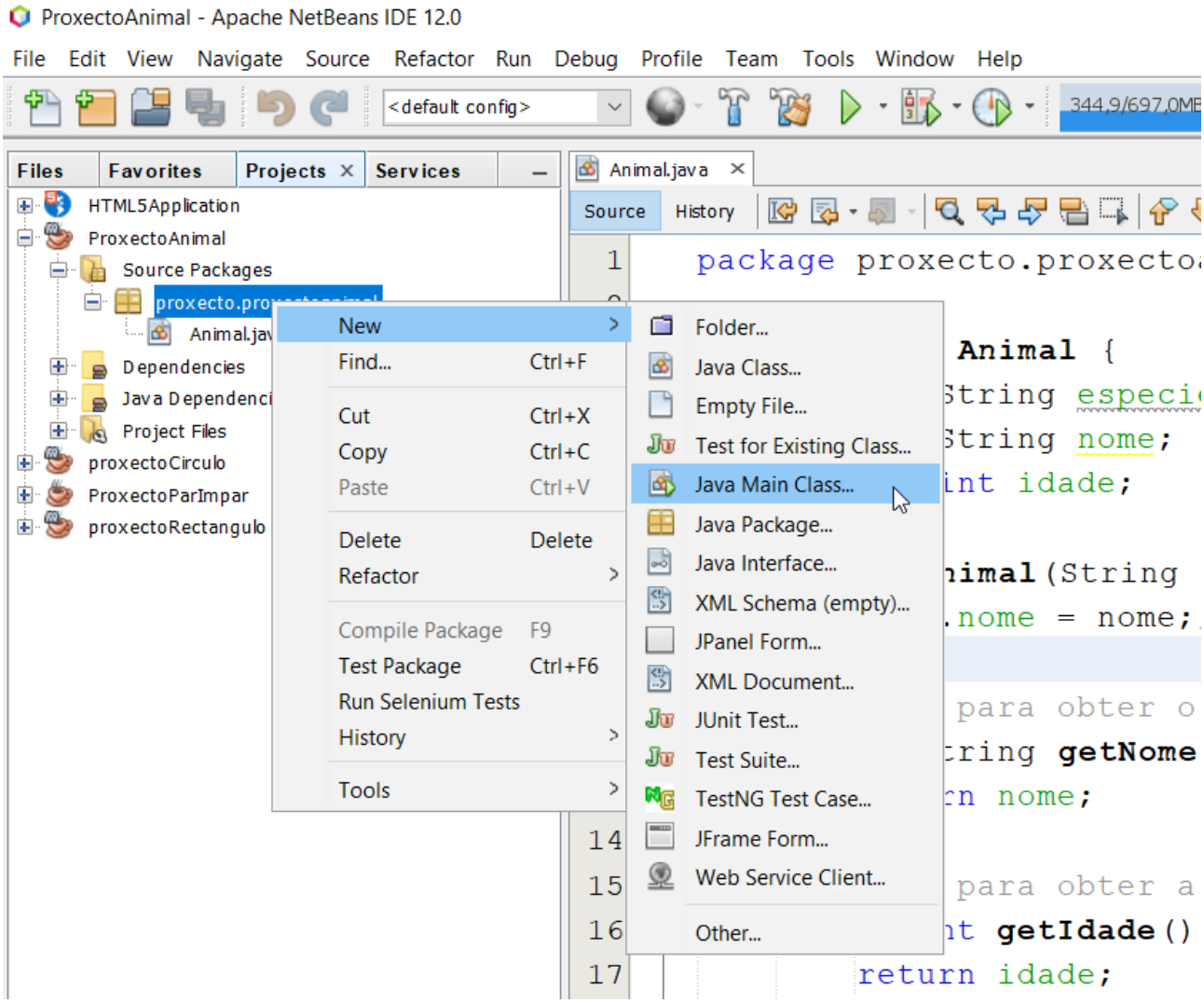
Tamén pode facerse:

```
UnhaClase unObxecto = new UnhaClase(); //Declaramos e creamos un obxecto
```

Imos crear agora un obxecto ou instancia da clase Animal.

Nesta ocasión temos un construtor que recibe un parámetro (o nome do animal) e que ten tres métodos que usaremos neste exemplo.

Para crear a clase que imos usar para este exemplo que imos chamar ProbaAnimal en NetBeans podemos situarnos no paquete na ventá Files e facer clic co botón dereito do rato para que apareza o menú contextual e eliximos **New > Java Main Class...**



Créanos un esqueleto co seguinte código:

```
package proxecto.proxectoanimal;

public class ProbaAnimal {

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

}
```

No esqueleto vemos o TODO (por facer) onde debemos colocar a lóxica da aplicación.

Para facer a proba creamos un animal Ferno ao que lle asignamos 3 anos de idade e, posteriormente, visualizamos os datos do animal creado.

O código sería o seguinte:

```
package proxecto.proxectoanimal;

public class ProbaAnimal {
    public static void main(String[] args) {
        //Creamos un animal de nome Ferno
        Animal unAnimal = new Animal("Ferno");
        //Poñémoslle unha idade de 3 anos a Ferno.
        unAnimal.setIdade(3);
        //Amosaremos o nome do animal por pantalla
        System.out.println("O nome é: " + unAnimal.getNome());
        //Amosaremos a idade do animal por pantalla
        System.out.println(" e ten " + unAnimal.getIdade() + " anos");
        //Este código deberia imprimir "O nome é: Ferno e ten 3 anos"
    }
}
```

É importante ter en conta que é análogo facer uso do método set para establecer atributos a usar o construtor enviándolle parámetros, é dicir, o resultado é o mesmo ao usar un construtor baleiro e despois establecer os valores dos atributos por medio dunha función set ou usar un construtor que reciba unha serie de parámetros que usará para establecer valores aos atributos. A diferenza radica esencialmente na axilidade e número de liñas, ao usar o construtor e enviarlle os parámetros, o construtor fai todas as asignacións por nos e usamos unha soa liña de código mentres que ao usar métodos set debemos usar un método set por cada atributo e, se temos cinco ou máis atributos xa se volve un pouco molesto traballar desta forma sobre todo se temos en conta que debemos usar un construtor de todas formas.

En todo caso, sempre é unha boa práctica declarar os métodos set e get para cada atributo pois se debemos modificar o valor dun atributo calquera, debemos facelo por medio dun método set e non creando un obxecto novo.