El concepto de Java Casting es uno de los conceptos que más cuesta entender a una persona que está empezando a programar. ¿Que es un Java Casting? .Vamos a explicarlo de forma sencilla . Para ello partiremos de dos clases , Televisor y Alexa que son dos dispositivos que podemos tener en nuestra casa . Vamos a ver el código fuente de cada uno de ellos.

```
package com.arquitecturajava;

public class Alexa {

  public void comandoVoz(String comando, String aparato) {
    System.out.println("alexa" + comando + " , el "+ aparato);
  }
}
```

```
package com.arquitecturajava;

public class Televisor {

  public void on() {
    System.out.println("el televisor se enciende");
  }
  public void off() {
    System.out.println("el televisor se apaga");
  }
  public void cambiarCanal(int canal) {
    System.out.println("el televisor cambia al canal" + canal);
  }
}
```

Como nos podemos dar cuenta las clases son muy muy sencillas . Es momento de crear dos objetos uno de cada tipo e invocar alguno de los métodos para ver que los objetos funcionan correctamente.

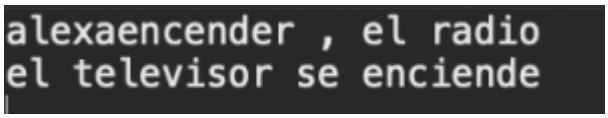
```
package com.arquitecturajava;

public class Principal {

  public static void main(String[] args) {
    Alexa a= new Alexa();
    Televisor t= new Televisor();
    a.comandoVoz("encender", "radio");

    t.on();
  }
}
```

Como vemos creamos los objetos con el operador new e invocamos al método comandoVoz del dispositivo alexa y al método on() del televisor , los mensajes aparecerán por la consola. Todo funciona correctamente.



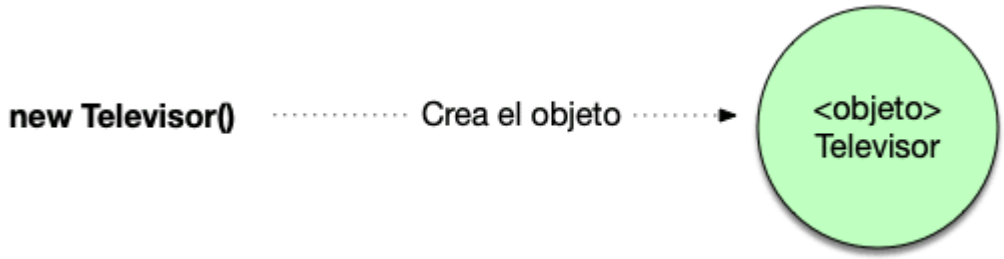
Sin embargo no todo el mundo cuando empieza entiende realmente que significada una linea de código tan sencilla como esta :

```
Televisor t= new Televisor();
```

En principio la mayoría de la gente dirá que se trata de la instanciación de un objeto en memoria . La respuesta es relativamente correcta ya que si se crea un objeto en memoria. Pero para crear un objeto en memoria es suficiente con :

```
new Televisor();
```

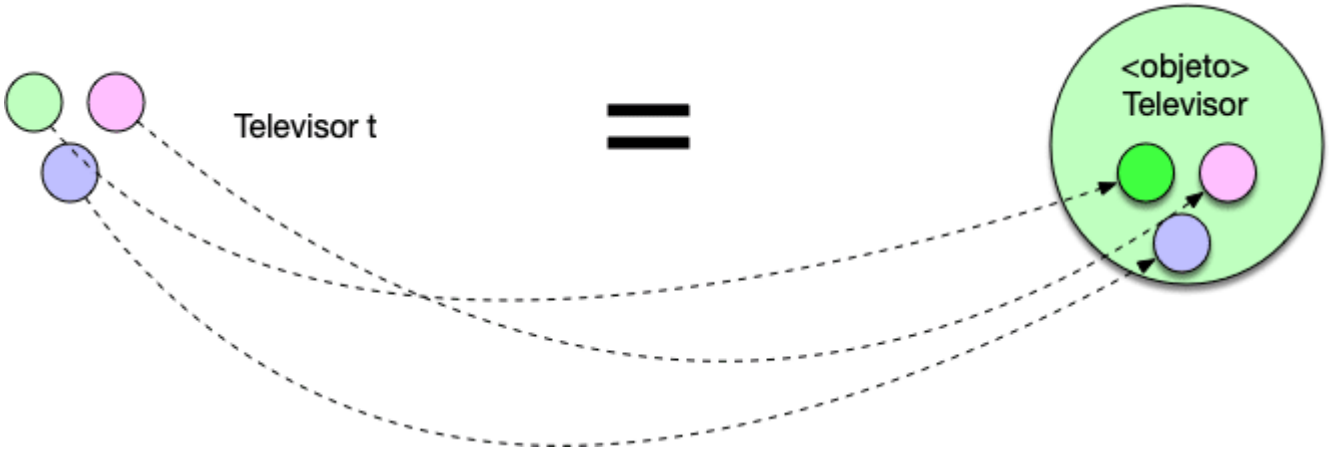
Eso genera un objeto en memoria.



Ahora bien para que vale la otra parte del código :

```
Televisor t;
```

Esta parte del código lo que hace es crear una variable en memoria de Tipo Televisor , es decir una variable con la estructura del televisor que le permite acceder a todas las propiedades y métodos que el televisor como objeto tiene en memoria.



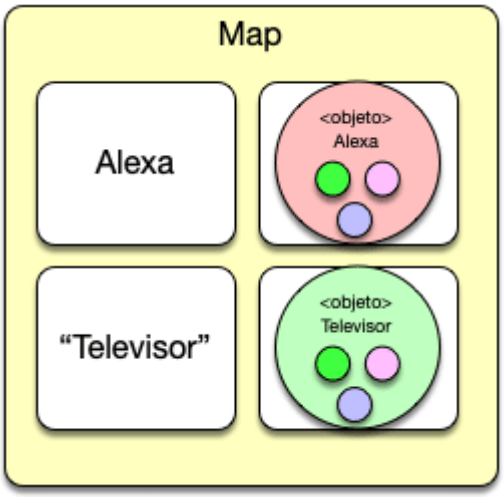
Por eso la variable es capaz de acceder a los métodos podemos entender la variable como el “mando a distancia” del televisor una vez asignados y enlazados cada uno de ellos pues trabajan en equipo . Lo mismo sucede con el dispositivo Alexa dispone de una variable y dispone de un objeto en memoria.

Java Casting y Mapas

Muchas veces es más que habitual almacenar objetos en memoria en estructuras como Listas o Mapas para su posterior reutilización , Vamos a verlo:

```
Map<String,Object> caja= new HashMap<String,Object>();
  caja.put("alexa",a);
  caja.put("mitelevisor",t);
```

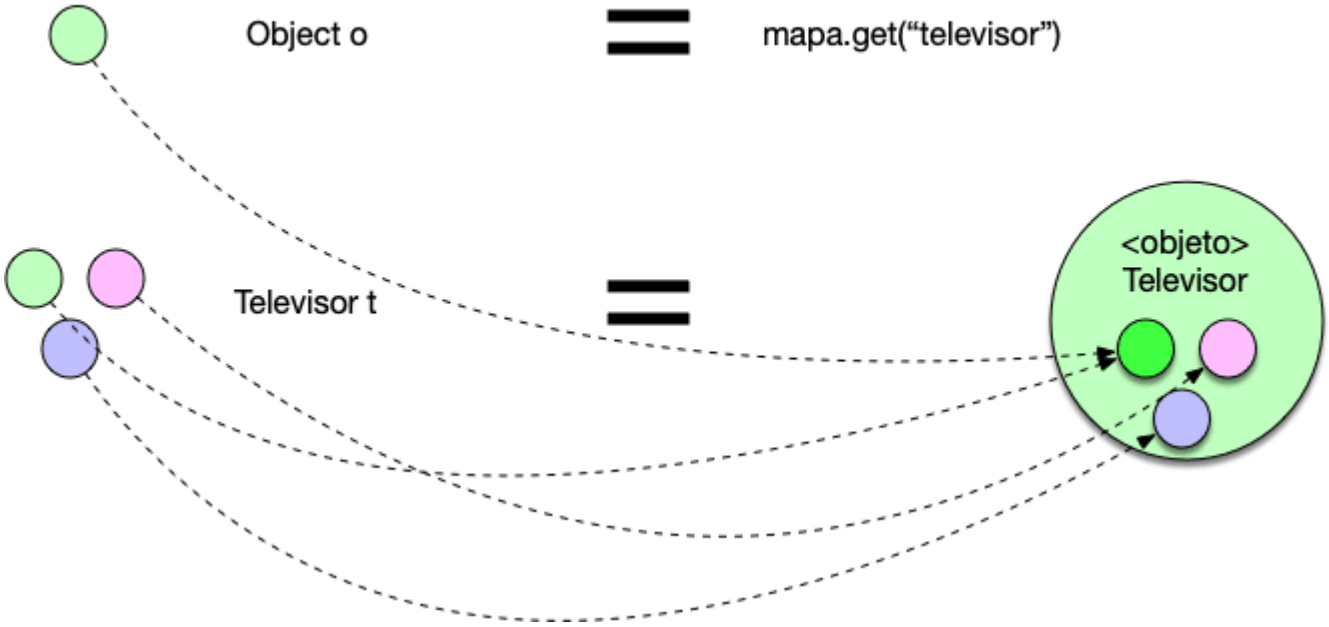
Esto es como si en la vida real acabo de añadir a una caja dos objetos y a cada uno de ellos les he asignado una etiqueta “alexa”, “mitelevisor” :



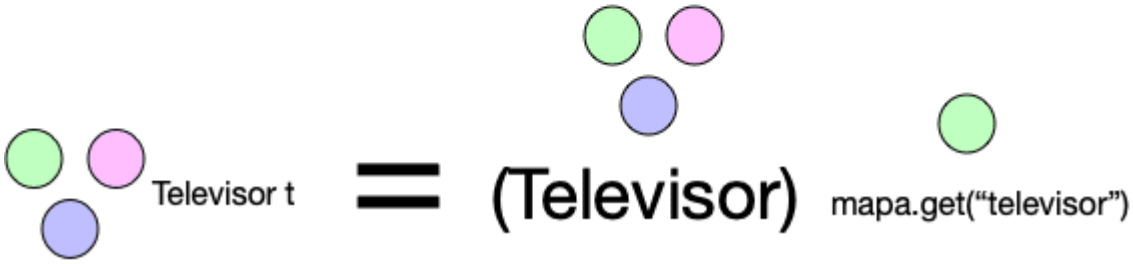
Es momento de extraer estos elementos de la caja y volver a tener acceso a ellos , para ello podemos usar el método get del Mapa o Dicccionario .

```
Televisor t2= caja.get("mitelevisor");
```

Lamentablemente si intentamos ejecutar este código simplemente no compilará . Esto se debe a que cuando un objeto sale de un mapa o de otro tipo de colección es muy común que salga como objeto genérico y no tengamos acceso a todas sus propiedades.



Recordemos que todas las clases que diseñemos son hijas de la clase Object. El problema es que nosotros no queremos un objeto genérico sino que queremos tratarlo como el televisor que "es" y asignarlo a una variable televisor. Por lo tanto nos encontramos ante una situación en la que tenemos que obligar al programa Java a ampliar el tipo de variable que maneja y realizar un "Java Casting" .



De tal forma que indicamos al compilador que realmente el objeto que extraemos de la caja no es un objeto “general” sino que asumimos que es un televisor .

```
Televisor t2= (Televisor)caja.get("mitelevisor");
    t2.on();
    t2.cambiarCanal(2);
```

De esta forma el programa compila y se ejecuta sin problemas

Otros artículos relacionados

- Java Finally y el cierre de recursos
- Java Sobrecarga de métodos y constructores
- Java Interfaces y el concepto de simplicidad