**Repaso. TypeCasting (conversión de datos)**

Realizar un “casteo” es avisarle al compilador que un objeto *X* es más específico que un objeto *Y* (herencia), y por lo tanto se puede asignar una referencia del primero al segundo.

Recordemos que la clase *Object* es la superclase de todos las clases en Java. Por lo tanto el siguiente código es válido.

|  |
| --- |
| *Object cadena1 = "Cadena de caracteres.";*  *String cadena2 = " otra cadena";*  *System.out.println(cadena1 + cadena2);* |

En este caso no es necesario el *casteo*, ya que el compilador entiende que está asignando una cadena de caracteres a un tipo más general: *Object*.

Por otro lado, la asignación inversa produce un error de compilación:

|  |  |
| --- | --- |
| *cadena2 = cadena1;* | ***Error!!!*** |

Esto se debe a que *Object* no es un tipo específico de *String*. Sin embargo en el objeto de tipo Object SI hay almacenado un objeto del tipo *String*, solamente hay que “avisarle” al compilador esta situación para poder realizar la asignación.

|  |
| --- |
| *cadena2 = (String) cadena1;* |

También produciría un error de compilación el siguiente código

|  |  |
| --- | --- |
| *cadena2 = (Integer) 120;* | ***Error!!! Un entero no puede ser un String*** |

Pero sí es válido

|  |  |
| --- | --- |
| *cadena1 = (Integer) 120;* | ***Un entero se puede asignar a un Object*** |

En cualquier caso, en tiempo de ejecución, no sabemos qué tipo de objeto puede estar almacenado en *cadena1*, ya que es del tipo *Object* y puede contener una referencia a cualquier otro tipo.

Podemos “preguntar” por el tipo de objeto almacenado en tiempo de ejecución a través de la invocación del método *getClass()*, que poseen todos los objetos.

|  |
| --- |
| *cadena1 = (Integer) 120;*  *System.out.println( cadena1.getClass().toString() );* |

O bien podemos preguntar a través del operador ***instanceof*** si el objeto es de un tipo determinado: obtendremos *true* o *false* como respuesta.

|  |
| --- |
| *cadena1 = (Integer) 120;*  *System.out.println( cadena1* ***instanceof*** *String );* |

**Otro ejemplo**

|  |  |
| --- | --- |
|  | *class Persona {*  *String Nombre;*  *}*  *class Alumno extends Persona {*  *int legajo;*  *}* |

Luego, las siguientes asignaciones son validas:

|  |
| --- |
| *Persona p = new Persona(); //alguna duda ???*  *Alumno a = new Alumno();*  *Alumno b = new Persona(); //No se puede !!!*  *Persona t = new Alumno(); //SI se puede!!!* |

**IMPORTANTE**: Incluso esta última asignación se puede realizar si fuera *Persona* una clase **abstracta**. (En este caso no se podría realizar la primera asignación *Persona p= new Persona();* )

**Tarea**: Probar el *typecasting* entre diferentes tipos numéricos entre sí; tipos de datos abstractos creados por el programador, etc.