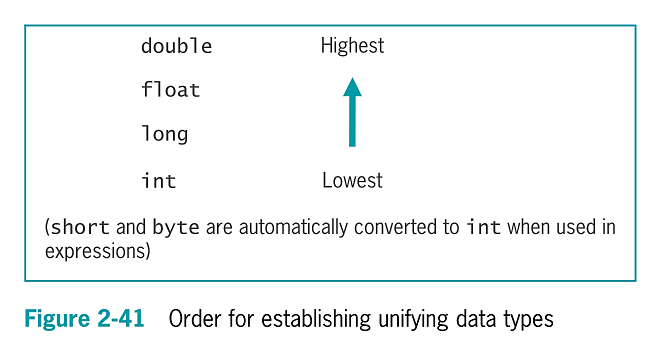
**UNDESRTANDING TYPE CONVERSION**

* Cuando se realizan operaciones aritméticas con variables o constantes del mismo tipo, el resultado de la operación conserva/guarda el mismo tipo.
* Por ejemplo, cuando divides dos enteros, el resultado es un entero. Y cuando restamos dos doubles, el resultado es un double.
* En ocasiones, se necesita realizar operaciones matemáticas con operandos de tipos diferentes. (int y double)
* El proceso de convertir un tipo de dato a otro se conoce como **type conversion.**
* Java realizar algunas conversiones automáticamente de manera implícita, pero otras conversiones deben ser solicitadas explícitamente por el programador.

**AUTOMATIC TYPE CONVERSIÓN**

* Cuando realizamos operaciones aritméticas con operandos de diferentes tipos, java elige un **tipo unificador (unifying type)**
* El **tipo unificador** es el tipo de dato al cual todos los operadores en una expresión son convertidos de tal manera que sean compatibles entre ellos.
* Java realiza una **conversación implícita**, esto significa que convierte los datos diferentes al tipo unificador
* Las conversiones implícitas también son llamadas **promotions**.
* La siguiente figura muestra el orden para establecer tipos unificados entre valores.

****

* Cuando dos tipos de datos diferentes son usados en una expresión, el tipo unificador es el que está en la parte más alta de la lista.
* Cuando un operando que está en la parte más baja de la lista es combinado con un tipo que está más alto, el operador de tipo más bajo es convertido al más alto.
* Por ejemplo: la suma de un double y un int resulta en un double.
* La resta de un long a un float resulta un float.
* Por ejemplo: asumiendo que un entero (horasTrabajadas) y un double (pagoPorHora) son definidos y multiplicados como se muestra a continuación:

|  |
| --- |
|  |

* El resultado de la multiplicación es un double, porque cuando un double y un int son multiplicados, el int es **promocionado** al rango más alto de tipo unificador (double, el tipo que está más alto en la lista). Por lo tanto, asignarlo a la variable sueldo es legal.
* El siguiente código no compilará porque y pagoPorHora es double, y java no permite la pérdida de precisión que ocurre si intentamos almacenar un valor calculado a partir de datos doubles en una variable int.

|  |
| --- |
|  |

* Los datos **char**, **short** y **byte** son promocionados a int cuando son utilizados en sentencias o expresiones con tipos de datos diferentes.
* Si realizamos una combinación de valores char, short y byte, el resultado es un int por defecto.
* Por ejemplo, si sumamos dos bytes, el resultado es un int, no un byte.

**EXPLICIT TYPE CONVERSIONS**

* podemos sobre escribir, a propósito, el tipo unificador impuesto por java realizando un **cast**
* **Casting** fuerza a un valor de un tipo de dato a ser utilizado como valor de otro tipo de dato.
* Para realizar una operación de tipo **cast,** utilizamos el operador **cast**, el cual es creado poniendo el tipo de resultado deseado entre paréntesis.
* Usar el operador **cast** es una **conversión explícita.**
* El operador cast va seguido de la variable o constante a ser casteada.
* Por ejemplo:

|  |
| --- |
|  |

* El presupuesto semanal es 47,415, una cuarta parte de saldoBancario.
* En este ejemplo, al valor double de saldo bancario es dividido por el int 4, y el resultado es un double
* Después, el resultado double es convertido a float antes de que se almacenado en presupuestoSemanal.
* Sin la conversión, la sentencia que asigna el resultado a presupuestoSemanal no compilaría.
* Similarmente, un cast de float a int ocurre en el siguiente segmento de código.

|  |
| --- |
|  |

* dolares es 47, la parte entera de miDinero.
* En este ejemplo, el valor float de miDinero es convertido a un int antes de que se almacenado en la variable int dolares.
* Cuando el valor float es convertido a un int, los decimales son perdidos.
* La operación cast no altera ninguna variable, la alteración ocurre solamente durante la operación actual realizada.
* Si la miDinero fuera utilizada de nuevo posteriormente, seguirá conteniendo su valor float 47.82.
* No necesitamos o no es necesario realizar el cast cuando asignamos un valor pequeño a un tipo unificado más alto.
* Por ejemplo: ***double tarifaPago = 10;***
* Java automáticamente promociona la constante integer 10 a ser un double para que pueda ser almacenado en la variable tarifaPago.
* Sin embargo, para una mejor claridad, si deseamos asignar 10 a tarifaPago, es preferible escribirlo de la siguiente manera; ***double tarifaPago = 10.0;***
* El resultado es idéntico, ya sea que le asignemos el valor double 10.0 o el valor int 10 a la variable double.