Программирование на языке Java

Тема 45. Преобразование типов

Java – строго типизированный язык

Java – язык строго типизированный язык.
Компилятор и виртуальная машина всегда следят за работой с типами, гарантируя надежность выполнения программы.

Однако, есть случаи, когда нужно осуществить то или иное для реализации логики программы.

Преобразование к long

int

Сложение int и char

Сложение int и long

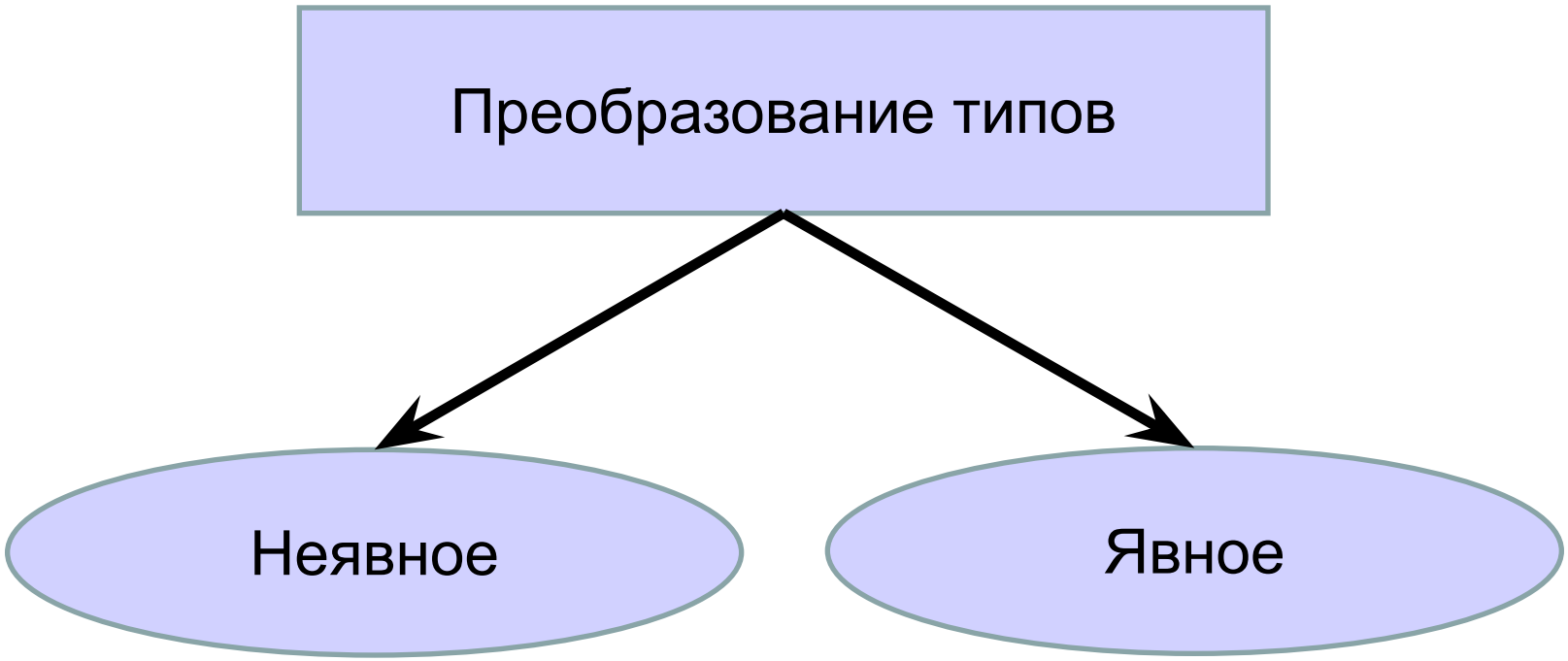
Преобразование к double

```
a = 5 + 'A' + a;
System.out.print("a="+Math.round(a/2.0));
```

Преобразование к String

Преобразование к int

Преобразование типов



Неявное преобразование типов

Выполняется в случае, если выполняются условия:

1. Оба типа совместимы;
2. Длина целевого типа больше длины исходного типа.

Пример. Преобразование от **`byte`** к **`int`**.

Явное преобразование типов

Общая форма явного преобразования

```
(<целевой тип>) <значение>
```

1. Если длина целевого типа меньше длины исходного типа, то происходит **преобразование с сужением**.
2. Если значение переменной вещественного типа присваивается переменной целого типа, то выполняется **усечение** (отбрасывается дробная часть).

Сужение

Пример 1. Преобразование от **`int`** к **`short`**.

Пример 2. Преобразование от **`float`** к **`int`**.

Усечение

Преобразование типов. Пример

```
int b = 1;
byte a = b;
byte c = (byte)-b;
int i = c;
```

Ошибка!

Явное преобразование переменной типа `int` к типу `byte`

Автоматическое неявное преобразование переменной типа `byte` к типу `int`

Виды приведений

- тождественное (**`identity`**);
- расширение примитивного типа (**`widening primitive`**);
- сужение примитивного типа (**`narrowing primitive`**);
- расширение объектного типа (**`widening reference`**);
- сужение объектного типа (**`narrowing reference`**);
- преобразование к строке (**`String`**);
- запрещенные преобразования (**`forbidden`**).

Тождественное преобразование

В **Java** преобразование выражения любого типа к точно такому же типу **всегда** допустимо и успешно выполняется.

Для чего нужно такое преобразование?

- Любой тип в **Java** может участвовать в преобразовании, хотя бы в тождественном.

Тип **boolean** можно привести только к **boolean**.

- Облегчение чтения кода для разработчика.

```
print(getCity().getStreet().getHouse().getFlat());

print((Flat)(getCity().getStreet().getHouse().
   getFlat()));
```

Преобразование типов

Простой тип, расширение	Объектный тип, расширение
Простой тип, сужение	Объектный тип, сужение

Расширение простого типа

Расширение простого типа – переход от менее емкого типа к более емкому.

Это преобразование **безопасно** – новый тип вмещает все данные, которые хранились в старом.

Не происходит потери данных.

Например, от типа **`byte`** (1 байт) к типу **`int`** (4 байта)

```
byte b = 3;
int a = b;
```

Типы данных

Тип	Размер	Min	Max
`byte` (байт)	8 бит	-128	+127
`short` (короткое целое)	16 бит	-2^{15}	$2^{15}-1$
`int` (целое число)	32 бита	-2^{31}	$2^{31}-1$
`long` (целое число двойного размера)	64 бита	-2^{63}	$2^{63}-1$
`char` (символьный)	16 бит	0	$2^{16}-1$
`float` (вещественные)	32 бита	3.4e-038	3.4e+038
`double` (вещественные двойной точности)	64 бита	1.7e-308	1.7e+308

Расширения простого типа

- от **byte** к **short, int, long, float, double**
- от **short** к **int, long, float, double**
- от **char** к **int, long, float, double**
- от **int** к **long, float, double**
- от **long** к **float, double**
- от **float** к **double**

19

Сколько существует расширяющих преобразований простого типа?

Почему?

Нельзя провести расширение

- от типов **byte** и **short** к типу **char**
- от типа **char** к типу **short**

Искажения при расширении

У дробных типов значащих разрядов (разрядов, отведенных на представление мантиссы) меньше, чем у некоторых целых типов, поэтому возможны **искажения значений** при:

- приведении значений **`int`** к типу **`float;`**
- приведении значений типа **`long`** к типу **`float`** или **`double`**.

```
long a=11111111111L;
float f = a;
System.out.println(f);
a = (long) f;
System.out.print(a);
```

```
1.11111111e11
```

```
11111111056
```

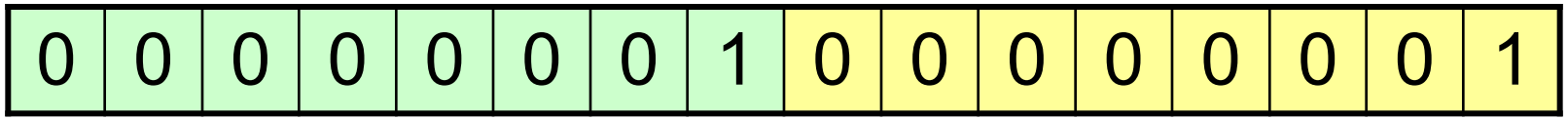
Сужения простого типа

Сужение простого типа – переход осуществляется от более емкого типа к менее емкому.

Риск потерять данные! При сужении целочисленного типа все старшие биты отбрасываются.

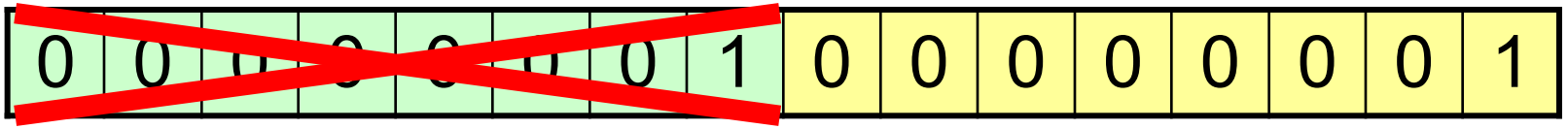
16 бит

```
short a = 257;
```



8 бит

```
byte b = (byte)a;
```




```
System.out.print(b);
```

```
1
```

Сужения простого типа

- от **byte** к **char**
- от **short** к **byte**, **char**
- от **char** к **byte**, **short**
- от **int** к **byte**, **short**, **char**
- от **long** к **byte**, **short**, **char**, **int**
- от **float** к **byte**, **short**, **char**, **int**, **long**
- от **double** к **byte**, **short**, **char**, **int**, **long**, **float**

Сужения простого типа. Примеры

```
System.out.print((byte)383);

System.out.print((byte)384);

System.out.print((byte)-384);
```

127

-128

-128

Кроме значения может быть потерян знак

```
char c=40000;
System.out.print((short)c);
```

-25536

Сужение дробного типа до целочисленного

Не число (`Not A Number`)

1. Дробное значение преобразуется в **`long`**, если целевым типом является **`long`**, или в **`int`** – в противном случае, т.е. дробная часть отбрасывается.
   - Если исходное число – **`NaN`**, то результат 0.
   - Если исходное число – положительная или отрицательная бесконечность, то результат – максимальное или минимальное значение выбранного типа.
   - Если дробное число конечной величины, но после округления получилось слишком большое по модулю число выбранного типа, то результатом будет максимальное или минимальное число выбранного типа.
   - Если результат округления уложился в диапазон типа, то он и будет результатом.

Сужение дробного типа до целочисленного

2. Производится дальнейшее сужение от выбранного целочисленного типа к целевому, если это необходимо.

Сужение дробного типа. Пример

```
float fmin = Float.NEGATIVE_INFINITY;
float fmax = Float.POSITIVE_INFINITY;
print("long:"+(long)fmin+".."+(long)fmax);
print("int:"+(int)fmin+".."+(int)fmax);
print("short:"+(short)fmin+".."+(short)fmax);
print("char:"+(char)fmin+".."+(char)fmax);
print("byte:"+(byte)fmin+".."+(byte)fmax);
```

long: -9223372036854775808..9223372036854775807

int: -2147483648..2147483647

short: 0..-1

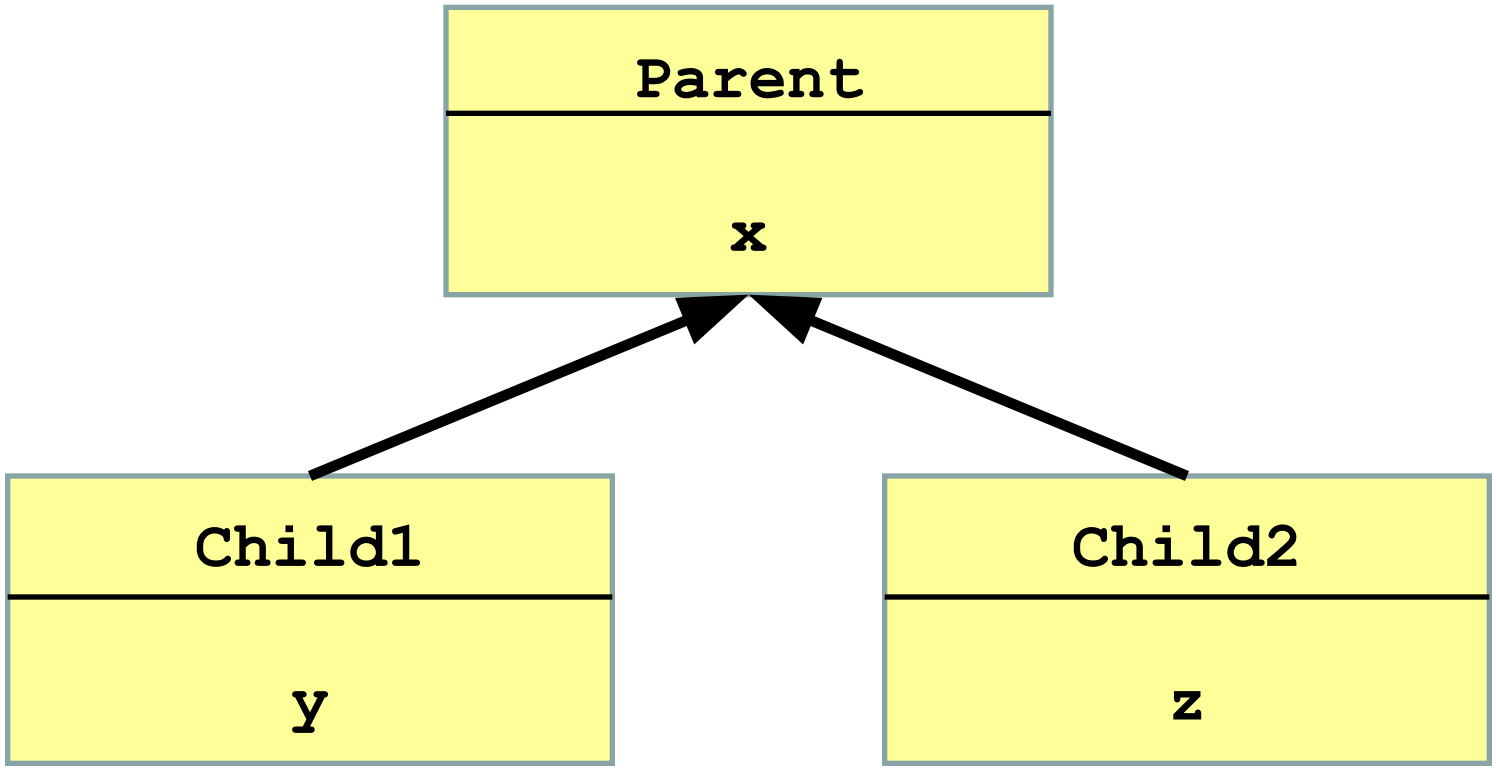
char: 0..65535

byte: 0..-1

Преобразование объектных типов

```
// Объявляем класс Parent
class Parent {
int x;
}
// Объявляем класс Child и наследуем
// его от класса Parent
class Child1 extends Parent {
int y;
}
// Объявляем второго наследника
// класса Parent - класс Child2
class Child2 extends Parent {
int z;
}
```

Схема наследования



Расширение объектных типов

Расширение – переход от более конкретного типа к менее конкретному, т.е. переход от детей к родителям.

Производится JVM автоматически.

Пример. Преобразование от наследника (**`Child1`**, **`Child2`**) к родителю (**`Parent`**).

```
Parent p1 = new Child1();
Parent p2 = new Child2();
```

Внимание! С помощью ссылок **`p1`** и **`p2`** можно обращаться только к полю **`x`**, а поля **`y`** и **`z`** недоступны.

Расширение объектных типов

1. От класса **A** к классу **B**, если **A** наследуется от **B** (важным частным случаем является преобразование от любого ссылочного типа к **Object**);
2. От **null**-типа к любому объектному типу.

Значение **null** может принять переменная любого ссылочного типа. Это означает, что ее ссылка никуда не указывает, объект отсутствует.

```
Parent p = null;
```

Пустая ссылка

Сужение объектных типов

Сужение – переход от менее конкретного типа к более конкретному, т.е. переход от родителей к детям.

Внимание! Такой переход не всегда возможен.

```
Parent p = new Child1();
Child1 c = (Child1)p;

Parent p2 = new Child2();
Child1 c2 = (Child1)p2;
```

Успешное преобразование

Ошибка!

Оператор instanceof

Оператор **instanceof** проверяет принадлежность объекта указанному типу, возвращает значение типа **boolean**.

Объект

Класс

```
p instanceof Parent
```

```
Parent p = new Child1();
if (p instanceof Child1) {
  Child1 c = (Child1)p; }
Parent p2 = new Child2();
if (p2 instanceof Child1) {
  Child1 c = (Child1)p2; }
Parent p3 = new Parent();
if (p3 instanceof Child1) {
  Child1 c = (Child1)p3; }
```

Преобразование выполнено

Преобразование не выполнено

Преобразование не выполнено

Сужение объектных типов

1. От класса **`A`** к классу **`B`**, если **`B`** наследуется от **`A`** (важным частным случаем является сужение типа **`Object`** до любого другого ссылочного типа).

Преобразование к строке

Любой тип может быть преобразован к строке, т.е. к экземпляру класса **`String`**.

- Числовые типы записываются в текстовом виде без потери точности представления.
- Булевская величина приводится к строке "**`true`**" или "**`false`**" в зависимости от значения.
- Для объектных величин вызывается метод **`toString`**(). Если метод возвращает **`null`**, то результатом будет строка "**`null`**".
- Для **`null-значения`** генерируется строка "**`null`**".

Запрещенные преобразования

Внимание! Попытка осуществить запрещенное преобразование вызовет ошибку компиляции.

Запрещенные преобразования:

- Переходы от любого объектного типа к примитивному
- Переходы от примитивного типа к объектному
- Тип **`boolean`** нельзя привести ни к какому другому типу, кроме **`boolean`** (за исключением приведения к строке).
- Невозможно привести друг к другу типы, находящиеся не на одной, а на соседних ветвях дерева наследования
- и другие.

Применение приведений

1. Присвоение значений переменным (**`assignment`**).
2. Вызов метода. Это преобразование применяется к аргументам вызываемого метода или конструктора. Явное приведение. В этом случае явно указывается, к какому типу требуется привести исходное значение. Допускаются все виды преобразований, кроме приведений к строке и запрещенных.
3. Оператор конкатенации производит преобразование к строке своих аргументов.
4. Числовое расширение (**`numeric promotion`**). Числовые операции могут потребовать изменения типа аргумента(ов).

Автоматическое повышение типов

Правила повышения типа:

1. Тип значения **`byte`**, **`short`** и **`char`** повышаются до **`int`**.
2. Если один из операндов имеет тип **`long`**, то тип всего выражения повышается до **`long`**.
3. Если один из операндов имеет тип **`float`**, то тип всего выражения повышается до **`float`**.
4. Если один из операндов имеет тип **`double`**, то тип всего выражения повышается до **`double`**.