Лекция 7

Приведение примитивных типов: расширение и сужение.

Типизация примитивных типов

- В Java у каждого объекта и у каждой переменной есть свой жёстко заданный неизменяемый тип. Тип переменной определяется ещё в процессе компиляции программы, тип объекта — при его создании. Тип нового созданного объекта и/или переменной остаются неизменными все их время жизни.

Код на Java	**Описание**

`int a = 11;`
`int b = 5;`
`int c = a / b; // c == 2`
	a / b — это деление нацело. Ответом будет два. Остаток от деления просто отбрасывается.

`int a = 13;`
`int b = 5;`
`int d = a % b; // d == 3`
	В d будет храниться остаток от деления a на b нацело. Остаток равен 3.

- Есть и пара интересных нюансов, которые следует помнить.

- Во-первых, ссылочная переменная не всегда хранит значение такого же типа, как и она.

- Во-вторых, при взаимодействии переменных двух разных типов, они должны быть сначала преобразованы к одному общему типу.

- При делении двух целых чисел, результатом тоже будет целое число. Если разделить 5 на 3, то ответом будет 1 и два в остатке. Остаток при этом отбрасывается.

- Если разделить 1 на 3, то получится 0 (и единицу в остатке, который отбросится). Если вы хотите получить 0.333, то в Java числа перед делением лучше всего приводить к вещественному (дробному) типу путем умножения их на вещественную единицу – 1.0

	Код на Java	Описание
1	**int** a = 1/3;	a будет содержать 0
2	**double** d = 1/3;	d будет содержать 0.0
3	**double** d = 1.0 / 3;	d будет содержать 0.333(3)
4	**double** d = 1 / 3.0;	d будет содержать 0.333(3)
5	**int** a=5, b=7; **double** d = (a*1.0) / b;	d будет содержать 0.7142857142857143

Список базовых типов данных

	Тип	Размер, байт	Диапазон значений	Значение по умолчанию	Описание
1	byte	1	-128 .. 127	0	Самое маленькое целое — один байт
2	short	2	-32,768 .. 32,767	0	Короткое целое, два байта
3	int	4	— 2*10^9 .. 2*10^9	0	Целое число, 4 байта
4	long	8	— 9*10^18 .. 9*10^18	0L	Длинное целое, 8 байт
5	float	4	-10^127 .. 10^127	0.0f	Дробное, 4 байта
6	double	8	-10^1023 .. 10^1023	0.0d	Дробное, двойной длины, 8 байт
7	boolean	1	true, false	false	Логический тип (только true & false)
8	char	2	0..65,535	\u0000 ‘	Символы, 2 байта, все больше 0
9	Object	4	Любая ссылка или null.	null	Хранит ссылки на объекты типа Object или классов наследников

- Тип byte — это самый маленький тип целых чисел. Каждая переменная этого типа занимает всего один байт памяти, поэтому он может хранить значения в диапазоне от -128 до 127.

- Тип **short** — ровно в два раза длиннее типа byte и тоже хранит только целые числа. Самое большое число, которое в него вмещается — это 32767. Самое большое отрицательное число — это -32768.

- Тип **int** может хранить целые числа до двух миллиардов, как положительные, так и отрицательные.

- Тип **float** — создан для хранения вещественных (дробных) чисел. Его размер 4 байта. Все дробные числа хранятся в памяти в очень интересной форме.
  Например, число 987654.321 можно представить как $0.987654321*10^{6}$. Поэтому в памяти оно будет представлено как два числа «0.987654321» (**мантисса — значащая часть числа**) и «6» (**экспонента — степень десятки**)
  Такой подход позволяет хранить гораздо большие числа, чем int, используя всего 4 байта. Но при этом мы жертвуем точностью. Часть памяти расходуется на хранение мантиссы, поэтому такие числа хранят всего 6-7 знаков после запятой, остальные отбрасываются.
  Такие числа еще называют «числа с плавающей запятой» или «числа с плавающей точкой (**float**ing point number)». Отсюда, кстати, и название типа — **float**.

- Тип **double** – это такой же тип, как и **float**, только в два раза длиннее – он занимает восемь байт. (**double** – двойной, по-английски). И предельный размер мантиссы и количество значащих цифр в нем больше. Если вам нужно хранить вещественные числа – старайтесь использовать именно этот тип.

- Тип **char** – гибридный тип. Его значения можно интерпретировать и как числа (их можно складывать и умножать) и как символы. Так было сделано потому, что хоть символы и имеют визуальное представление, для компьютера они в первую очередь просто числа. И работать с ними как с числами гораздо удобнее. Тут еще есть одно замечание: тип **char** строго положительный – отрицательных значений он хранить не может.

- Тип **boolean** – логический тип и может хранить всего два значения: **true** (истина) и **false**(ложь).

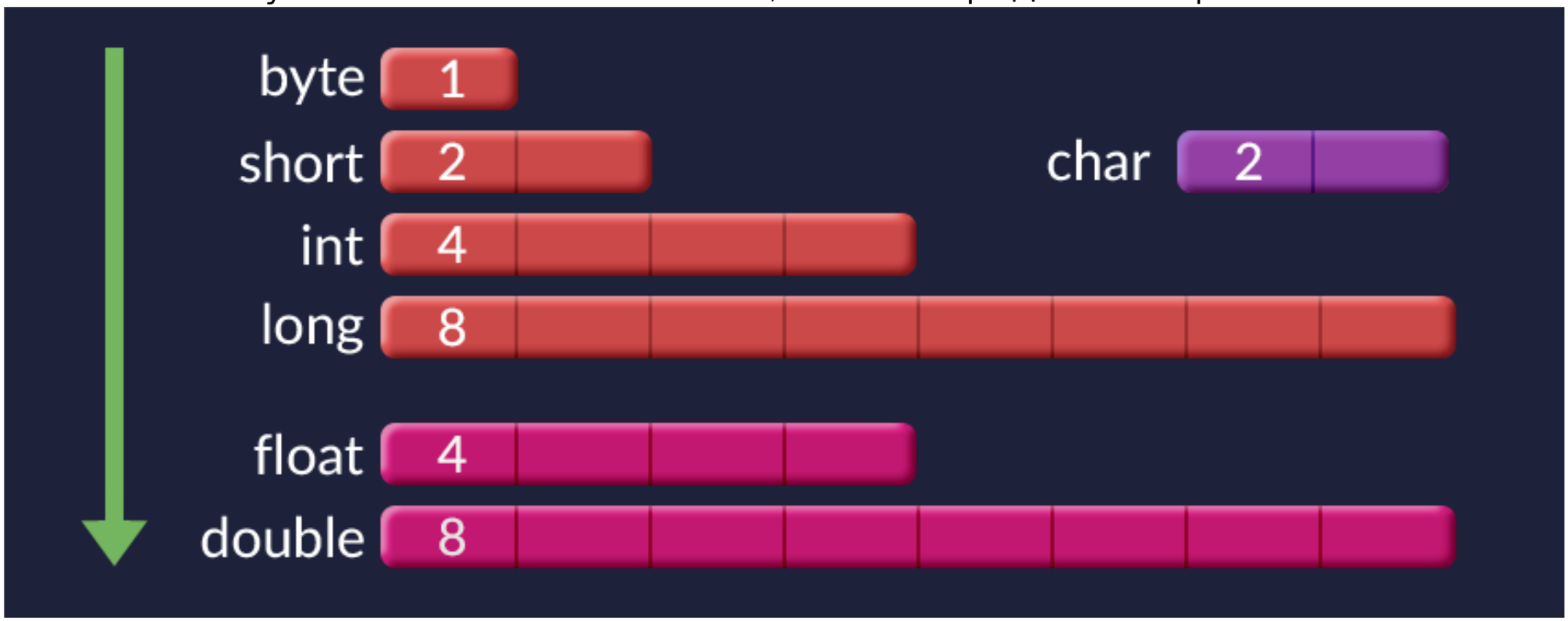
- Тип **Object**, хоть и представлен в таблице, примитивным типом не является. Это базовый класс для всех классов в Java. Во-первых, все классы считаются унаследованными от него, а значит, содержат его методы. А во-вторых, ему можно присваивать ссылки на объекты любого типа. В том числе и **null** – пустую ссылку.

Преобразование типов

- Хоть типы переменных всегда неизменны, есть место, где можно проводить преобразование типов. И место это — **присваивание**.

- Можно присваивать друг другу переменные разных типов. При этом значение, взятое из переменной одного типа, будет преобразовано в значение другого типа и присвоено второй переменной.

- В связи с этим, можно выделить два вида преобразования типов: расширение и сужение. Расширение похоже на перекладывание из маленькой корзинки в большую — операция проходит незаметно и безболезненно. **Сужение типа** — это перекладывание из большой корзинки в маленькую: места может не хватить, и что-то придётся выбросить.



- Тут есть пара замечаний:
  1. **char** такая же корзинка, как и **short**, но свободно перекладывать из одной в другую нельзя: при перекладывании значения из **short** в **char**, всегда будут теряться значения меньше 0. При перекладывании из char в short будут теряться значения больше 32-х тысяч.
  2. При преобразовании из целых чисел в дробные могут отбрасываться самые младшие части числа. Но т.к. смысл дробного числа в том, чтобы хранить приблизительное значение, то такое присваивание разрешается.

- При сужении типа нужно явно показать компилятору, что вы не ошиблись и отбрасывание части числа сделано намеренно. Для этого используется оператор приведения типа. Это **имя типа в круглых скобочках.**

	Код на Java	Описание
1	**byte** a = 115; **int** b = a;	Расширение типа. Все отлично.
2	**int** c = 10000; **byte** d = (**byte**) c;	Сужение типа. Нужно явно отбросить лишние байты.
3	**int** c = 10; **byte** d = (**byte**) c;	Сужение типа. Нужно явно отбросить лишние байты, даже если они равны 0.
4	**float** f = 10000; **long** l = (**long**) (f * f); **float** f2 = l; **long** l2 = (**long**) f2;	При присваивании к float, происходит расширение типа. При присваивании значения float к long, происходит сужение – необходимо приведение типа.
5	**double** d = 1; **float** f = (**float**) d; **long** l = (**long**) f; **int** i = (**int**) l; **short** s = (**short**) i; **byte** b = (**byte**) s;	Сужение во всех операция присваивания, кроме первой строки, требует указания явного преобразования типа.

- **Оператор приведения типа** нужно указывать перед числом/переменной каждый раз, когда происходит отбрасывание части числа или сужение типа. Он действует только на число/переменную, которое идет непосредственно за ним.

	Код на Java	Описание
1	**float** f = 10000; **long** l = (**long**) f * f;	К типу long приводится только одна переменная из двух: умножение long и float дает float. Данный пример кода не будет скомпилирован.
2	**float** f = 10000; **long** l = (**long**) (f * f);	Тут все выражение приводится к типу long. Этот пример кода будет скомпилирован.

Преобразование к типу String

- В Java к типу String можно преобразовать любые типы данных. Практически все типы можно приводить к типу String неявно. Лучше всего это заметно, когда мы складываем две переменных: String и «не String». При этом «не String» переменная преобразовывается к типу String.

	Команда	Что происходит на самом деле
1	**int** x = 5; **String** text = "X=" + x;	**int** x = 5; **String** s = "X=" + Integer.toString(x);
2	**Cat** cat = **new** Cat("Vaska"); **String** text = "My cat is " + cat;	**Cat** cat = **new** Cat("Vaska"); **String** text = "My cat is" + cat.toString();
3	**Object** o = **null**; **String** text = "Object is " + o;	**Object** o = **null**; **String** text = "Object is " + "null";
4	**String** text = 5 + '\u0000' + "Log";	**int** i2 = 5 + (**int**) '\u0000'; **String** text = Integer.toString(i2) + **"Log"**;
5	**String** text = "Object is " + (**float**) 2 / 3;	**float** f2 = ((**float**) 2) / 3; **String** text="Object is " + Float.toString(f2);

- **Вывод:** Если складываете **String** и «любой другой тип», то второй объект преобразуется к типу String.
  Обратите еще внимание на четвертую строку таблицы. Все операции выполняются слева направо, поэтому сложение 5 + ‘\u0000’ происходит как сложение целых чисел.
  Т.е. если код типа: String s = 1+2+3+4+5+"m", то получится s = «15m»,потому, что сначала произойдет сложение чисел, а затем – преобразование их к строке.

Преобразования ссылочных типов

- Переменной типа Object можно присвоить ссылку любого типа (**расширение типа**). Но чтобы выполнить обратное присваивание (**сужение типа**) приходится явно указывать операцию приведения:

	Код	Описание
1	**String** s = "mama"; Object o = s; // o хранит String	Типичное расширение ссылочного типа

2	**Object** o = "mama";// o хранит String
	String s2 = (String) o;
		Типичное сужение ссылочного типа.

3	**Integer** i = 123; //o хранит Integer
	Object o = i;
		Расширение типа.

4	**Object** o = 123; //o хранит Integer
	String s2 = (String) o;
		Ошибка во время исполнения!
		Невозможно привести ссылку на число к ссылке на строку.

5	**Object** o = 123; //o хранит Integer
	Float s2 = (Float) o;
		Ошибка во время исполнения!
		Невозможно привести ссылку на целое число к ссылке на дробное число.

6	**Object** o = 123f; // o хранит Float
	Float s2 = (Float) o;
		Приведение к своему типу. Операция сужения ссылочного типа.

- При расширении или сужении ссылочных типов никакого изменения объекта не происходит. Сужающей (или расширяющей) является именно операция присваивания, при которой, либо выполняется «проверка соответствия типов переменной и ее нового значения» либо нет.

- Чтобы не было ошибок, как в примерах, есть способ узнать, какой именно тип сохранили в переменную типа Object:

```
int i = 5;
float f = 444.23f;
String s = "17";
Object o = f;                  //o хранит объект типа Float

if (o instanceof  Integer)
{
   Integer i2 = (Integer) o;
}
else if (o instanceof  Float)
{
   Float f2 = (Float) o;          //отработает именно этот if
}
else if (o instanceof  String)
{
   String s2 = (String) o;
}
```

Вещественные типы

- Пример: **float** f = 3 / 5; в результате данного вычисления значение f будет равно нулю!

- На самом деле тут нет никакой ошибки. В делении участвуют два целых числа, поэтому происходит деление нацело, остаток от деления просто отбрасывается. Чтобы такого не было, нужно, чтобы хотя бы одно из двух чисел, участвующих в делении, было дробным. Если одно из чисел дробное, то сначала второе число будет преобразовано к дробному типу, а затем будет выполнено деление.

- Вот как можно решить данную проблему:

Запись дробного числа:

```
float f = 3.0f / 5.0f;
```

```
float f = 3.0f / 5;
```

```
float f = 3 / 5.0f;
```

- Если в делении участвуют переменные

Преобразование целой переменной в вещественную

```
int a = 3, b = 5;
float f = (a * 1.0f) / b;
```

```
int a = 3, b = 5;
float f = a / (b * 1.0f);
```

```
int a = 3, b = 5;
float f = (a*1.0) / (b*1.0);
```

```
int a = 3, b = 5;
float f = (float) a / b;
```

Литералы

- Литералы — это все данные, которые записаны прямо в Java-коде.

Примеры:

	Литерал	Тип	Описание
1	123676	**int**	Целое число
2	222233334444333322222**L**	**long**	Длинное целое число
3	12.323232323**f**	**float**	Дробное число
4	12.33333333333333333**d**	**double**	Длинное дробное число
5	«Mama» «» «Mama\nMila\nRamu\u123 »	**String**	Строка
6	\u3232’ ‘T’ ‘5’	**char**	Символ
7	**true, false**	**boolean**	Логический тип
8	**null**	**Object**	Ссылка на объект

- T.e. код – это методы, классы, переменные,..., а литералы – это конкретные значения переменных, записанные прямо в коде.

Задачи

1. Расставьте правильно **операторы приведения типа**, чтобы получился ответ: d > 0

```
public class Solution {
   public static void main(String[] args) {
      int a = 0;
      int b = (byte) a + 46;
      byte c = (byte) (a * b);
      double f = (char) 1234.15;
      long d = (short) (a + f / c + b);
      System.out.println(d);
   }
}
```

2. Расставьте правильно **операторы приведения типа**, чтобы получился ответ: d=3.765

```
public class Solution {
   public static void main(String[] args) {
      int a = 15;
      int b = 4;
      float c = a / b;
      double d = a * 1e-3 + c;
      System.out.println(d);
   }
}
```

3. Уберите ненужные **операторы приведения типа**, чтобы получился ответ: result: 1000.0

```
public class Solution {
   public static void main(String[] args) {
      double d = (short) 2.50256e2d;
      char c = (short) 'd';
      short s = (short) 2.22;
      int i = (short) 150000;
      float f = (short) 0.50f;
      double result = f + (i / c) - (d * s) - 500e-3;
      System.out.println("result: " + result);
   }
}
```

4. Уберите ненужные **операторы приведения типа**, чтобы получился ответ: 1234567

```
public class Solution {
   public static void main(String[] args) {
      long l = (byte) 1234_564_890L;
      int x = (byte) 0b1000_1100_1010;
      double m = (byte) 110_987_654_6299.123_34;
      float f = (byte) l++ + 10 + ++x - (float) m;
      l = (long) f / 1000;
      System.out.println(l);
   }
}
```