OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

05 – Java Data Types (Часть 2)

Дмитрий Коган

Как меня слышно и видно?

Если нет – напишите, если слышите – смайлик в чат.

Цели :

- Объявим, инициализируем и присвоим значения переменным
- Разберёмся с привидением типов данных
- Потренируемся в «заклинаниях»

Начинаем?

Темы экзамена

- ❑ Java Basics
- ❑ Working with Java Data Types
- ❑ Using Operators and Decision Constructs
- ❑ Creating and Using Arrays
- ❑ Using Loop Constructs
- ❑ Working with Methods and Encapsulation
- ❑ Working with Inheritance
- ❑ Handling Exceptions
- ❑ Working with Selected classes from the Java API

Подтемы экзамена

Working with Java Data Types

- **Declare and initialize variables (including casting of primitive data types)**
- **Differentiate between object reference variables and primitive variables**
- **Know how to read or write to object fields**
- **Explain an object's lifecycle (creation, "dereference by reassignment" and garbage collection)**
- **Develop code that uses wrapper classes such as Boolean, Double and Integer**

Объявление переменных

Множественные переменные

```
String s1, s2;
String s3 = "yes", s4 = "no";
```

```
int i1, i2, i3 = 0;
```

```
int num, String value; // DOES NOT COMPILE
```

Множественные переменные

```
boolean b1, b2;
String s1 = "1", s2;
double d1, double d2;
int i1; int i2;
int i3; i4;

int i1;
int i2;
int i3;
i4;// DOES NOT COMPILE
```

Упражнение

```
public static void main (String [] args){
    int a, b, c = 0;                                    // line 1
    int a, b, c;                                        // line 2
    int g, int h, int i = 0;                            // line 3
    int d, e, FIELD;                                    // line 4
    int k = l = m = 0;                                  // line 5
}
```

Which two declarations compile?

A. On line 1
B. On line 2
C. On line 3
D. On line 4
A. On line 5

Ответ: AD

Вопросы?

Инициализация переменных

Порядок важен

```
int j, k = 1, l, m = k + 3;  // legal: k is initialized before m uses it
int j, k = m + 3, l, m = 1;  // illegal: m is not declared is initialized
                             // before k uses it
int x, y = x + 1, z;         // illegal: x is not initialized before y uses it
```

Локальные переменные

```
4: public int notValid() {
5:  int y = 10;
6:  int x;
7:  int reply = x + y; // DOES NOT COMPILE
8:  return reply;
9: }

Test.java:5: variable x might not have been initialized
        int reply = x + y;
                    ^
```

Локальные переменные

```
public int valid() {
  int y = 10;
  int x; // x is declared here
  x = 3; // and initialized here
  int reply = x + y;
  return reply;
}
```

Локальные переменные

```
public void findAnswer(boolean check) {
  int answer;
  int onlyOneBranch;
  if (check) {
    onlyOneBranch = 1;
    answer = 1;
  } else {
    answer = 2;
  }
  System.out.println(answer);
  System.out.println(onlyOneBranch); // DOES NOT COMPILE
}
```

Упражнение

```
public class Assignment {
  public static void main(String[] args) {
    int a, b, c;
    b = 10;
    a = b = c = 20;
    System.out.println(a);
  }
}
```

Select the one correct answer.

(a) The program will fail to compile, since the compiler will report that the variable `c` in the multiple assignment statement `a = b = c = 20;` has not been initialized.

(b) The program will fail to compile, because the multiple assignment statement `a = b = c = 20;` is illegal.

(c) The code will compile, and print `10` at runtime.

(d) The code will compile, and print `20` at runtime.

Ответ: D

Локальные ссылочные переменные

```
import java.util.Date;
public class TimeTravel {
  public static void main(String [] args) {
    Date date;
    if (date == null)
      System.out.println("date is null");
  }}
```

```
%javac TimeTravel.java
TimeTravel.java:5: Variable date may not have been initialized.
        if (date == null)
1 error
```

Упражнение

```
public class MyClass {
  public static void main(String[] args) {
    String a, b, c;
    c = new String(“mouse”);
    a = new String(“cat”);
    b = a;
    a = new String(“dog”);
    c = b;
    System.out.println(c);
  }
}
```

Select the one correct answer.

(a) The program will fail to compile.

(b) The program will print `mouse` at runtime.

(c) The program will print `cat` at runtime.

(d) The program will print `dog` at runtime.

(e) The program will randomly print either `cat` or `dog` at runtime.

Ответ: C

Нелокальные переменные

Variable Type	Default Value
Object reference	`null` (not referencing any object)
`byte`, `short`, `int`, `long`	`0`
`float`, `double`	`0.0`
`boolean`	`false`
`char`	`'\u0000'`

Проблема

```
public class Book {
  private String title;              // instance reference variable
  public String getTitle() {
    return title;
  }
  public static void main(String [] args) {
    Book b = new Book();
    String s = b.getTitle();      // Compiles and runs
    String t = s.toLowerCase();   // Runtime Exception!
  }
}
```

Решение

```
public class Book {
  private String title;              // instance reference variable
  public String getTitle() {
    return title;
  }
  public static void main(String [] args) {
    Book b = new Book();
    String s = b.getTitle();     // Compiles and runs
    if (s != null) {
      String t = s.toLowerCase();
    }
  }
}
```

Упражнение

```
public class Init {

  String title;
  boolean published;

  static int total;
  static double maxPrice;

  public static void main(String[] args) {
    Init initMe = new Init();
    double price;
    if (true)
      price = 100.00;
    System.out.println("|" + initMe.title + "|" + initMe.published + "|"
+
                              Init.total + "|" + Init.maxPrice + "|" + price +
"|");
    }
}
```

Select the one correct answer.

(a) The program will fail to compile.

(b) The program will compile, and print `|null|false|0|0.0|0.0|` at runtime.

(c) The program will compile, and print `|null|true|0|0.0|100.0|` at runtime.

(d) The program will compile, and print `|  |false|0|0.0|0.0|` at runtime.

(e) The program will compile, and print `|null|false|0|0.0|100.0|` at runtime.

Ответ: E

Вопросы?

Присвоение значений переменным

Пока всё просто

```
int x = 7;      // literal assignment
int y = x + 2; // assignment with an expression
                // (including a literal)
int z = x * y; // assignment with an expression
```

И вот всё пропало

```
byte b = 27;
```

```
byte b = (byte) 27; // Explicitly cast the int literal to a byte
```

Приведение типа

- ✓ Кастинг – это процесс, которым мы заставляем ту или иную переменную вести себя по правилам другого типа. Кастинг применим и к примитивным и к ссылочным типом, и может быть явным или неявным.

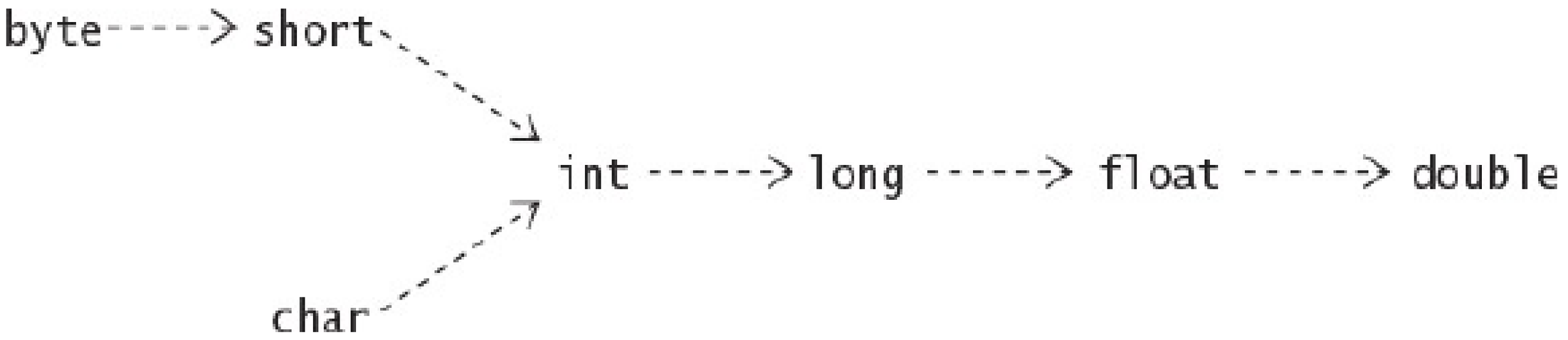
- ✓ Автоматическое приведение типа, выполняемое Джава-компилятором по собственному почину, является неявным или имплицитным (implicit casting). Имплицитный кастинг позволяет преобразовать "узкий" тип к более "широкому".

- ✓ А вот кастинг, который делает сам программист, является явным или эксплицитным (explicit casting). Такой кастинг обязателен, если надо "сузить" тот или иной тип.

Приведение типа

- ✓ Преобразование узкого типа к широкому называется расширением (widening). Расширение безопасно, и вот почему Джава-компилятор не покажет красную карточку, даже если программист не воспользуется оператором кастинга (T).
- ✓ И напротив, сужение (narrowing) небезопасно по самой своей природе, вот почему программист в таких случаях обязан ставить кастинг-оператор.

Расширение примитивов



Кастинг примитивов

```
int a = 100;
long b = a;      // Implicit cast, an int value always fits in a long
```

```
float a = 100.001f;
int b = (int)a; // Explicit cast, the float could lose info
```

```
double d = 100L; // Implicit cast
```

Сужение примитивов

```
class Casting {
  public static void main(String [] args) {
    int x = 3957.229; // illegal
  }
}
```

```
%javac Casting.java
Casting.java:3: Incompatible type for declaration. Explicit cast
needed to convert double to int.
      int x = 3957.229; // illegal
1 error
```

Сужение примитивов

```
class Casting {
  public static void main(String [] args) {
    int x = (int)3957.229; // legal cast
    System.out.println("int x = " + x);
  }
}
```

```
int x = 3957
```

Сужение примитивов

```
class Casting {
  public static void main(String [] args) {
    long l = 56L;
    byte b = (byte)l;
    System.out.println("The byte is " + b);
  }
}
```

Сужение примитивов

```
class Casting {
  public static void main(String [] args) {
    long l = 130L;
    byte b = (byte)l;
    System.out.println("The byte is " + b);
  }
}
```

```
%java Casting
The byte is -126
```

Слишком большие литералы

```
byte a = 128; // byte can only hold up to 127
```

```
TestBytes.java:5: possible loss of precision
found   : int
required: byte
byte a = 128;
```

```
byte a = (byte) 128;
```

Слишком большие литералы

```
int iMax = 12345678901;          // численные литералы не должны
                                 // вываливаться за свои границы
long lMax = 31234567890;         // в литерал не влезает int
                                 // хотя в long он бы влез
long lMaxTrue = 31234567890L;
```

Примеры расширения

```
// Widening Primitive Conversions
int    smallOne = 1234;                 // No widening necessary.
long   bigOne   = 2000;                 // Widening: int to long.
double largeOne = bigOne;               // Widening: long to double.
double hugeOne  = (double) bigOne;      // Cast redundant but allowed.
```

Потеря точности

Преобразование из `int` или `long` во `float` или из `long` в `double` может привести к потери точности.

```
long bigInteger = 98765432112345678L;
float fpNum = bigInteger;  // Widening but loss of precision: 9.8765436E16
```

Упражнение

Given the following declaration:

```
char c = 'A';
```

What is the simplest way to convert the character value in `c` to an `int`?

Select the one correct answer.

(a) `int i = c;`

(b) `int i = (int) c;`

(c) `int i = Character.getNumericValue(c);`

Ответ: А

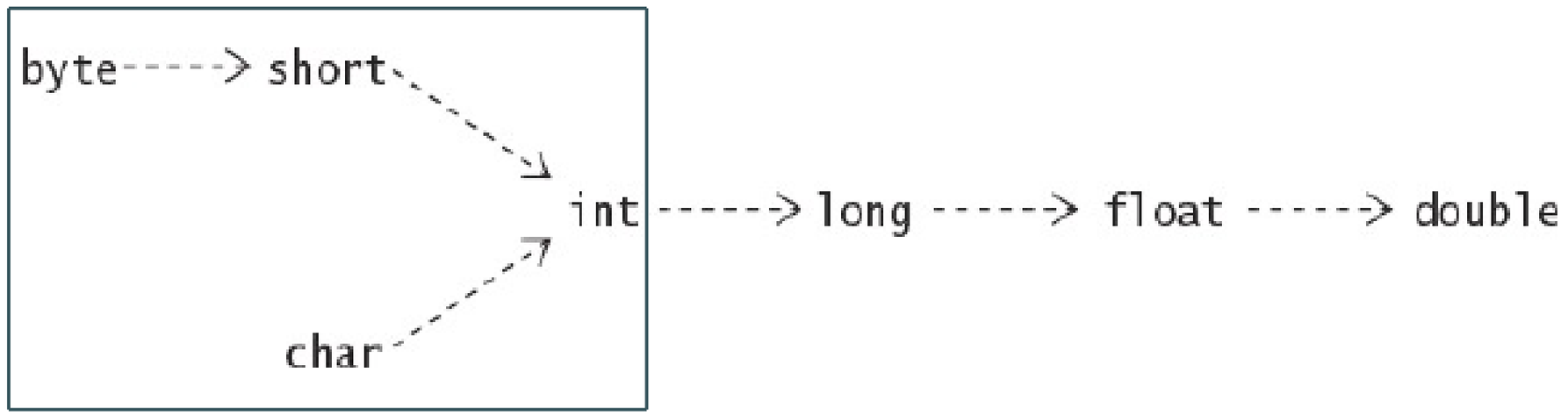
Кастинг символов

```
char c = (char)70000;   // The cast is required; 70000 is
                        // out of char range
char d = (char) -98;    // Ridiculous, but legal
```

```
char e = -29;     // Possible loss of precision; needs a cast
char f = 70000;   // Possible loss of precision; needs a cast
```

byte/short ←→ char

- ✓ Преобразования между byte- или short-переменными с одной стороны и char с другой – это всегда **сужение**.
- ✓ Сначала тип источника переводится в int, а затем осуществляется перевод из int в тип получателя.



Кастинг символов

Здесь главное помнить, что `char` всегда положителен, в то время как `byte`, `short` и проч. могут быть отрицательными, и вот почему **`char`** = **`short`** и т.д. *не скомпилируется*.

Кроме того, `char` может вместить до 65535, а вот `short` вмещает максимум 32767. Именно поэтому **`short`** = **`char`** *тоже не скомпилируется*. Здесь на помощь приходится звать эксплицитный кастинг.

Константы

Никакую ПЕРЕМЕННУЮ кроме `char` нельзя присвоить *чару* без явного кастинга, а вот КОНСТАНТЫ можно присваивать *чару*, но только если значение в нем поместится.

```
final int s = 10;
char c = s;    //
```

OK, потому что **s** – это константа (она ведь `final`) и ее значение умещается в `char`.

Имплицитное сужение

Additionally, implicit narrowing primitive conversions on assignment can occur in cases where *all* of the following conditions are fulfilled:

- The source is a *constant expression* of either `byte`, `short`, `char`, or `int` type.
- The target type is either `byte`, `short`, or `char` type.
- The value of the source is determined to be in the range of the target type at compile time.

Константы

```
int result = 100;                    // Not a constant variable. Not declared
final.
final char finalGrade = 'A';         // Constant variable.
System.out.printf("%d%n%s%n%d%n%.2f%n%b%n%d%n%d%n",
    2106,                            // Constant expression.
    "Trust " + "me!",                // Constant expression.
    2 + 3 * 4,                       // Constant expression.
    Math.PI * Math.PI * 10.0,        // Constant expression.
    finalGrade == 'A',               // Constant expression.
    Math.min(2015, 2016),            // Not constant expression. Method call.
    ++result                         // Not constant expression. Increment
operator.
);
```

Имплицитное сужение

```
// Conditions fulfilled for implicit narrowing primitive conversions.
short s1 = 10;          // int value in range.
short s2 = 'a';         // char value in range.
char c1 = 32;           // int value in range.
char c2 = (byte)35;  // byte value in range. (int value in range, without
cast.)
byte b1 = 40;           // int value in range.
byte b2 = (short)40; // short value in range. (int value in range, without
cast.)
final int i1 = 20;   // Constant variable
byte b3 = i1;           // final value of i1 in range.
```

Эксплицитное сужение

```
// Conditions not fulfilled for implicit narrowing primitive conversions.
// A cast is required.
int i2 = -20;              // i2 is not a constant variable. i2 is not final.
final int i3 = i2;         // i3 is not a constant variable, since i2 is not.
final int i4 = 200;        // i4 is a constant variable.
final int i5;              // i5 is not a constant variable.
short s3 = (short) i2;     // Not constant expression.
char  c3 = (char)  i3;     // Final value of i3 not determinable at compile
time.
char  c4 = (char)  i2;     // Not constant expression.
byte  b4 = (byte)  128;    // int value not in range.
byte  b5 = (byte)  i4;     // Value of constant variable i4 is not in range.
i5 = 100;                  // Initialized at runtime.
short s4 = (short) i5;     // Final value of i5 not determinable at compile
time.
```

Числа с плавающей точкой

```
float f = 32.3;   // не скомпилируется
float f = (float) 32.3;
float g = 32.3f;
float h = 32.3F;
```

Потери от сужения

```
// The value is truncated to fit the size of the target type.
float huge   = (float) 1.7976931348623157d;  // double to float.
long  giant  = (long)  4415961481999.03D;    // (1) double to long.
int   big    = (int)   giant;                // (2) long to int.
short small  = (short) big;                  // (3) int to short.
byte  tiny   = (byte)  small;                // (4) short to byte.
char  symbol = (char)  112.5F;               // (5) float to char.
```

Binary	Decimal	
0000000000000000000001000000010000**10**1011110100001100001100001111	4415961481999	(1)
00**10**1011110100001100001100001111	735101711	(2)
1100001100001111	-15601	(3)
00001111	15	(4)
0000000001110000	'p'	(5)

Имплицитное сужение

Имплицитное сужение распространяется лишь на `byte`, `char`, `short` и `int` → такая операция невозможна с типами `long`, `float` или `double`.

Вот почему следующие строчки не скомпилируются:

```
int i = 129L;      // (потому что long не допускает неявного сужения)
char ch = 30L;     // (тоже ошибка, пусть даже 30 помещается в типе char)
```

Бинарные операции

```
byte a = 3;      // No problem, 3 fits in a byte
byte b = 8;      // No problem, 8 fits in a byte
byte c = a + b; // Should be no problem, sum of the two bytes
                 // fits in a byte
```

```
TestBytes.java:5: possible loss of precision
found   : int
required: byte
    byte c = a + b;
                 ^
```

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

Бинарные операции с byte, short, char переводят их в int !

Бинарные операции

If `T` is wider than `int`, both operands are converted to `T`; otherwise, both operands are converted to `int`.

This means that the resulting type of the operands is at least `int`.

```
if(5==5.0f) System.out.println("true"); // всегда true; int расширается до float
if(5==5.0) System.out.println("true"); // всегда true; int расширается до double
```

Если один из операндов шире int, оба операнда переводится в его тип.

Составные операторы

```
byte b = 3;
b += 7;                    // No problem - adds 7 to b (result is 10)
```

Составные операторы

```
byte b = 3;
b += 7;                    // No problem - adds 7 to b (result is 10)
```

and it is equivalent to this:

```
byte b = 3;
b = (byte) (b + 7);   // Won't compile without the
                      // cast, since b + 7 results in an int
```

Действительно для операторов +=, -=, *=, /= .

Упражнение

```
 3. public static void main(String[] args)  {
 4.         float fVar = 123.123f;
 5.         double dVar  = 123;
 6.         short sVar = 123;
 7.         int iVar = 123;
 8.         long lVar = 123;
 9.         iVar = fVar;
10.         fVar = iVar;
11.         dVar = fVar;
12.         lVar = sVar;
13.         fVar = dVar;
14.         dVar = iVar;
15.         iVar = dVar;
16. }
```

How many LOCs fail to compile?

A. one
B. two
C. three
D. four

Ответ: C (9, 13, 15)

Исключения (кастинг примитивов)

- ✓ Ни расширение, ни сужение примитивов никогда не выбрасывают исключений (runtime exceptions) на этапе исполнения.

Приведение ссылочных переменных

```
Object obj = “Upcast me”;  // Widening: Object <-- String
```

```
String str = (String) obj; // Narrowing requires cast: String <-- Object
```

Исключения (кастинг ссылочных переменных)

- ✓ Сужение ссылочных переменных может выбрасывать ClassCastException на этапе исполнения.

null

И наконец, `null` можно присваивать и эксплицитно кастировать до любого ссылочного типа:

```
"Hello".equals((String)null);        // прекрасно компилируется
```

Вопросы?

Домашнее задание

Тест

Домашнее задание

Дифференциация между ссылочными и примитивными переменными

Цель: Закрепление материала вебинара с помощью прохождения теста, аналогичного экзаменационному.

1. Пройдите, пожалуйста, тест: https://forms.gle/tHzxuf3QiUroSCUz5
2. Сообщите о прохождении в Чате с преподавателем.

Критерии оценки: Тест считается пройденным, если результат - выше 65%.

Рекомендуем сдать до: 11.11.2020

Статус: не сдано

Чат с преподавателем

Пожалуйста, пройдите опрос

https://otus.ru/polls/17811/

Спасибо за внимание!

Кастуйте правильно! ☺