OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

06 – Java Basics (Часть 3)

Дмитрий Коган
Петрелевич Сергей

Как меня слышно и видно?

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

Цели :

- Обойдём области действия всех типов переменных вдоль и поперёк
- Сделаем большой шаг вперёд к сдаче экзамена

OTUS

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

Java Basics

- **Define the scope of variables**
- **Define the structure of a Java class**
- **Create executable Java applications with a main method; run a Java program from the command line; produce console output**
- **Import other Java packages to make them accessible in your code**
- **Compare and contrast the features and components of Java such as: platform independence, object orientation, encapsulation, etc.**

Области действия переменных

Области действия

- Область действия означает ту часть кода, внутри которой переменная видима или, другими словами, доступна.
- Переменные могут действовать на разных уровнях и, стало быть, обладать разными областями действия:
  - переменные класса (class-level variables),
  - переменные экземпляра (instance variables),
  - локальные переменные (local variables), в т.ч. переменные циклов (loop vars), и
  - аргументы методов (method arguments).

Локальные переменные

- Локальные переменные чаще всего встречаются в теле метода или конструктора, а также в подблоках (sub-blocks).
- Область действия той или иной переменной ограничена ближайшими парными фигурными скобками (matching curly braces), {}. К примеру, область действия локальной переменной будет у́же области действия метода, если эта переменная объявлена в подблоке, расположенном внутри данного метода. К таким подблокам относятся: оператор `if`, `switch`-конструкции, циклы, TCF-конструкции, а также любые группы операторов, взятые в парные фигурные скобки.
- Локальные переменные не видны (*читай*: не доступны) вне того метода или подблока, где они были определены.

Сколько тут локальных?

```
public void eat(int piecesOfCheese) {
    int bitesOfCheese = 1;
}
```

Ответ для экзамена

OCA Oracle Certified Associate Java SE 8 Programmer I
Stady Guide Exam 1Z0-808
Jeanne Boyarsky and Scott Selikoff

Стр. 31
There are **two local variables** in this method. bitesOfCheese is declared inside the method. piecesOfCheese is called a method parameter.
It is also local to the method.
Both of these variables are said to have a scope local to the method.
This means they cannot be used outside the method.

Что видим в коде

```
public void eat(int);
   descriptor: (I)V
   flags: (0x0001) ACC_PUBLIC
   Code:
     stack=1, locals=3, args_size=2
        0: iconst_1
        1: istore_2
        2: return
   LineNumberTable:
     line 8: 0
     line 9: 2
   LocalVariableTable:
     Start  Length  Slot  Name   Signature
         0       3     0  this   Lru/petrelevich/bytes/LocalVars;
         0       3     1 piecesOfCheese   I
         2       1     2 bitesOfCheese   I
```

Куда копать

Escape Analysis

Локальные переменные

```
3: public void eatIfHungry(boolean hungry) {
4:  if (hungry) {
5:   int bitesOfCheese = 1;
6:  }  // bitesOfCheese goes out of scope here
7:   System.out.println(bitesOfCheese);// DOES NOT COMPILE
8: }
```

```
bitesOfCheese cannot be resolved to a variable
```

Локальные переменные

```
16: public void eatIfHungry(boolean hungry) {
17:   if (hungry) {
18:     int bitesOfCheese = 1;
19:     {
20:       boolean teenyBit = true;
21:       System.out.println(bitesOfCheese);
22:     }
23:   }
24:   System.out.println(teenyBit);  // DOES NOT COMPILE
25: }
```

Локальные переменные

```
11: public void eatMore(boolean hungry, int amountOfFood) {
12:  int roomInBelly = 5;
13:  if (hungry) {
14:    boolean timeToEat = true;
15:    while (amountOfFood > 0) {
16:      int amountEaten = 2;
17:      roomInBelly = roomInBelly - amountEaten;
18:      amountOfFood = amountOfFood - amountEaten;
19:    }
20:  }
21:  System.out.println(amountOfFood);
22: }
```

Line	First line in block	Last line in block
`while`	15	19
`if`	13	20
Method	11	22

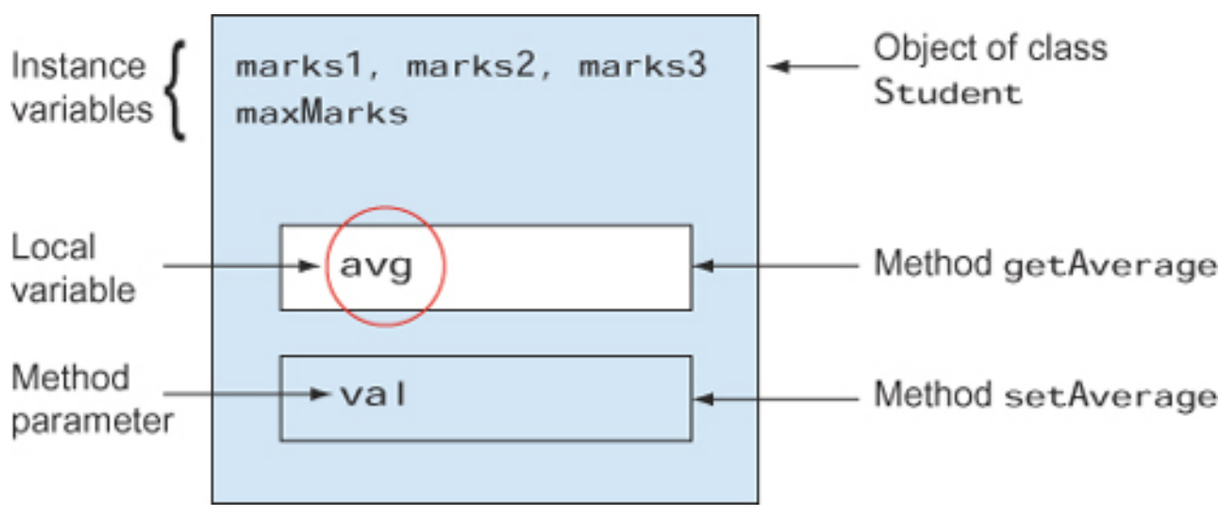
Локальные переменные

```
class Student {
    private double marks1, marks2, marks3;
    private double maxMarks = 100;
    public double getAverage() {
        double avg = 0;
        avg = ((marks1 + marks2 + marks3) / (maxMarks*3)) * 100;
        return avg;
    }
    public void setAverage(double val) {
        avg = val;
    }
}
```

Instance variables

Local variable avg

This code won't compile because avg is inaccessible outside the method getAverage.



Вызов методов

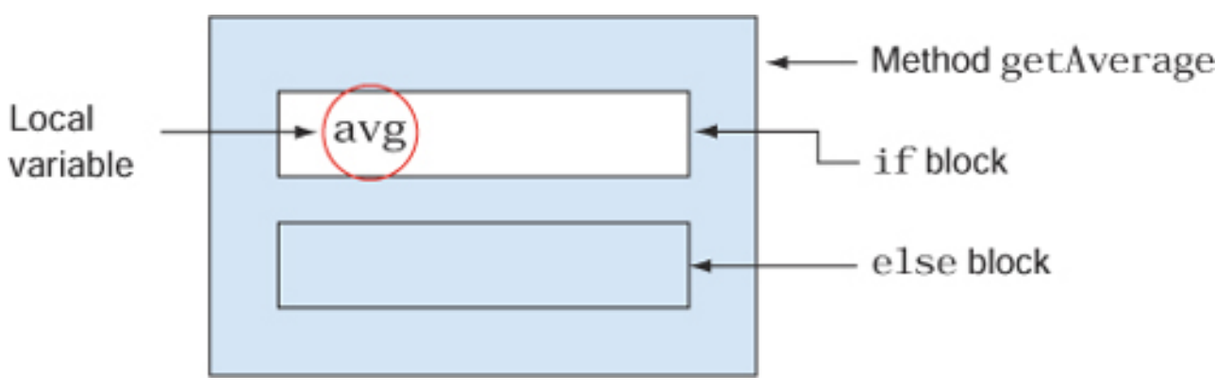
```
class ScopeErrors {
  public static void main(String [] args) {
    ScopeErrors s = new ScopeErrors();
    s.go();
  }
  void go() {
    int y = 5;
    go2();
    y++;          // once go2() completes, y is back in scope
  }
  void go2() {
    y++;          // won't compile, y is local to go()
  }
}
```

Локальные переменные

```
public double getAverage() {
    if (maxMarks > 0) {
        double avg = 0;
        avg = (marks1 + marks2 + marks3)/(maxMarks*3) * 100;
        return avg;
    }
    else {
        avg = 0;
        return avg;
    }
}
```

Variable avg is local to if block

Variable avg can't be accessed because it's local to the if block. Variables local to the if block can't be accessed in the else block.



Порядок важен

```
public void forwardReference() {
    int a = b;
    int b = 20;
}
```

Won't compile

```
public void noForwardReference() {
    int b = 20;
    int a = b;
}
```

No forward reference; code compiles

Внутри метода объявленная переменная действует на всё, что после и не видна до объявления!

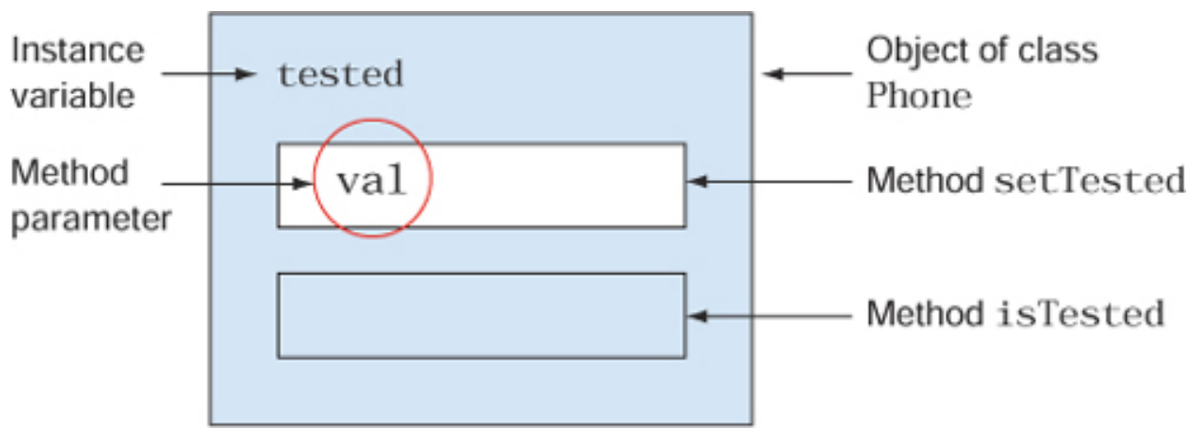
Параметры метода

```
class Phone {
    private boolean tested;
    public void setTested(boolean val) {
        tested = val;
    }
    public boolean isTested() {
        val = false;
        return tested;
    }
}
```

Method parameter val is accessible only in method setTested

Variable val can't be accessed in method isTested

This line of code won't compile.



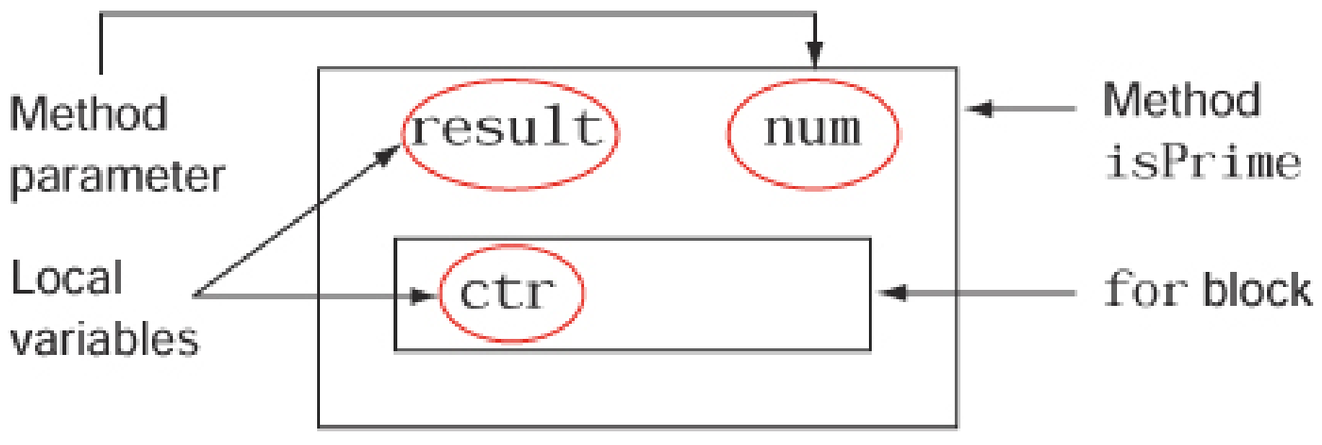
Параметры метода

```
boolean isPrime(int num) {
    if (num <= 1) return false;
    boolean result = true;
    for (int ctr = num-1; ctr > 1; ctr--) {
        if (num%ctr == 0) result = false;
    }
    return result;
}
```

Method parameter num

Local variable result

Local variable ctr



Переменные объекта и класса

- Переменные экземпляра (instance variables) определены и доступны внутри объекта, означая тем самым, что для работы с такими переменными нам нужен объект. Любой нестатический метод может обращаться к любым переменным экземпляра в рамках своего класса.
- Переменные класса (class-level variables), также именуемые статическими переменными (`static` variables) являются общими для всех экземпляров этого класса; мало того, такие переменные доступны даже при полном отсутствии объектов данного типа.

Переменные объекта и класса

```
1:  public class Mouse {
2:    static int MAX_LENGTH = 5;
3:    int length;
4:    public void grow(int inches) {
5:      if (length < MAX_LENGTH) {
6:        int newSize = length + inches;
7:        length = newSize;
8:      }
9:    }
10: }
```

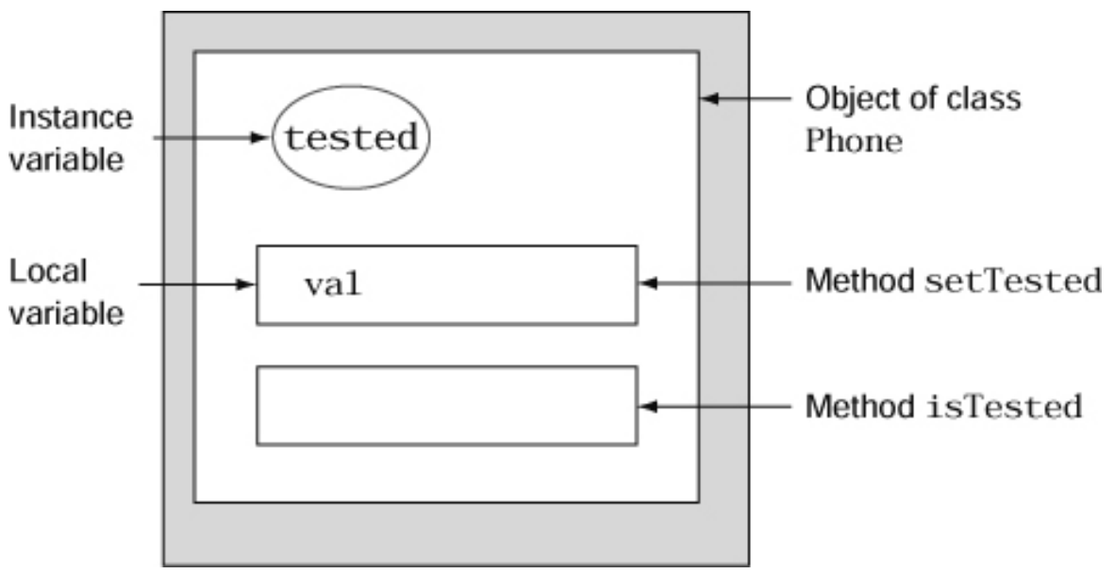
Переменные объекта

```
class Phone {
    private boolean tested;
    public void setTested(boolean val) {
        tested = val;
    }
    public boolean isTested() {
        return tested;
    }
}
```

Instance variable tested

Variable tested is accessible in method setTested

Variable tested is also accessible in method isTested



Переменные класса

```
package com.mobile;
class Phone {
    static boolean softKeyboard = true;
}
```

Class variable softKeyboard

```
package com.mobile;
class TestPhone {
    public static void main(String[] args) {
        Phone.softKeyboard = false;
        Phone p1 = new Phone();
        Phone p2 = new Phone();
        System.out.println(p1.softKeyboard);
        System.out.println(p2.softKeyboard);
        p1.softKeyboard = true;
        System.out.println(p1.softKeyboard);
        System.out.println(p2.softKeyboard);
        System.out.println(Phone.softKeyboard);
    }
}
```

Accesses the class variable by using the name of the class. It can be accessed even before any of the class's objects exist.

Prints false. A class variable can be read by using objects of the class.

Prints false

Prints true

A change in the value of this variable will be reflected when the variable is accessed via objects or class name.

Переменные класса

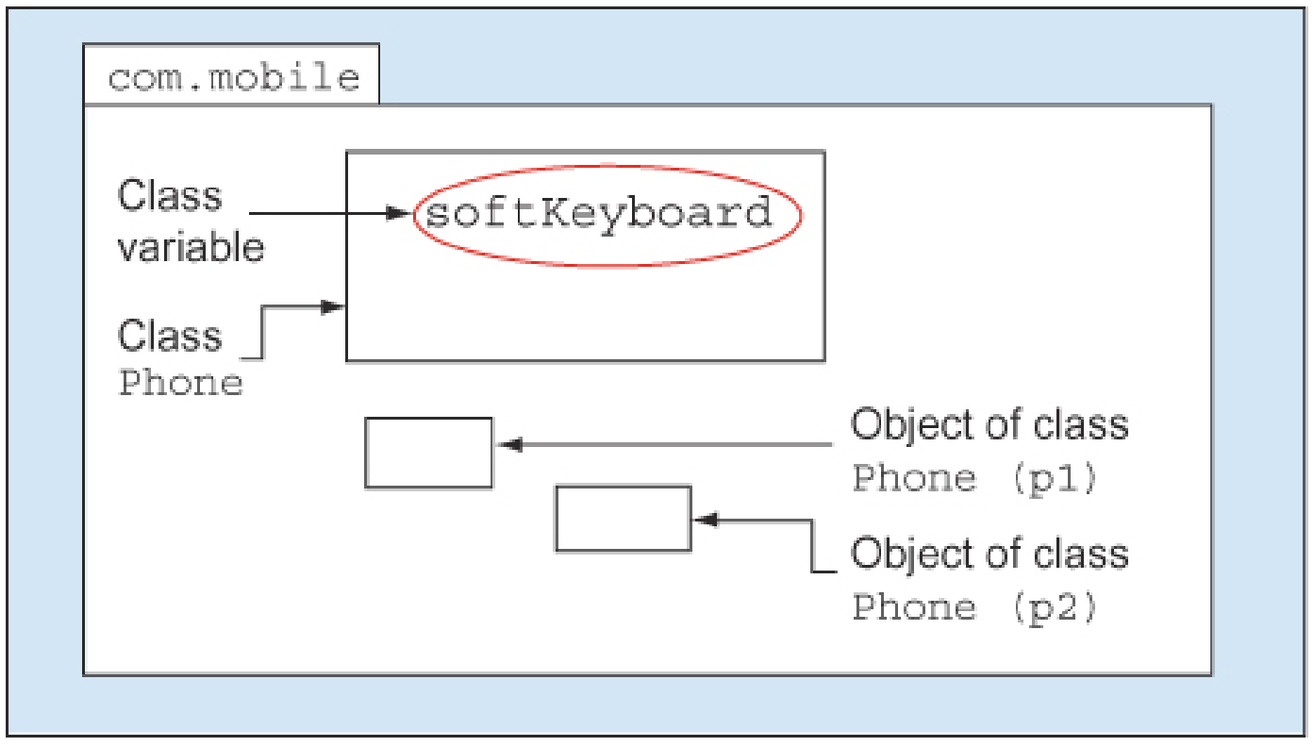
Надо помнить, что переменные класса – это, по сути, глобальные переменные.
А глобальные переменные – это очень плохая практика.

Переменные класса

- `Phone.softKeyboard`
- `p1.softKeyboard`
- `p2.softKeyboard`

```
Phone p1 = null;
System.out.println(p1.softKeyboard);
```

Won't throw an exception, even though p1 is set to null



Частая ошибка

```
class ScopeErrors {
  int x = 5;
  public static void main(String[] args) {
    x++;   // won't compile, x is an 'instance' variable
  }
}
```

Доступ к членам класса

```
class SuperName {
  int instanceVarInSuper;
  static int staticVarInSuper;

  void instanceMethodInSuper()       { /* … */ }
  static void staticMethodInSuper() { /* … */ }
  // …
}

class ClassName extends SuperName {
  int instanceVar;
  static int staticVar;

  void instanceMethod()       { /* … */ }
  static void staticMethod() { /* … */ }
  // …
}
```

Доступ к членам класса

Member declarations	Non-static code in the class ClassName can refer to the member as	Static code in the class ClassName can refer to the member as

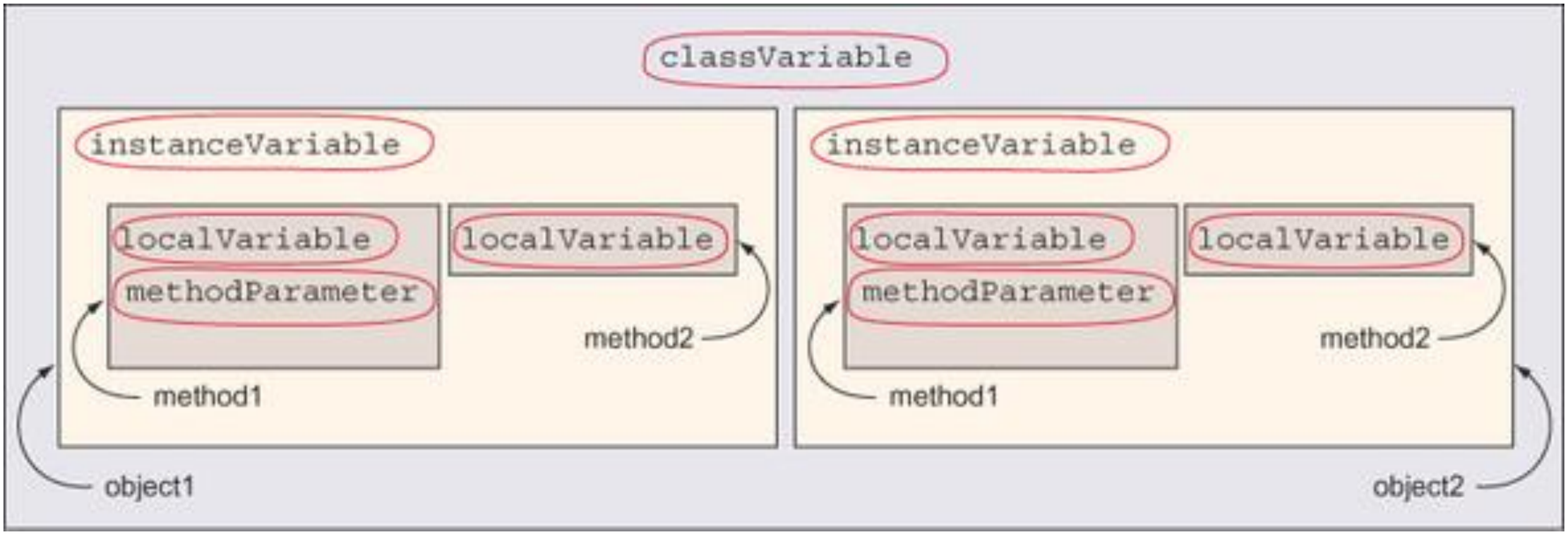
Instance variables	`instanceVar`
	`this.instanceVar`
	`instanceVarInSuper`
	`this.instanceVarInSuper`
	`super.instanceVarInSuper`
		Not possible

Instance methods	`instanceMethod()`
	`this.instanceMethod()`
	`instanceMethodInSuper()`
	`this.instanceMethodInSuper()`
	`super.instanceMethodInSuper()`
		Not possible

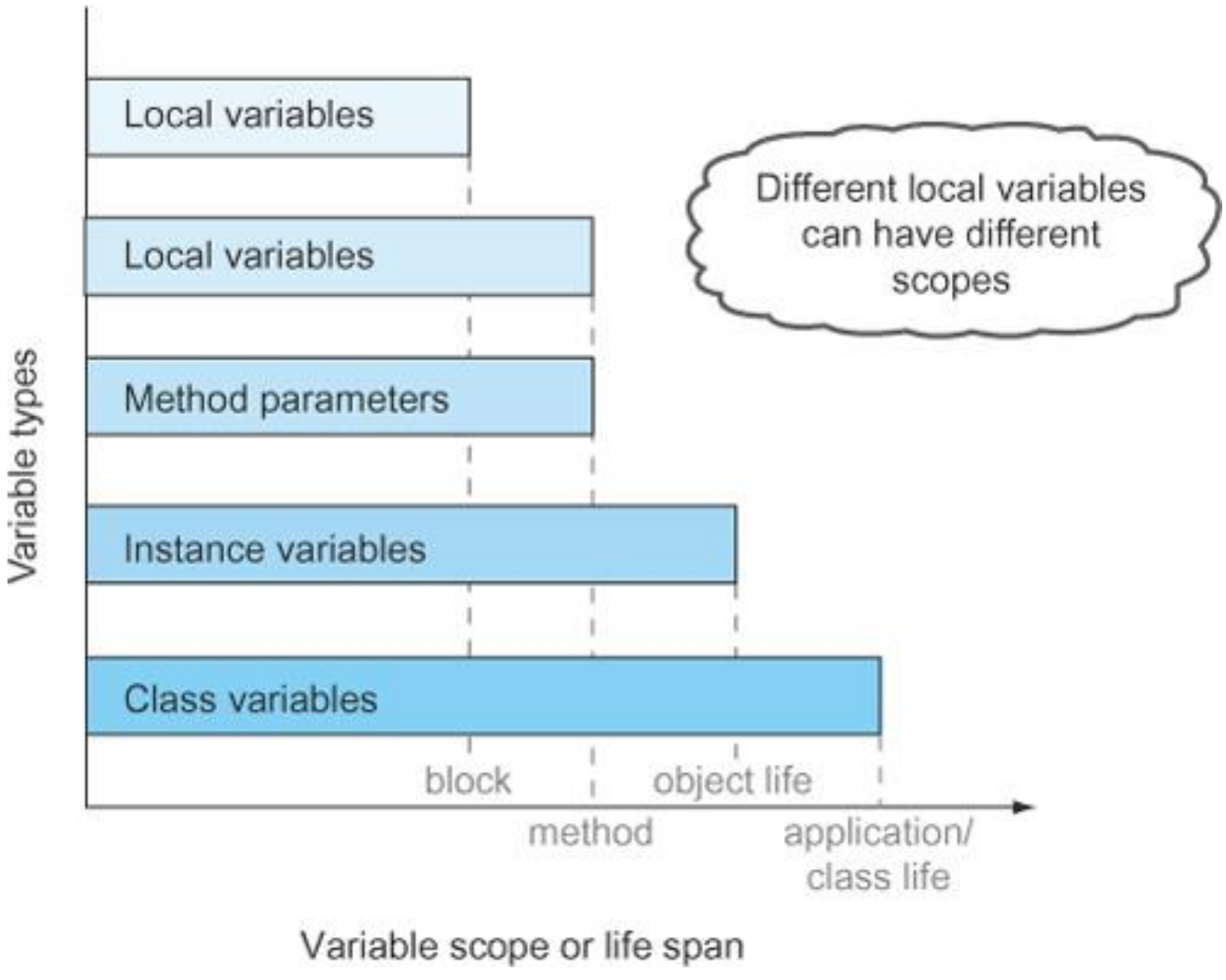
Static variables	`staticVar`
	`this.staticVar`
	`ClassName.staticVar`
	`staticVarInSuper`
	`this.staticVarInSuper`
	`super.staticVarInSuper`
	`ClassName.staticVarInSuper`
	`SuperName.staticVarInSuper`
		`staticVar`
		
		`ClassName.staticVar`
		`staticVarInSuper`
		
		`ClassName.staticVarInSuper`
		`SuperName.staticVarInSuper`

Static methods	`staticMethod()`
	`this.staticMethod()`
	`ClassName.staticMethod()`
	`staticMethodInSuper()`
	`this.staticMethodInSuper()`
	`super.staticMethodInSuper()`
	`ClassName.staticMethodInSuper()`
	`SuperName.staticMethodInSuper()`
		`staticMethod()`
		
		`ClassName.staticMethod()`
		`staticMethodInSuper()`
		
		`ClassName.staticMethodInSuper()`
		`SuperName.staticMethodInSuper()`

Пересечение областей



Время жизни


Local variables
Local variables
Method parameters
Instance variables
Class variables
Variable types
block
method
object life
application/
class life
Variable scope or life span
Different local variables
can have different
scopes

Время жизни

- Область действия любой переменной простирается с точки ее объявления. (Хотя в реальности дело обстоит сложнее, для нашего экзамена этого правила достаточно).
- Переменные класса действуют вплоть до окончания работы программы.
- Действие переменной объекта заканчивается, как только данный объект будет убран сборщиком мусора.
- Действие локальных переменных заканчивается, как только завершится выполнение данного метода / подблока.

Затенение

- Аргумент метода и локальная переменная не могут иметь одинаковое имя.
- То же самое относится к переменным класса и переменным экземпляра: для них нельзя использовать одинаковый идентификатор.
- Зато локальные переменные и переменные класса / экземпляра вполне можно объявлять через одинаковое имя. Если в методе объявлена локальная переменная, чье имя совпадает с именем переменной класса или экземпляра, локальная переменная "затеняет" (shadows) своего тезку (т.е. превращает его в "невидимку").
- Области действия переменных частенько перекрываются; единственный способ понять, доступна ли та или иная переменная – это вычленить блок, в котором она определена (путем анализа фигурных скобок).

Затенение

```
class MyPhone {
    static boolean softKeyboard = true;
    boolean softKeyboard = true;
}
```

Won't compile. Class variable and instance variable can't be defined using the same name in a class.

```
void myMethod(int weight) {
    int weight = 10;
}
```

Won't compile. Method parameter and local variable can't be defined using the same name in a method.

Затенение

```
class MyPhone {
    static boolean softKeyboard = true;
    String phoneNumber;
    void myMethod() {
        boolean softKeyboard = true;
        String phoneNumber;
    }
}
```

Class variable softKeyboard

Instance variable phoneNumber

Local variable softKeyboard can coexist with class variable softKeyboard

Local variable phoneNumber can coexist with instance variable phoneNumber

Упражнение

```
class Phone {
    String phoneNumber = "123456789";
    void setNumber () {
        String phoneNumber;
        phoneNumber = "987654321";
    }
}
class TestPhone {
    public static void main(String[] args) {
        Phone p1 = new Phone();
        p1.setNumber();
        System.out.println (p1.phoneNumber);
    }
}
```

a 123456789

b 987654321

c No output

d The class `Phone` will not compile.

Ответ

В методе setNumber локальная переменная «затенит» поле класса.
Поэтому значение константы будет присвоено локальной переменной, а не полю класса.
Поэтому значение поля phoneNumber не изменится.

Правильный ответ: a

Блоки

```
public static void main(String args[]) {                    // Block 1
//  String args = "";      // (1) Cannot redeclare parameters.
    char digit = 'z';

    for (int index = 0; index < 10; ++index) {               // Block 2

      switch(digit) {                                        // Block 3
          case 'a':
              int i;     // (2)
          default:
          //  int i;     // (3) Already declared in the same block
      } // end switch

      if (true) {                                            // Block 4
          int i;         // (4) OK
      //  int digit;     // (5) Already declared in enclosing Block 1
      //  int index;     // (6) Already declared in enclosing Block 2
      } // end if

    } // end for

    int index;           // (7) OK

} // end main
```

Вложенный цикл

```
int i = 1;
// int j;   // Если раскомментировать, в следующей строке надо писать for (j = 0; j < 5; j++)
for (int j = 0; j < 5; j++) {        // Затенить локальную j нельзя
    // int i = 2;                    // Затенить локальную i нельзя
    i = 2;                           // Так можно, конечно
    for (int k = 0; k < 5; k++) {    // for (int j = 0; j < 5; j++) не проходит
        int l = 1;       // Каждый раз создаём l по новой. Не беда
        // int j = 3;    // Не проходит
        // int i = 4;    // Не проходит
    }
    boolean k = true;    // Область действия внутреннего цикла закончилась
    // l = 1;            // Не скомпилируется, l более не известна
}
```

Ещё не всё

```
class Layout {                              // class
  static int s = 343;                       // static variable
  int x;                                    // instance variable
  { x = 7; int x2 = 5; }                    // initialization block
  Layout() { x += 8; int x3 = 6;}           // constructor

  void doStuff() {                          // method
    int y = 0;                              // local variable
    for(int z = 0; z < 4; z++) {    // 'for' code block
      y += z + x;
    }
  }
}
```

- `s` is a static variable.
- `x` is an instance variable.
- `y` is a local variable (sometimes called a "method local" variable).
- `z` is a block variable.
- `x2` is an `init` block variable, a flavor of local variable.
- `x3` is a constructor variable, a flavor of local variable.

Объединяем

```
 1 class VarScope{
 2     static int x = 4, y;
 3     static{
 4         x = 44;
 5     }
 6     int a = 1, b;
 7     {
 8         b = 11;
 9     }
10     void run(int b){
11         int a = b;
12         int c;
13         {
14             // int c = 666;                    // INVALID
15             int x = 444;
16         }
17         for (int d = 0; d < 3; d++){
18             // int a = 3;                      // INVALID
19             int e = 5;
20             e++;
21             System.out.println("e = " + e);    // раз за разом печатает 6
22         }
23     }
24     public static void main(String[] args) {
25         int a = 3;
26         new VarScope().run(a);
27     }
28 }
```

Упражнение

```
class Looper {
    public static void main(String[] args) {
        for (int kk = 0; kk < 4; kk++) {
            System.out.print("kk = "+ kk + ", ");
            kk = kk + 1;
        }
        System.out.println("kk = "+ kk + ", ");
    }
}
```

What is the result?

A. kk = 0, kk = 2, kk = 0,
B. kk = 0, kk = 2, kk = 2,
C. kk = 0, kk = 2, kk = 4,
D. Compilation fails.

Ответ

```
public static void main(String[] args) {
        for (int kk = 0; kk < 4; kk++) {
            System.out.print("kk = " + kk + ", ");
            kk = kk + 1; ←плохая практика в нескольких местах менять индекс цикла
        }
        System.out.println("kk = " + kk + ","); ← с подсветкой синтаксиса ответ
                                                    очевиден: ошибка компиляции
}
```

Упражнение

```
 3. public class Dark {
 4.   int x = 3;
 5.   public static void main(String[] args) {
 6.     new Dark().go1();
 7.   }
 8.   void go1() {
 9.     int x;
10.     go2(++x);
11.   }
12.   void go2(int y) {
13.     int x = ++y;
14.     System.out.println(x);
15.   }
16. }
```

What is the result?

A. 2

B. 3

C. 4

D. 5

E. Compilation fails

F. An exception is thrown at runtime

Ответ

```
class Dark {
   int x = 3;
   public static void main(String[] args) {
      new Dark().go1();
   }

   void go1() {
      int x;  ← локальная переменная перекрывает поле класса
      go2(++x); ← использование переменной без инициализации
   }
   void go2(int y) {
      int x = ++y;
      System.out.println(x);
   }
}
```

Ответ: ошибка компиляции

что будет, если int x; заменить на int x = 0; ?

Вопросы?

Домашнее задание

http://...

Пожалуйста, пройдите опрос

https://otus.ru/polls/xxxx/

Спасибо за внимание!

Постоянства в переменных!