Правила хорошего кода

**Quick Documentation**

Ctrl +J

**Оборачивание**: Ctrl+Alt(opt)+T – для быстрого for или if и тд вокруг кода

1. Выделить код, который вы хотите обернуть другим кодом
2. Нажать **Ctrl+Alt+T**
3. Выбрать один из вариантов оборачивания:
   1. if
   2. if-else
   3. while
   4. do while
   5. for
   6. try catch
   7. try finally
   8. try catch finally
   9. synchronized

## Стиль кода: Ctrl+Alt(opt)+L – выравнивает код

**1.**

**if** () {  
 ...  
}

- скобку фигурную – на той же строке что и if и тп

2.

не пишем на русском языке, несмотря на то, что можно

3

Код пишут всегда один раз – не нужно копировать одинаковые участки кода – под них пишутся отдельные методы. Хотя бы исходя из принципа, что если допустили ошибку в коде, справляем в одном месте, а не ищем потом где же мы это вставили.

4.

Класс следует писать так, чтобы можно было подключить его как библиотеку в свой main и пользоваться его функционалом – по сути это и есть инкапсуляция, самодостаточность, обособленность.

5.

mypackage – для пэкэджейтолько маленькими, даже если сложное название

MyClass – название классов всегда с большой буквы КэмэлКейсом

myObject - объекты

myMethod – методы

\_ - в названии обычно не применяется

однако применяется для final- переменных MY\_FINAL + разделение цифр в большом числе 0\_0

6.

Начинать не с цифр или \_

Иногда с $ начинается – в особых случаях вэба

7

в java запрещено использование слова const (в си – это не модифицируемая переменная) – вместо него final

8

запрещено goto (выход из глубокого цикла) – вместо него break;

Остальные зарезервированные слова – такие же как в си

9.

Название проекта – это очень важная часть

- Не называть на русском (в разных системах могут не совпадать кодировки)

- Нельзя пробелы

- Не рекомендуется \_

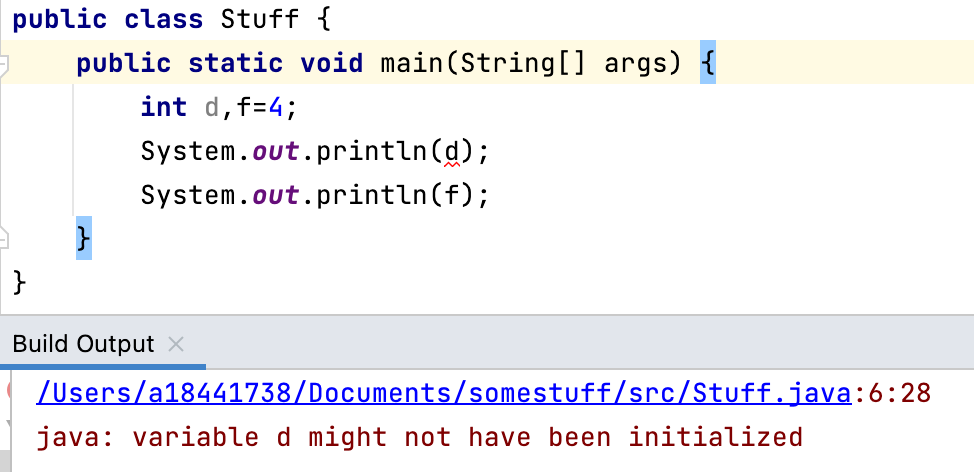
- Расположение проекта можно регулировать

10. Лучше не в src сразу фигачить, а еще доппэкедж создавать внутри для классов

11

Main может быть сколько угодно, но главный класс может быть только один. Программа начинает выполнение не с main а с главного класса. Обычно мэйн только один

12 помни





**13**

**вот такая запись 1.2e-100 или 1.2e+100 это 1.2\*10 в минус сотой или сотой**

14

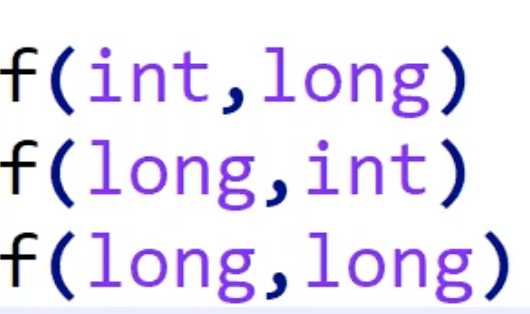
Вообще, не стоит забывать, что любое объявление переменной тратит память, и иногда бывает смысл делать объявление внутри if например, если она по коду раньше не нужна - тогда она инициализируется только если мы проскочим в этот блок.

Однако, при прочих равных, чаще всего их объявляют так, чтобы читать код удобно было.

15

в сигнатуре перегруженных может быть не только разное количество аргументов, но и разные типы аргументов, но второй вариант небезопасен, тк можно запутаться.

Самый остой – это перемена местами



- так не надо)

16

Не надо делать перегруженные методы, которые не совсем одно и то же делают. Самый лучший вариант – это либо тип аргументов разный, либо кол-во параметров разное.

17

Статическим импортом не стоит пользоваться.

Оператор import, предваряемый ключевым словом static, можно применять для импорта статических членов класса или интерфейса. Благодаря статическому импорту появляется возможность ссылаться на статические члены непосредственно по именам, не уточняя их именем класса.

18

Если мы хотим создать метод по сложению двух точек, то следует его писать не в классе где main, а в классе, который описывает точку – тогда созданная точка сможет вызывать этот метод для сложения, а аргументом этого метода будет вторая точка.

p3=p1.addTo(p2); - это To методы.

19. Массивы принято называть c s на конце (например int [] numbers)

20

- return, помимо возвращения значения, можно использовать как выход из метода, даже если метод void:

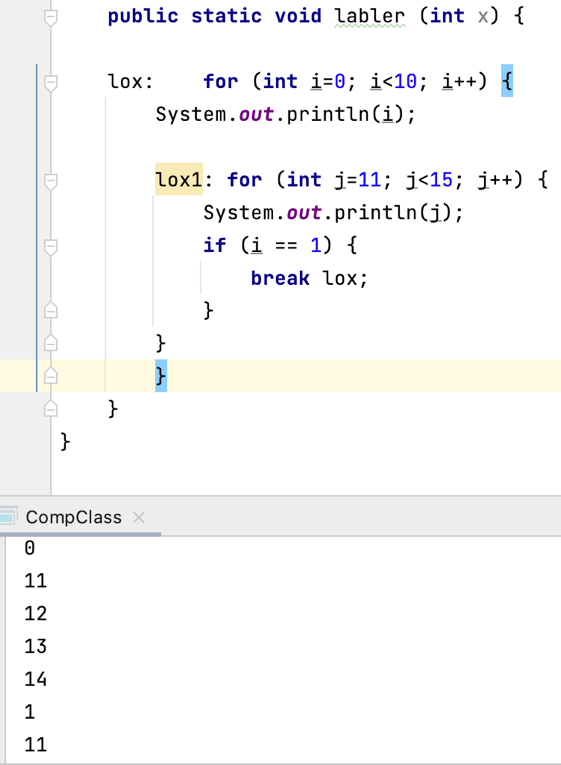
**public static void** returner (**int** a,**int** b) {  
 System.***out***.println();  
 **if** (a<b) {  
 **return**;  
 }  
 **else** {  
 System.***out***.println(**"lox"**);  
 }  
}

(по факту он всегда выполняется в конце void-метода, только неявно)

21

- goto() - этого ключевого слова нет в Java. Оно зарезервировано чтобы быть запрещенным.

Однако можно использовать его аналог – проименовать цикл с помощью метки (идентификатор для цикла с последующим двоеточием) и прыгать:



для while будет

lox:

while (true) {}

Единственная причина для использования меток в Java – наличие внутренних и внешних циклов и умная работа с continue и break в обоих из них.