|  |  |
| --- | --- |
| Review | Code review is commonly acknowledged to be one of the development stages that greatly helps to [increase software quality](https://www.devart.com/review-assistant/learnmore/benefits.html). It helps to optimize code, to detect and fix bugs early before they get to QA. And the earlier the bugs are detected the cheaper they are to fix.  There have been two methods of peer review: **formal** (or heavyweight) and **lightweight**. Formal code review takes meetings, printed copies of code, takes much of company's resources and keeps developers from actually developing the product. It gets the job done, but also slows down the development process. |
| [Lightweight peer code review](http://codingsight.com/lightweight-code-review/) | [Lightweight peer code review](http://codingsight.com/lightweight-code-review/), on the contrary, is less formal and can (and must) be a part of the common development process. Lightweight code inspection can be divided into the following types:   * *Pair programming* – two authors develop the code together at the same workstation * *Over-the-shoulder* – the code's author gives a walkthrough to a colleague * *Email pass-around* – reviewers receive code changes sent by the source code management system via e-mail, after check-ins are made * *Tool-assisted code review* |
| Post-Commit andPre-Commit Review | In this regard, there are two types of **code review: pre-commit and post-commit**. Pre- and post-commit review concepts are quite self-explanatory: pre-commit is a type of review when the code is reviewed before it goes to the main repository of the version control system. Post-commit review takes place after the code has been submitted to the public repository. |
| Pre-Commit Reviews | Pros:  * Company's coding **quality standards are met** before the work is committed to the main repository * This scenario helps to make sure the **review has been performed**, not postponed or omitted * Pre-commit reviews ensure other developers in your **team won't be affected by bugs**that may be found during a review   Cons:  * **Decreases productivity of each developer**, since further work on the submitted code is impossible until a successful review, and takes even longer if multiple reviewers are involved * After successfully passing a review, the **developer could commit a different piece of code**, by mistake or otherwise |
| Post-Commit Reviews | Pros:  * A developer can work and **commit changes to the repository continuously** * Other **team members see the code changes** and can alter their work accordingly * Some changes can be complex and require multiple steps, so it's convenient to**examine each step separately** after all of them have been committed   Cons:  * Increased **chances of poor code making it into the main repository**, hence affecting the entire team's work * When defects are found, it **may take a while** for the developer **to switch back** to the module they had been working on. |
| **Good review practices** | **Good review practices** include:   * Committing early and often. It's far easier to run multiple small reviews than a single huge one * Encouraging developers to produce **well-documented code before review** is issued * **Committing code to a 'test' branch** first and pushing it to the main one after it successfully passes the review * **Tracking defects** in order to make sure they are actually fixed after the review * **Reviewing the code**, not the authors |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1 Java8 Default methods in interfaces: purpose, main concept Oleksandr Matiash   
2 Java8 Functional interface Oleksandr Vechirko   
3 Java8 How to pass code into the method in java8 and java7 Oleksandr Vechirko   
4 OOP Whai is OOP Oleksandr Vechirko   
5 OOP What are the core concepts of OOP? Oleksandr Vechirko   
6 OOP What is Abstraction? Oleksandr Vechirko   
7 OOP What is Encapsutaion? Oleksandr Vechirko   
8 OOP What is Polymorphism? Oleksandr Vechirko   
9 OOP What is Inheritance? Oleksandr Vechirko   
10 OOP What is the difference between Inheritance and Polymorphism? Oleksandr Vechirko   
11 OOP Why Java does not support multiple inheritance? Oleksandr Vechirko   
12 Java core "if" vs "switch" Oleksandr Vechirko   
13 Java core keyword final (class, variable, parameter, method) Oleksandr Vechirko   
14 Java core Default toString implementation Oleksandr Vechirko   
15 Java core Generics (main reasons of appearance) Oleksandr Vechirko   
16 Java core Wildcards Oleksandr Vechirko   
17 Java core Default equals implementation Oleksandr Vechirko   
18 Java core Anonymous clases Oleksandr Vechirko   
19 Java core Serialization Oleksandr Vechirko   
20 Java core What is serialVersionId? Oleksii Bobko   
21 Java core Static (class, variable, method) Oleksandr Vechirko   
22 Java core Equals and hashcode contract Oleksandr Vechirko   
23 Java core What does volatile keyword means? Oleksandr Mashtaler   
24 Exceptions Exceptions hierarchy Oleksandr Vechirko   
25 Exceptions Unchecked and Checked exceptions Oleksandr Vechirko   
26 Exceptions try-catch (multy, with resources) Oleksandr Vechirko   
27 Exceptions Own exceptions Oleksandr Vechirko   
28 Collection&Map Collections hierarchy Oleksandr Vechirko   
29 Collection&Map ArrayList vs LinkedList Oleksandr Vechirko   
30 Collection&Map List vs Set Oleksandr Vechirko   
31 Collection&Map Stack vs Queue Oleksandr Vechirko   
32 Collection&Map Comparable vs Comparator Oleksandr Vechirko   
33 Collection&Map ConcurrentModificationException Oleksandr Vechirko   
34 Collection&Map Why Map doesn't inherit from Collection Oleksandr Vechirko   
35 Collection&Map Equalls and hashCode contract, why do we need it, in what collections it is important? Oleh Sukhonosov   
36 Collection&Map HashMap, TreeMap, linkedMap (differences) Oleksandr Vechirko   
37 Collection&Map Collisions in HashMap Oleksandr Vechirko   
38 Collection&Map How to sort List Oleksandr Vechirko   
39 Collection&Map Iterator (main methods) Oleksandr Vechirko   
40 Collection&Map Listiterator vs Iterator Oleksandr Vechirko   
41 Collection&Map LinkedHashSet (difference from HashSet) Oleksandr Vechirko   
42 Patterns Factory Oleksandr Vechirko   
43 Patterns Singleton Oleksandr Vechirko   
44 Patterns Adapter Oleksandr Vechirko   
45 Patterns Proxy Oleksandr Vechirko   
46 Patterns Decorator Oleksandr Vechirko   
47 Patterns Facade Oleksandr Vechirko   
48 Patterns What does Dependency Injection means? Oleksandr Mashtaler   
49 Design SOLID principles Oleksii Bobko   
50 Multithreading Two basic ways to start new thread Oleksandr Vechirko   
51 Multithreading Difference between run() and start() Oleksandr Vechirko   
52 Multithreading Difference between sleep() and wait() Oleksandr Vechirko   
53 Multithreading Difference between Runnable and Callable Oleksandr Vechirko   
54 Multithreading Volatile Oleksandr Vechirko   
55 Multithreading Mutex Oleksandr Vechirko   
56 Multithreading Deadlock Oleksandr Vechirko   
57 Multithreading Notify vs NotifyAll Oleksii Bobko   
58 Multithreading Fair vs unfair synchronization Oleksii Bobko   
59 Multithreading Atomicity and atomic variables Oleksii Bobko   
60 Multithreading What’s an Immutable Object? Oleksii Bobko   
61 Multithreading Syncronized block Oleksandr Vechirko

62 Multithreading ConcurrentHashMap vs. HashMap: Locking mechanism Oleksandr Mashtaler   
63 SQL Database normalization Oleksandr Vechirko   
64 SQL What is primary key? Oleksandr Vechirko   
65 SQL What is foreign key? Oleksandr Vechirko   
66 SQL Join types Oleksandr Vechirko   
67 SQL HAVING vs WHERE Oleksandr Vechirko   
68 SQL UNION vs UNION ALL Oleksandr Vechirko   
69 Exceptions When is "finally" block executed? Are there any cases when it is not executed? Oleksii Kovetskyi   
70 Exceptions What happens when you return some value in a "try" block, and some another value in "finally" block? Oleksii Kovetskyi   
71 DB Pessimistic locking vs Optimistic locking Oleksii Bobko   
72 DB ACID, Isolation levels, phenomena Oleksii Kovetskyi   
73 Java core Oleksii Kovetskyi   
74 Java core Is it possible to call one constructor from another? Oleksii Kovetskyi   
75 Java core Does Java automatically initialize variables? If so, with what values? Oleksii Kovetskyi   
76 Java core Is it possible to call method from constructor. Artem Lazurenko   
77 Java core Composition vs Inharitance. Incapsulation violation. Artem Lazurenko   
78 Java core How parameters are being passed in Java: by value or by reference Artem Lazurenko   
79 Java core What is memory leak. Is it possible to hava memory leak in Java. Provide examples if so. Artem Lazurenko   
80 Java core What is a JVM and why do we use it?   
81 Java core What are the parts of Java Memory Model? What does it consist of?   
82 Java core How does Java release memory?   
83   
84 Computer Science Why do we need GC? STUDRADA   
85 Computer Science What is memory fragmentation? STUDRADA   
86 Design Mutable and Immutable pros and cons. STUDRADA   
87 Java core How to create immutable object? STUDRADA   
88 Java core Shallow vs Deep object comparison. STUDRADA   
89 Java core Equals vs '==' STUDRADA   
90 Java core How to clone object? STUDRADA   
91 Java core Passing by value & by reference STUDRADA   
92 SQL UNION vs UNION ALL STUDRADA   
93 SQL What is LEFT INNER JOIN? STUDRADA   
94 SQL SQL table aliases (... FROM users 'u') STUDRADA   
95 Java core InputStream vs Reader STUDRADA   
96 Java core Read text using InputStream STUDRADA   
97 Java core What will happen if you override Thread's run() and invoke it using Thread reference? STUDRADA   
98 Java core ForkJoinPool definition & usages STUDRADA   
99 Code snippet Boxed Integer comparison using equals sign (==) with values <= 127 & > 127. (Snippet #1) STUDRADA   
100 Java core Default methods what are they and why did they appear in JDK 8? STUDRADA   
101 Java core Private interface methods. STUDRADA   
102 Java core Static interface methods. STUDRADA   
103 Java core var (jdk 10) pros and cons STUDRADA   
104 Java core What are Marker Interfaces in Java? STUDRADA   
105 Java core How would you load file contents into String? STUDRADA   
106 WEB HTTP methods STUDRADA   
107 WEB HTTP form submission (allowed methods) STUDRADA   
108 Java core Can private methods of a class be accessed from outside of a class? STUDRADA   
109 Computer Science Synchronized & Asynchronous meaning STUDRADA   
110 Computer Science Difference between parallel and concurrent STUDRADA   
111 Web List JSON datatypes STUDRADA   
112 Computer Science RandomAccess vs SequentialAccess STUDRADA   
113 Computer Science Time complexity Big O Notation STUDRADA   
114 WEB What is Rest? Rest restictions? Levels of Richardson REST Maturity Model? What is HATEOAS?   
115 WEB HTTP STUDRADA   
116 Computer Science How would you clone file with large size over 300GB?   
117 Computer Science What is the class? What is the difference between class and object? Anna Zakrevska   
118 Java core What is the "autoboxing" + difference between object and primitive Anna Zakrevska

119 Java core What types of classes do you know? (nested...) Anna Zakrevska   
120 Code snippet What will happen with stream? (Snippet #3) STUDRADA   
121 Java core What is enum? How it works internally? What Enum class serves for? Oleksii Bobko   
122 Lambdas What is the purpose of terminal operation? Oleh Sukhonosov   
123 Lambdas What can you do with stream without terminal operation? Oleh Sukhonosov