Clean Code

|  |  |
| --- | --- |
| **Checklist Item** | **Category** |
| Use Intention-Revealing Names | Meaningful Names |
| Pick one word per concept | Meaningful Names |
| Use Solution/Problem Domain Names | Meaningful Names |
| Classes should be small! | Classes |
| Functions should be small! | Functions |
| Do one Thing | Functions |
| Don't Repeat Yourself (Avoid Duplication) | Functions |
| Explain yourself in code | Comments |
| Make sure the code formatting is applied | Formatting |
| Use Exceptions rather than Return codes | Exceptions |
| Don't return Null | Exceptions |

\* Reference: <http://techbus.safaribooksonline.com/book/software-engineering-and-development/agile-development/9780136083238>

Security

|  |  |
| --- | --- |
| **Checklist Item** | **Category** |
| Make class final if not being used for inheritance | Fundamentals |
| Avoid duplication of code | Fundamentals |
| Restrict privileges: Application to run with the least privilege mode required for functioning | Fundamentals |
| Minimize the accessibility of classes and members | Fundamentals |
| Document security related information | Fundamentals |
| Input into a system should be checked for valid data size and range | Denial of Service |
| Avoid excessive logs for unusual behavior | Denial of Service |
| Release resources (Streams, Connections, etc) in all cases | Denial of Service |
| Purge sensitive information from exceptions (exposing file path, internals of the system, configuration) | Confidential Information |
| Do not log highly sensitive information | Confidential Information |
| Consider purging highly sensitive from memory after use | Confidential Information |
| Avoid dynamic SQL, use prepared statement | Injection Inclusion |
| Limit the accessibility of packages,classes, interfaces, methods, and fields | Accessibility Extensibility |
| Limit the extensibility of classes and methods (by making it final) | Accessibility Extensibility |
| Validate inputs (for valid data, size, range, boundary conditions, etc) | Input Validation |
| Validate output from untrusted objects as input | Input Validation |
| Define wrappers around native methods (not declare a native method public) | Input Validation |
| Treat output from untrusted object as input | Mutability |
| Make public static fields final (to avoid caller changing the value) | Mutability |
| Avoid exposing constructors of sensitive classes | Object Construction |
| Avoid serialization for security-sensitive classes | Serialization Deserialization |
| Guard sensitive data during serialization | Serialization Deserialization |
| Be careful caching results of potentially privileged operations | Serialization Deserialization |
| Only use JNI when necessary | Access Control |

 \* Reference: <http://www.oracle.com/technetwork/java/seccodeguide-139067.html>

Performance

|  |  |
| --- | --- |
| **Checklist Item** | **Category** |
| Avoid excessive synchronization | Concurrency |
| Keep Synchronized Sections Small | Concurrency |
| Beware the performance of string concatenation | General Programming |
| Avoid creating unnecessary objects | Creating and Destroying Objects |

\* Reference: <http://techbus.safaribooksonline.com/book/programming/java/9780137150021>

General

|  |  |
| --- | --- |
| **Category** | **Checklist Item** |
| Use checked exceptions for recoverable conditions and runtime exceptions for programming errors | Exceptions |
| Favor the use of standard exceptions | Exceptions |
| Don't ignore exceptions | Exceptions |
| Check parameters for validity | Methods |
| Return empty arrays or collections, not nulls | Methods |
| Minimize the accessibility of classes and members | Classes and Interfaces |
| In public classes, use accessor methods, not public fields | Classes and Interfaces |
| Minimize the scope of local variables | General Programming |
| Refer to objects by their interfaces | General Programming |
| Adhere to generally accepted naming conventions | General Programming |
| Avoid finalizers | Creating and Destroying Objects |
| Always override hashCode when you override equals | General Programming |
| Always override toString | General Programming |
| Use enums instead of int constants | Enums and Annotations |
| Use marker interfaces to define types | Enums and Annotations |
| Synchronize access to shared mutable data | Concurrency |
| Prefer executors to tasks and threads | Concurrency |
| Document thread safety | Concurrency |
| Valid JUnit / JBehave test cases exist | Testing |

\* Reference: <http://techbus.safaribooksonline.com/book/programming/java/9780137150021>

Static Code Analysis

|  |  |
| --- | --- |
| **Category** | **Checklist Item** |
| Check static code analyzer report for the classes added/modified | Static Code Analysis |

Published at DZone with permission of its author, [Mahesh Chopker](http://java.dzone.com/users/mchopker).