Code Quality Java

Sukesh Kancharla

1. Bugs:
    - Prepared Statement and ResultSet methods should be called with valid indices:
        - Prepared statement indices start with 1 (not with 0). Even it throws an exception when the index crosses the upper bound. ResultSet indices also start with 1. We need to be more careful while using indices with these.
    - "wait" should not be called when multiple locks are held:
        - Wait releases only one lock when 2 locks are held. The other lock will be released only when other thread requests for this object which may not happen and in-turn results into a deadlock.
    - Resources should be closed:
        - Connection objects, Streams, Files which are implementations of closable objects should be closed before leaving the program. It is advisable to use try catch block always while requesting for a resource.
    - Loops should not be infinite:
        - Every loop should have an end condition. Program will get out of the loop only when an end condition is met or break is executed. If program enters into an infinite loop, we have to kill.
    - “=+" should not be used instead of "+=":
        - =+, =-, =! Should not be used in place of +=,-=,!= and there is ambiguity using =+ where if we give a space between two operators, the behavior will be different.
    - "Iterator.hasNext()" should not call "Iterator.next()":
        - We should not call next() in hasNext() function because next() will make iterator to forward an item Which is not expected from a hasNext(). This may change the expected behavior.
    - "equals" method overrides should accept "Object" parameters:
        - Equals method will be overridden from object class which takes Object parameter. We override this method to compare specific class types. It is always preferable to have a overridden method which accepts Object parameter.
    - All branches in a conditional structure should not have exactly the same implementation:
        - This is a bug because it violates the need of selection statements itself. It may be because of a copy paste error or something else has to be changed with other.
    - Collections should not be passed as arguments to their own methods:
        - Some methods may expect the arguments to be unmodified. If we pass collections to those methods may result in unexpected method.
    - Conditionally executed blocks should be reachable:

    - Conditions which always gives false or true should not be used because it may end up with dead code or wastage of a statement execution.

2. Vulnerabilities:
    - Credentials should not be hard-coded:
        - Credentials should not be hard coded because one can get those from the application compiled version which ends up credentials in the hands of attacker.
    - Defined filters should be used:
        - Filters declared in web.xml should be defined in filter-mapping or else those filters may not be invoked.
    - Cookies should be "secure":
        - Cookies should not be sent over plain text transfer where they can be accessed by attackers. We can use secure cookies which makes the transfer secure using http encryptions.
    - IP addresses should not be hardcoded
        - Because every time address change leads to the requirement of compilation. It forces to use same IP always.
    - Return values should not be ignored when they contain the operation status code:
        - When a method returns a status code, it cannot be ignored because status code gives the information about the success of the method. Example methods:- Iterator.hasNext(), Lock.tryLock();
    - Untrusted data should not be stored in sessions:
        - Data stored in session should be trust worthy and storing unauthenticated users may lead to misuse of data.
    - Member variable visibility should be specified:
        - If you don’t declare visibility of a member may lead to unexpected results. It is always preferable to declare visibility.
    - Mutable members should not be stored or returned directly:
        - Mutable objects should not be returned or accepted directly because it may lead to unexpected behavior.
    - Only standard cryptographic algorithms should be used:
        - Usage of non-standard algorithms may lead attackers to break the algorithm. So standard algorithms are suggested always.
    - Mutable fields should not be "public static":
        - Mutable fields should be moved into classes to reduce its visibility and avoid unexpected behavior.

3. Code Smells:
    - "switch" statements should not contain non-case labels:
        - Non-case labels should not be used in switch case.
    - Future keywords should not be used as names:

    - Using words which are keywords in the latest versions but not in older ones may make the code to be compiled and executed in older versions of java but it is not preferable.
- Short-circuit logic should be used in boolean contexts:
    - Usage of short-circuit logic is preferred because it helps us to reduce the execution time in some cases.
- Methods returns should not be invariant:
    - Method should not return a invariant value which may be a bug. It may be poor design but it should not affect the functionality.
- TestCases should contain tests:
    - If you have a Junit TestCase, it should have test methods. TestCase having no methods makes others think that the tests on that class has been covered which makes that module untested.
- "for" loop increment clauses should modify the loops' counters:
    - If we are not incrementing loop counters in increment clauses may lead to infinite loops.
- "indexOf" checks should not be for positive numbers:
    - Return value from indexOf function should be compared with zero and positive numbers because zero is also a valid index for an array or collection.
- String literals should not be duplicated:
    - We should not duplicate string literals because if we want to change values, refracting is difficult.
- "switch" statements should end with "default" clauses:
    - An action has to be taken in switch case and it is suggested to show why no action has been performed.
- "@Override" should be used on overriding and implementing methods:
    - Helps when method is misspelled by giving a warning from compiler and makes the code readable as well.