1. ***Variable, Attribute, and Constant Declaration Defects (VC):***

Descriptive variable and constant names used in accord with naming conventions.

Example: List<String[]> csvData = new ArrayList<String[]>();

No confusingly similar names for variables or attributes.

Example: List<String[]> csvData and String[] row.

Correct typing for variables and attributes.

Example: List<String[]> csvData.

Variables and attributes properly initialized.

Example: String[] row = null;.

No non-local variables that could be made local.

For-loop control variables declared in the loop header.

Example: for(int i = 0; i < csvData.size(); i++).

Literal constants could be named constants (e.g., replace "startup\_funding.csv" with a named constant).

No attributes that should be local variables.

Example: No instance variables in the provided snippet.

Attributes have appropriate access modifiers.

Example: public static List<Map<String, String>> where(...).

No check for static attributes that should be non-static or vice-versa.

1. ***Method Definition Defects (FD):***

Descriptive method names used in accord with naming conventions.

Example: public static List<Map<String, String>> where(...).

Method parameter values checked before being used.

Example: if(options.containsKey("company\_name")).

No check for correct values at every method return point.

Methods have appropriate access modifiers.

Example: public static List<Map<String, String>> where(...).

No check for static methods that should be non-static or vice-versa.

1. ***Class Definition Defects (CD):***

No check for appropriate constructors and destructors.

No check for common members that should be in the superclass.

No check for simplifying the class inheritance hierarchy.

1. ***Data Reference Defects (DR):***

No check for array reference bounds.

No check for non-null values in object or array references.

1. ***Computation/Numeric Defects (CN)***

No check for mixed data types in computations.

No check for overflow or underflow in computations.

No check for correctness of assumptions about order of evaluation and precedence.

1. ***Comparison/Relational Defects (CR):***

No check for correct boolean conditions.

No check for correct comparison operators.

No check for simplification of boolean expressions.

No check for improper side-effects or interchanged logical operators.

1. ***Control Flow Defects (CF):***

Best choice of looping constructs used.

Example: for(int i = 0; i < csvData.size(); i++).

No check for loop termination.

No check for multiple exits from loops.

No check for default case in switch statement.

No check for missing switch case break statements.

No check for correct named break statements.

No check for nested if statements or converting them into switch statements.

No check for handling exceptions.

No check for method termination.

1. ***Input-Output Defects (IO):***

Files opened before use.

Example: CSVReader reader = new CSVReader(new FileReader("startup\_funding.csv"));.

No check for consistency in input object attributes.

Files closed after use.

Example: reader.close();.

No check for spelling or grammatical errors in printed or displayed text.

No check for handling I/O exceptions.

1. ***Module Interface Defects (MI):***

No check for agreement in method call parameters.

No check for consistency in units.

No check for changes in objects or arrays passed to methods.

1. ***Comment Defects (CM):***

No check for appropriate header comments for methods, classes, and files.

No check for comments on attribute, variable, and constant declarations.

No check for consistency between comments and code.

No check for comments helping in understanding the code.

No check for an adequate number of comments.

1. ***Layout and Packaging Defects (LP):***

No check for standard indentation and layout.

No check for method length.

No check for module length.

1. ***Modularity Defects (MO):***

No check for low coupling between modules.

No check for high cohesion within each module.

No check for repetitive code that could be replaced with method calls.

No check for appropriate use of Java class libraries.

1. ***Storage Usage Defects (SU):***

No check for array size.

No check for setting object and array references to null.

1. ***Performance Defects (PE):***

No check for better data structures or more efficient algorithms.

No check for arranging logical tests for efficiency.

No check for reducing the cost of recomputing values.

No check for storing and using computed values.

No check for optimizing loop operations.

No check for unrolling short loops.

No check for combining two loops operating on the same data.

No check for declaring frequently used variables register.

No check for declaring short and commonly called methods inline.