- Two public classes in the same file.
- Main method calling a non-static method.
- Methods with the same name as the constructor(s).
- Thread initiation with classes that don't have a run() method.
- Local inner classes trying to access non-final vars.
- Case statements with values out of permissible range.
- Math class being an option for immutable classes !!
- instanceOf is not same as instanceof
- Private constructors
- An assignment statement which looks like a comparison if (a=true)...
- System.exit() in try-catch-finally blocks.
- Uninitialized variable references with no path of proper initialization.
- Order of try-catch-finally blocks matters.
- main() can be declared final.
- -0.0 == 0.0 is true.
- A class without abstract methods can still be declared abstract.
- Map doesn't implement Collection.
- Dictionary is a class, not an interface.
- Collection is an Interface where as Collections is a helper class.
- Class declarations can come in any order (derived first, base next etc.).
- Forward references to variables gives compiler error.
- Multi dimensional arrays can be sparce ie., if you imagine the array as a matrix, every row need not have the same number of columns.
- Arrays, whether local or class-level, are always initialized,
- Strings are initialized to null, not empty string.
- An empty string is NOT the same as a null string.
- A declaration cannot be labelled.
- continue must be in a loop (for, do , while). It cannot appear in case constructs.
- Primitive array types can never be assigned to each other, eventhough the primitives themselves can be assigned. ie., ArrayofLongPrimitives = ArrayofIntegerPrimitives gives compiler error eventhough longvar = intvar is perfectly valid.
- A constructor can throw any exception.
- Initilializer blocks are executed in the order of declaration.
- Instance initializer(s) gets executed ONLY IF the objects are constructed.
- All comparisons involving NaN and a non-Nan would always result false.
- Default type of a numeric literal with a decimal point is double.
- integer (and long) operations / and % can throw ArithmeticException while float / and % will never, even in case of division by zero.

- == gives compiler error if the operands are cast-incompatible.
- You can never cast objects of sibling classes(sharing the same parent), even with an explicit cast.
- .equals returns false if the object types are different. It does not raise a compiler error.
- No inner class can have a static member.