net.sf.freecol.FreeCol:

Large Class:

The FreeCol class contains a large number of methods and properties, which might indicate that it's doing too many things.

Duplicated Code:

There are instances of duplicated code, such as similar error handling patterns found in different methods.

Duplicated code can be a hard to mantain as it requires updates in multiple places if logic changes. My suggestion is to extract the duplicated code and handle the situation with a more general solution.

Net.sf.freecol.server.generator.TerrainGenerator:

Long Method:

The generateMap method is quite lengthy, performing multiple tasks such as importing tiles, setting regions, creating mountains, rivers, lakes, and bonuses. Long methods can be harder to understand, debug, and maintain. Breaking down this method into smaller, more focused methods, each responsible for a specific task, would improve readability and make the code easier to manage.

Data Clumps:

There are multiple groups of related parameters used across methods, such as latitude-related parameters. These data clumps suggest that certain parameters might be better organized into objects or data structures, creating classes or structures to encapsulate related parameters, would make the code more organized and selfexplanatory.

For example, latitude is passed to methods like getRandomLandTileType and getRandomOceanTileType.

Feature Envy:

Methods like createMountains, createRivers, and createLakeRegions heavily depend on properties and methods of the Map class. While the logic is encapsulated within these methods, the heavy reliance on external class properties may indicate a form of feature envy. Feature envy occurs when a method accesses properties or methods of another class more than its own class. To address this, i would move some of the logic into the Map class itself, promoting encapsulation and better object-oriented design principles. Methods like createMountains, createRivers, and createLakeRegions heavily use properties and methods of the Map class, such as getGame(), getSpecification(), and getType().

Net.sf.freecol.server.generator.SimpleMap:

Long Method:

The createEuropeanUnits method is quite long and performs multiple tasks, including handling different types of units, selecting starting positions, and checking various conditions. Long methods can be hard to understand, maintain, and test. My suggestion is refactoring this method into smaller, more focused methods that handle specific tasks.

Data Clumps:

There are instances of using groups of related data as method parameters, such as generateSkillForLocation taking Map, Tile, and NationType as parameters. This indicates a data clump, where certain groups of parameters are frequently passed together. Encapsulating related parameters into a class or structure to improve code readability and maintainability.

Feature Envy:

The createDebugUnits method seems to be more interested in the Player and Unit objects' data than its own class's data. It may be a sign that the method belongs to a different class or that the responsibilities need to be redistributed.