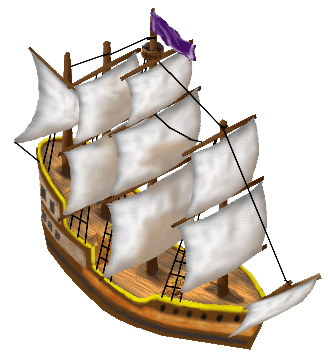
ENGENHARIA DE SOFTWARE

Licenciatura em Engenharia Informática



Freecol

Final delivery

Miguel Matos Barreto, nº61891

Índice

[Introduction 4](#_Toc152362498)

[1st Phase 5](#_Toc152362499)

[User Stories 5](#_Toc152362500)

[2nd Phase 6](#_Toc152362501)

[Code Metrics 6](#_Toc152362502)

[Design Patterns 7](#_Toc152362503)

[Code Smells 8](#_Toc152362504)

[Use Case Diagram 9](#_Toc152362505)

[3rd Phase 10](#_Toc152362506)

# Introduction

The goal of this project is to work as a team and develop good working environment to improve an open source project called Freecol.

Our group had a split After the 2nd deliverable was delivered.

We forked the repository and continued our work, our could not develop the 3 user stories that were defined while we were in the previous group, the tutorial mission that is implemented in our project is code that belongs to the previous group came when we forked the code, it has the beggining of the tutorial logic, and works for the first mission.

Our group develop the 1st user story funcionality:

* As a user, I wish the game to include special tiles with unique effects to make the gameplay more varied and strategic.

We implemented a funcionality where when you get a unit move into a river tile it triggers an event, the event has 85% chance to fail and 15% chance to give you a random amount of gold between 0 and 145.

# 

# 1st Phase

## User Stories

The user stories were made in the previous group and when we forked the project we kept them.

### 1st User Story

As a user I wish the game to include special tiles with unique effects to make the gameplay more varied and strategic.

### 2nd User Story

As a player I want the ability to deepen my interactions with the native characters in the game to enrich the narrative

### 3rd User Story

As a new player I want a set of starting missions to provide me with essential information and tips, so I can so I can quickly grasp the basic gameplay concepts without feeling overwhelmed.

# 2nd Phase

## Code Metrics

#### Miguel Barreto, nº 61891, mm-barreto

### Lines of Code Metric

The Lines of Code metric measures the total number of lines within different parts of a codebase. It's a quantitative measure used to express the size of a codebase.

It includes the following metrics:

* CLOC (Comment Lines of Code): The number of lines of code which are comments. Used to assess documentation quality within the code.
* JLOC (Java Lines of Code): Specific to Java, this metric counts the lines of Java code.
* LOC (Lines of Code): The total number of lines in a code segment, including comments and whitespace.
* NCLOC (Non-Comment Lines of Code): The number of lines of actual code, excluding comments and blank lines.
* RLOC (Relative Lines of Code): Indicates the proportion of the total lines of code that are actual, non-comment code.

A pie chart with different colors

Description automatically generatedAfter running the plugin on our codebase and looking specifically at the Non-Comment Lines of Code (NCLOC) metric by package, we can visualize the data with the following pie chart:

Upon inspecting the pie chart, we can observe that **net.sf.freecol.common.model** has a larger segment, suggesting that it contains more lines of code compared to other packages.

This could potentially be a trouble spot.

Packages with large amounts of code can be harder to maintain and understand and be more prone to bugs.

It relates to code smells like Long Methods and Long Classes.

## Design Patterns

#### Miguel Barreto, nº 61891, mm-barreto

GoF Patterns

**Singleton Pattern**

**Location:**

* **src/net/sf/freecol/tools/FSGConverter.java**

**Code snippet:**

A screenshot of a computer program

Description automatically generated

**Explanation:**

* The identification of this pattern is quite straightforward, we can easily identify the unique instance of the class and the lazy constructor, assuring that there is always only one instance of this class.

**Decorator Pattern**

**Location:**

* **src/net/sf/freecol/common/model/TileImprovementStyle.java**

**Code snippet:**

Uma imagem com texto, eletrónica, captura de ecrã, computador

Descrição gerada automaticamente

**Explanation:**

* The comments help us understand this pattern, it represents an additional feature of a tile, and it allow us to create different types of new features.

## Code Smells

#### Miguel Barreto, nº 61891, mm-barreto

*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.

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

Long Method:

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamenteThe generateMap method is quite lengthy, performing multiple tasks such as importing tiles, setting regions, creating mountains, rivers, lakes, and bonuses.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

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.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

*For example, latitude is passed to*

*methods like getRandomLandTileType*

*and getRandomOceanTileType.*

*Net.sf.freecol.server.generator.SimpleMapGenerator:*

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.

Uma imagem com texto, captura de ecrã, menu

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, menu

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, menu

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Data Clumps:

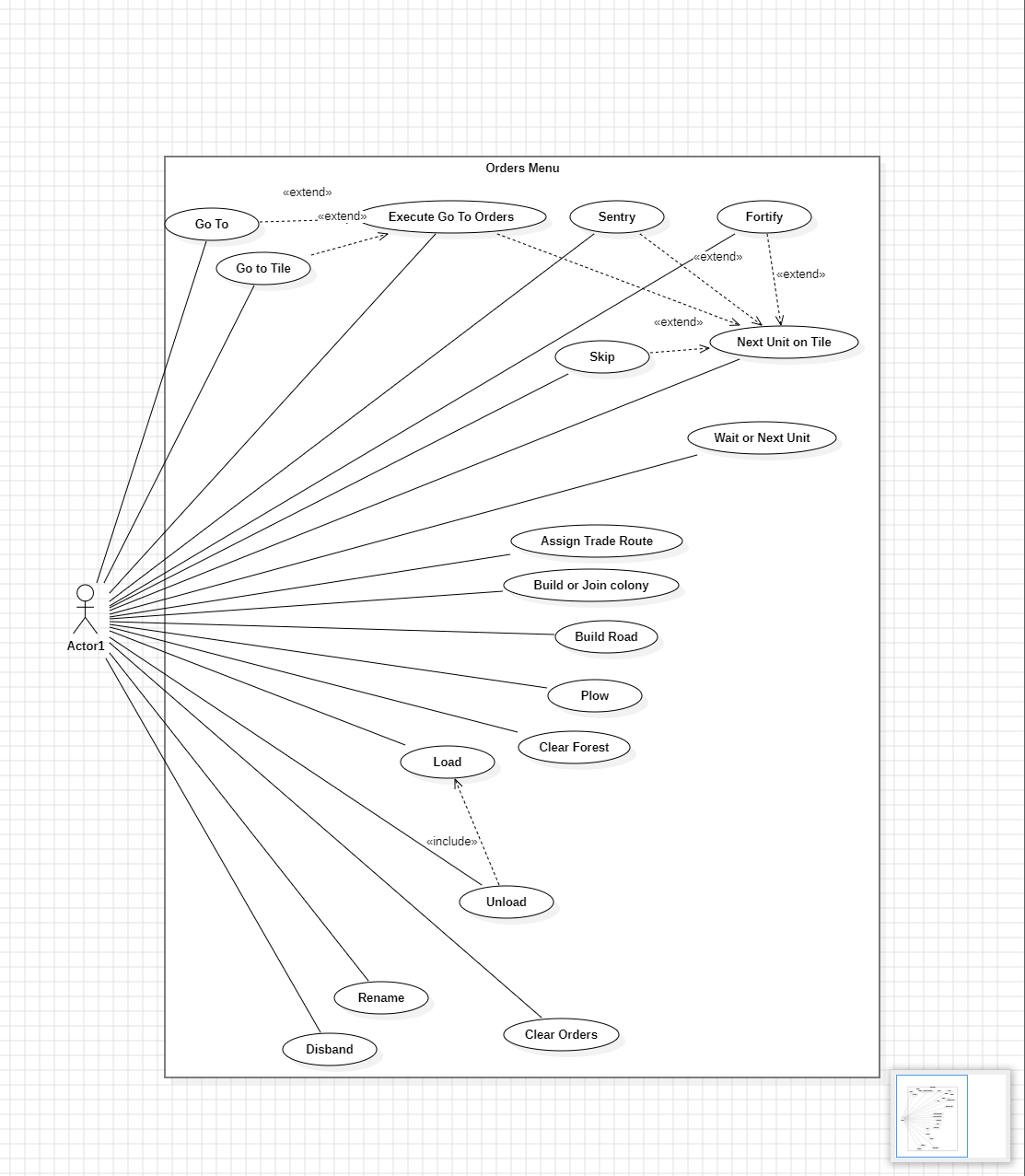
There are instances of using groups of related data as method parameters, such as generateSkillForLocation taking Map, Tile, and NationType as parameters. This indicates a data clump, where certain groups of parameters are frequently passed together. Encapsulating related parameters into a class or structure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

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

## Use Case Diagram

#### Miguel Barreto, nº 61891, mm-barreto



# Final Phase

On the 3rd phase our group was challenged with a group split, we split from the group and continued our group with only 2 members.

##### 1st User Story

We succeeded on implementing the 1st user story.

We developed a funcionality where when a unit crosses a river has 15% chance to trigger the event “you found gold in the river”.

[Demo video](https://www.youtube.com/watch?v=mbb_MfRe2C0&ab_channel=MiguelBarreto)

##### 2nd User Story

For the 2nd User Story

Our group tried to implemented a functionality where we could speak with natives that were close to a unit tile, we implemented A NativeRecruit class that would work like NativeTrade but to recruit natives, a UnitRecruitable that extends Recruitable (an abstract class created to make natives a recruitable Object. The abstract class extends FreeColGameObject and would work like the class TradeItem (that makes Goods become tradeable)) the UnitRecruitable Class would work like NativeTradeItem but to make units recruitable.

We could not make this work due to time shortage.

##### 3rd User Story

For the 3rd User Story

The 3rd user story was started in the old group by a non member of the new group, and all code made for the tutorials was not made by us, we kept the code because it existed at the moment of the fork.