Logistics:

1. When will a formal code inspection be warranted?

   During merge requests and at least weekly in case of no merge requests.

2. Who will take the lead on moderating inspections?

   Any of the people not working on the code being reviewed.

3. How will you share the results of inspections?

   Comments on the commit.

Criteria:

4. What are the key "code smells" from each chapter of Clean Code? **This is the big question.** Chapter 17 of Clean Code might help you recall them.
   a. Ch 2 - Names
      i. Single letter names
      ii. Ambiguous names
      iii. Encodings / Acronyms
      iv. Non-intention revealing terminology
      v. Misinformative names
      vi. Member prefixes
      vii. Not pronounceable
      viii. Mental Mapping
      ix. Non-consistent naming
      x. Puns
      xi. Humor
   b. Ch 3 - Functions
      i. Long functions with multiple purposes
      ii. Sections within Functions
      iii. Multiple levels of abstraction per function
      iv. Functions with side-effects
      v. Too many arguments
      vi. Switch statements with abstraction
      vii. Not separating command from query
      viii. Duplicate code
      ix. Preferring Error Codes to exceptions
   c. Ch 4 - Comments
      i. Inappropriate Information
      ii. Obsolete Comment
      iii. Redundant Comment

        iv. Javadocs for non public API methods
        v. Commented out code
    d. Ch 5 - Formatting
        i. Long distance between dependent functions
        ii. Long lines of code
        iii. Horizontal Alignment
        iv. Not following the newspaper Metaphor
        v. Not using consistent styling
    e. Ch 6 - Demeter
        i. Trainwrecks
        ii. Hybrids
        iii. Getters and setters which don't provide access control/abstraction in objects
    f. Ch 7 - Error handling
        i. Don't pass null
        ii. Don't return null
        iii. Favor unchecked over checked exceptions
        iv. Separate error logic from happy path logic
        v. Don't return error codes
    g. Ch 10 - Classes
        i. Large classes with more than one responsibility
        ii. Classes with low cohesion (each function uses few instance variables)
    h. Ch 9 & 12 - TDD
        i. More than one concept per test
        ii. Not following F.I.R.S.T
        iii. Doesn't follow the TDD rules

5. Will everyone apply all criteria from every chapter from Clean Code? Or will each person specialize in a few criteria?

    Everyone will need to consider all criteria from every chapter from Clean Code, but different team members will focus on some chapters more than others.

Scope:

6. Will your team inspect every file in your codebase? Every file you touch in your feature branch? Or something else entirely?

    Since code inspections are completed during merge requests, the focus will be on the diffs for that respective feature.

7. Of those files, will each person look at every file in consideration? Or will your team assign different files to different people?

Each person will look at every file in consideration.

Tools:

8. To what extent can your inspection criteria be automated? Automation will increase your inspection's speed and reliability.

    Style and formatting is automated with checkstyle, TDD is inspected with PIT mutation testing, and general bugs are found with findbugs. These are setup to be run with the gradle build script

9. Which aspects of your inspection criteria will need human intervention?

    Readability, interactions with the rest of the codebase, software design practices, ensuring each JUnit test only covers one concept