SOFTWARE DEVELOPMENT METHODS

UNIVERSITÀ DEGLI STUDI DI TRIESTE


# HECKMECK

Kevin Marzio, Davide Panarella, Davide Vidmar

# Table of **contents**


# 01

# RULES

How to play Heckmeck

# HECKMECK - RULES

## Components

- 15 worm tiles numbered from 21 to 36
- 8 dice with a worm symbol instead of "6"

## Objective

- Wins who accumulate most worms when tiles run out

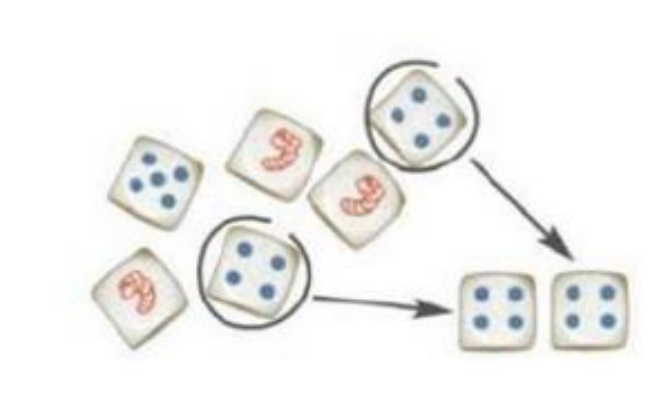
# HECKMECK - RULES

## How to pick tiles

- Dice score equal to tile number
- Collect at least a worm die

## Dice score

- Roll dice & pick a face
  (Not already taken)



# HECKMECK - RULES

**Player turn**

- Roll dice
- Pick a board tile
- Steal another player’s tile
- Bust

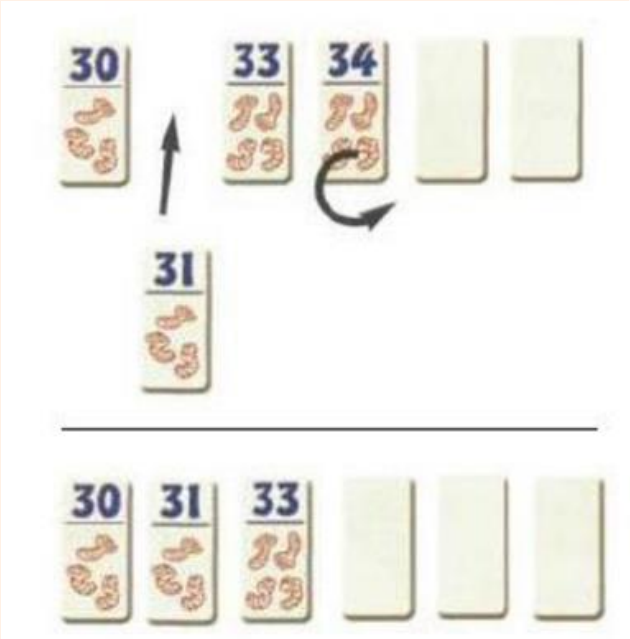
# HECKMECK - BUST

## Bust condition

- All dice faces are taken and no tiles can be picked or stolen, or no worm is chosen

## Bust Outcome

- Player places their last tile on the board (if available)
- Highest-valued tile is removed from play



# 02

# DEMO

# 03

# TOOLS

# TOOLS

**Java version**
- 18

**IDE**
- IntelliJ IDEA 2022.2.5

**Build Automation tool**
- Gradle 7.5.1

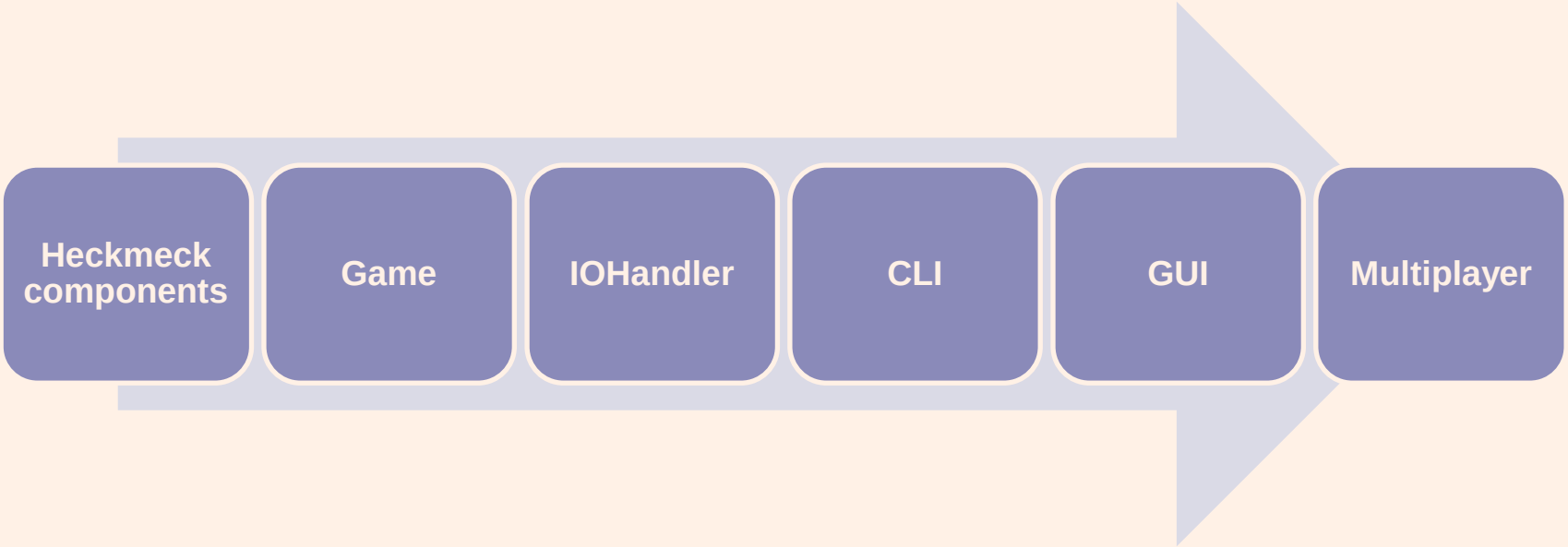
**Version Control System**
- Git + GitHub

**Continuous Integration**
- CircleCI

**Manage JSON**
- gson 2.10

**Test**
- JUnit 4, Mockito 2.22

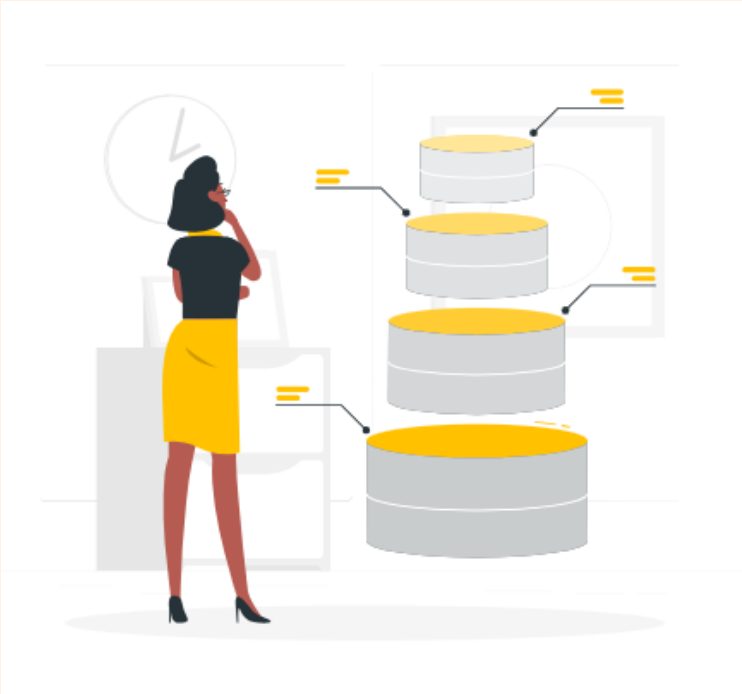
# DEVELOPMENT TIMELINE



HECKMECK

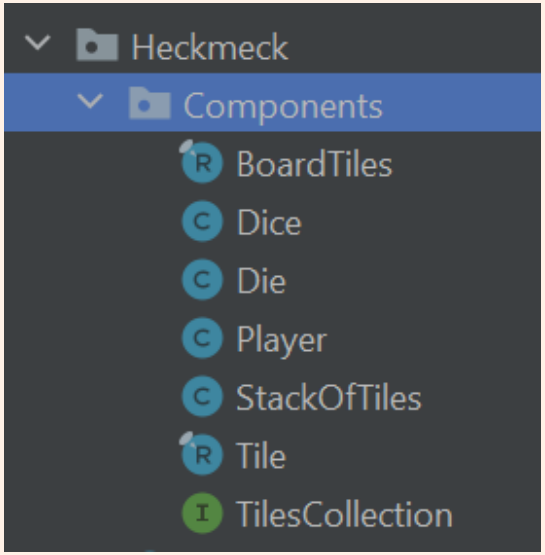
# 04

# FEATURES

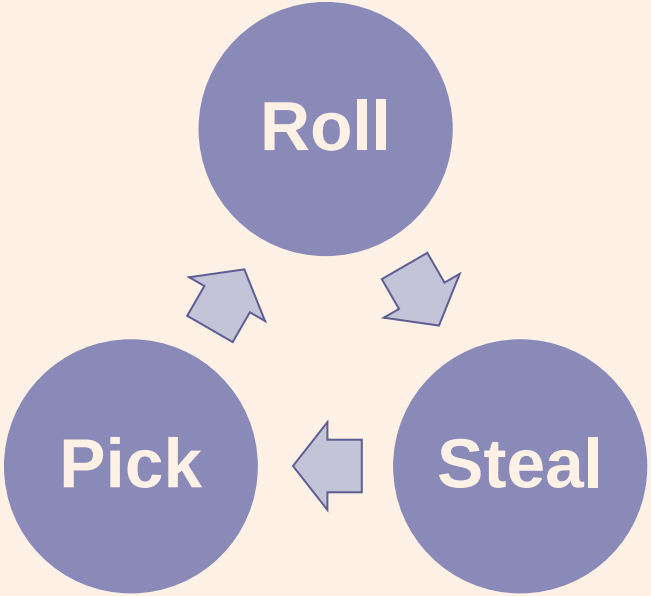


# HECKMECK - COMPONENTS

- Tile
- TileCollection
  - StackOfTiles
  - BoardTiles
- Dice
  - Die
- Player



# HECKMECK - GAME




**PlayerTurn cycle:**

- Roll the dice
- Steal check
- Pick check

Repeat until a bust or the player picks/steals.

# HECKMECK – IOHANDLER


- Users interact indirectly for enhanced flexibility
- Facilitates seamless integration with various UI alternatives

# HECKMECK - ADDITIONAL FEATURES

| | |
|---|---|
| **Properties Management** | • Handling of hardcoded strings via the PropertiesManager class |
| **Graphical User Interface** | • Introduced a GUI using java Swing for improved accessibility and visual experience |
| **Multiplayer** | • Multiplayer using TCP connections and the IOHandler interface |
| **Custom view** | • Customize game visuals through file editing |


# 05

# METHODOLOGY

# Design principles

# Open/Closed

```
4 usages   Davide Panarella +1
public Game(IOHandler io) throws IOException {
    this.io = io;
    propertiesManager = new PropertiesManager(getGameMessagePropertiesPath());
}
```

- Logic is coded inside the class
- Behavior changes according to IOHandler

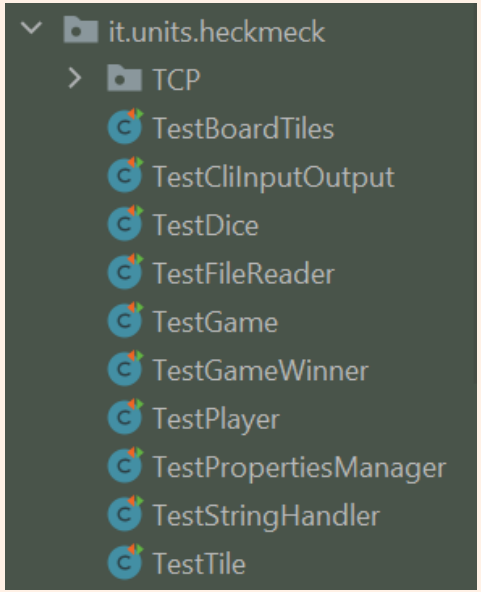
# Interface Segregation

- Implementations use all methods of the interface

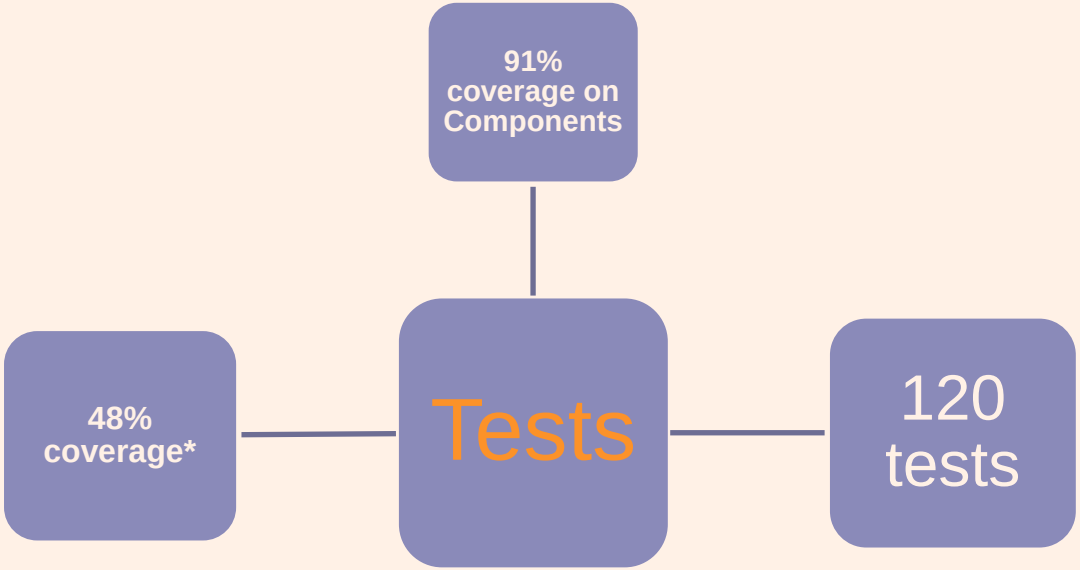
```
3 implementations   Davide Panarella +3
public interface IOHandler {
    3 implementations   Davide Panarella
    void printMessage(String message);
    3 usages   3 implementations   dew54
    void showTurnBeginConfirm(Player player);
    6 usages   3 implementations   Davide Panarella
    int chooseNumberOfPlayers();
    6 usages   3 implementations   dew54
    String choosePlayerName(Player player);
    3 usages   3 implementations   Davide Panarella
    void showBoardTiles(BoardTiles boardTiles);
    2 usages   3 implementations   Kevin Marzio
    void showRolledDice(Dice dice);
    4 usages   3 implementations   dew54
    boolean wantToPick(Player currentPlayer, int actualDiceScore, int availableTileNumber);
    2 usages   3 implementations   dew54
    boolean wantToSteal(Player currentPlayer, Player robbedPlayer);
    3 usages   3 implementations   dew54
    void showPlayerData(Player currentPlayer, Dice dice, Player[] players);
    6 usages   3 implementations   Davide Panarella
    Die.Face chooseDie(Player currentPlayer);
    2 usages   3 implementations   /ev!n
    void showBustMessage();
    7 usages   3 implementations   Davide Panarella
    void printError(String text);
    1 usage   3 implementations   /ev!n
    void backToMenu();
}
```

# Test Driven Development

- Tile
- Board Tile
- Die
- Dice
- File reading
- Player and his actions
- Properties reading



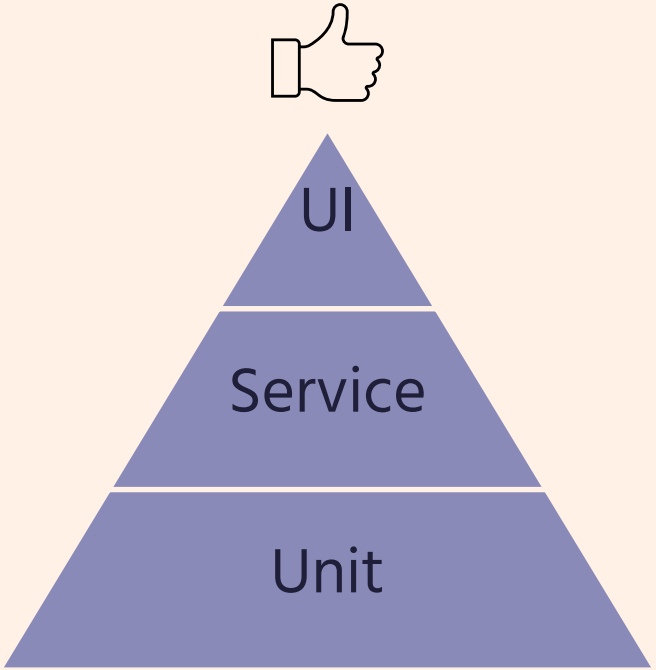
# Jacoco




*GUI excluded

# Test pyramid

- Exploratory tests
- TestCliIOHandler
- Remote game tests, TestProperties
- Tiles, dice, player

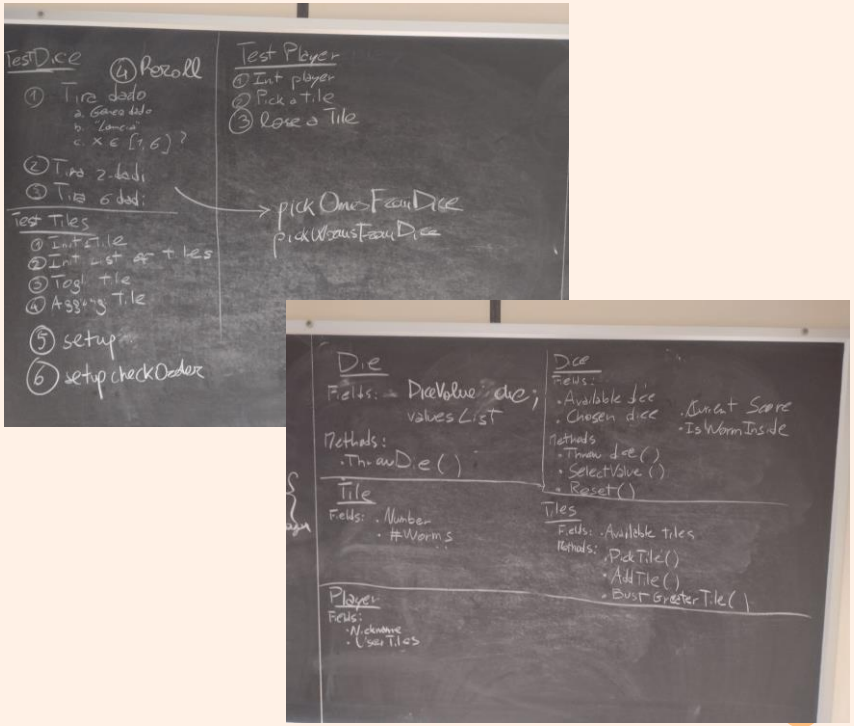


# Agile

0. Initial brain storming

1. Requirement analysis
2. Design
3. Implementation
4. Testing & Integration
5. Maintainance

6. Repeat 1-5

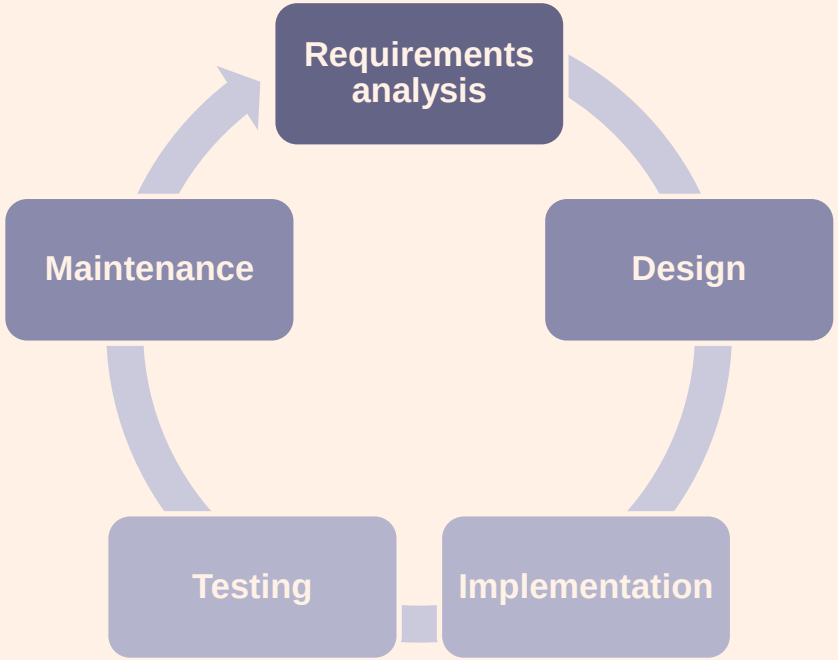


# Agile

0. Initial brain storming

1. Requirement analysis
2. Design
3. Implementation
4. Testing & Integration
5. Maintainance

6. Repeat 1-5



# Agile – Product Backlog

| | | | |
|---|---|---|---|
| **TODO** | High priority<br><br>Project blocking activities (bug fix etc.) | Medium priority<br><br>Non-blocking, new features implementation | Low priority<br><br>Non blocking, code revision |

# Agile – Scrum Board

| | To Start | Dev | Test | Done |
|---|---|---|---|---|
| Vid | Buy milk for your grandma | | | |
| Panna | | Take a shower | | |
| Kevin | | | | Prepare a carbonara with cream |

# HECKMECK – what we have learnt

- Git
- Java Swing
- TCP multiplayer
- Properties
- Threads
- Gradle
- Mockito
- CircleCI

HECKMECK

# Thanks for your attention!