Java Pedagogical Libraries for Code Analysis

Team members: Derek Chiou, Luca Doutt, Ghulam Mujtaba Qasimi, Kennedy Westry | **Faculty adviser:** Luke Gusukuma, Ph.D. | **Sponsor:** VCU College of Engineering | **Mentor:** Luke Gusukuma, Ph.D.

Problem

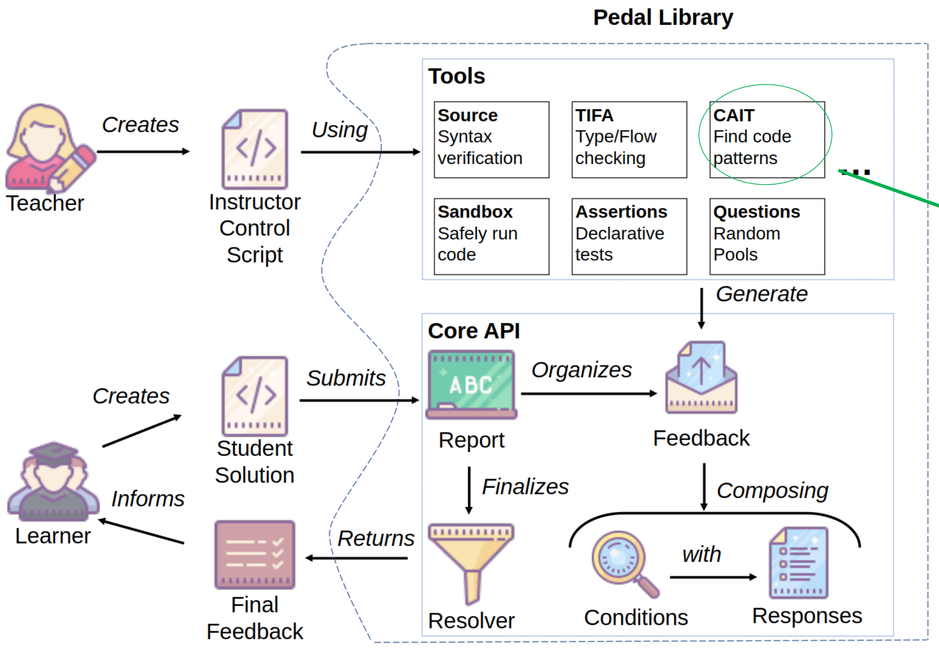
- Manual grading in programming is time-consuming and not scalable for large classes
- Current automated feedback systems typically detect surface-level syntax errors or match outputs using unit tests
- Compiler errors, by themselves, are unhelpful for beginners

Background

- **Pedal** (Pedagogical Library) by Luke Gusukuma and Austin Bart, an instructor-centric framework for automated code feedback
- Written in **Python**, works on Python files
- Many introductory courses instead use **Java**, motivated by teaching object-oriented concepts

Solution

- Create **JPedal**, the Java version of Pedal
- Implement the **CAIT (Capturing AST-Included Trees) module**, allowing the instructor to declaratively check for certain patterns in student code
- Lay the groundwork for the remaining modules
- Builds with Gradle and IntelliJ



CAIT

- Take in student submission, parsed as **Abstract Syntax Tree** by **Source** module
- Take in teacher pattern, formatted as Abstract Syntax Tree
- Attempt to **find matches**:
  - Semantically equal nodes are equal
  - Expression placeholders match any node
  - Variable placeholders match identifiers
    - Symbol table ensures consistency
- Recursively search for subtree matches
- Apply **horizontal stretching**
  - Siblings of commutative operations can be swapped (total + item -> item + total)
  - Irrelevant nodes can be deleted
- Output and save results for further interpretation

gradescope

Gradle

VCU College of Engineering

Java

asana