CS 2334: Project 1
Reading Data from Files
Self-Initiated Prone Progression Crawler

Kinematic capture suit: trunk and limb position at 50 Hz

Southerland (2012)
Project 1

• Read in and organize data from individual trials into a Java data structure

• Compute simple statistics over the trial
Project 1

Our focus:
• Load in single trial
• Each row of the data file consists of a single snapshot of the State of the infant
• For now, that State consists of the 3D position of the left and right wrists.
• 50Hz x 5 minutes = 15,000 samples of State
## Data Format

### CSV Format

- One file per trial
- Each row: data for one instant in time
- Some values are invalid
- All CSV files have the same format

<table>
<thead>
<tr>
<th>time</th>
<th><code>left_wrist_x</code></th>
<th><code>left_wrist_y</code></th>
<th><code>left_wrist_z</code></th>
<th><code>right_wrist_x</code></th>
<th><code>right_wrist_y</code></th>
<th><code>right_wrist_z</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.16</td>
<td>0.171148</td>
<td>0.290173</td>
<td>-0.12378</td>
<td>0.317962</td>
<td>-0.24333</td>
<td>-0.16704</td>
</tr>
<tr>
<td>1.18</td>
<td>0.171715</td>
<td>0.290036</td>
<td>-0.12423</td>
<td>0.31971</td>
<td>-0.24132</td>
<td>-0.16809</td>
</tr>
<tr>
<td>1.2</td>
<td>0.172315</td>
<td>0.290075</td>
<td>-0.12454</td>
<td>0.320938</td>
<td>-0.23966</td>
<td>-0.16913</td>
</tr>
<tr>
<td>1.22</td>
<td>0.172912</td>
<td>0.290134</td>
<td>-0.12502</td>
<td>0.321905</td>
<td>-0.2382</td>
<td>-0.17004</td>
</tr>
<tr>
<td>1.24</td>
<td>0.173715</td>
<td>0.290296</td>
<td>-0.12552</td>
<td>0.322622</td>
<td>-0.23678</td>
<td>-0.17103</td>
</tr>
<tr>
<td>1.26</td>
<td>0.174727</td>
<td>0.290514</td>
<td>-0.12599</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>1.28</td>
<td>0.175587</td>
<td>0.290816</td>
<td>-0.12638</td>
<td>0.324435</td>
<td>-0.23407</td>
<td>-0.17185</td>
</tr>
<tr>
<td>1.3</td>
<td>0.176138</td>
<td>0.291139</td>
<td>-0.12661</td>
<td>0.325562</td>
<td>-0.23263</td>
<td>-0.17182</td>
</tr>
<tr>
<td>1.32</td>
<td>0.176425</td>
<td>0.291356</td>
<td>-0.12685</td>
<td>0.326628</td>
<td>-0.23147</td>
<td>-0.17124</td>
</tr>
<tr>
<td>1.34</td>
<td>0.176407</td>
<td>0.291538</td>
<td>-0.12693</td>
<td>0.327646</td>
<td>-0.23003</td>
<td>-0.17042</td>
</tr>
<tr>
<td>1.36</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0.328245</td>
<td>-0.22878</td>
<td>-0.16944</td>
</tr>
<tr>
<td>1.38</td>
<td>0.176374</td>
<td>0.291932</td>
<td>-0.1275</td>
<td>0.328722</td>
<td>-0.22769</td>
<td>-0.16834</td>
</tr>
<tr>
<td>1.4</td>
<td>0.176417</td>
<td>0.292156</td>
<td>-0.12779</td>
<td>0.329146</td>
<td>-0.22695</td>
<td>-0.16705</td>
</tr>
<tr>
<td>1.42</td>
<td>0.176195</td>
<td>0.292609</td>
<td>-0.1277</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>1.44</td>
<td>0.176462</td>
<td>0.293069</td>
<td>-0.12728</td>
<td>0.329501</td>
<td>-0.22585</td>
<td>-0.16401</td>
</tr>
<tr>
<td>1.46</td>
<td>0.176843</td>
<td>0.293717</td>
<td>-0.12661</td>
<td>0.329438</td>
<td>-0.22538</td>
<td>-0.16245</td>
</tr>
</tbody>
</table>
Solution Design

```java
public class GeneralValue {
    // Properties
    private boolean valid;
    private double doubleValue;

    // Constructor
    public GeneralValue() {
        // Initialization
    }

    // Getters and Setters
    public boolean isValid() {
        return valid;
    }

    public double getDoubleValue() {
        return doubleValue;
    }

    public String toString() {
        // String representation
    }
}

public class Point3D {
    // Properties
    private GeneralValue x, y, z;

    // Constructor
    public Point3D(GeneralValue x, GeneralValue y, GeneralValue z) {
        // Initialization
    }

    // Getters and Setters
    public double getDimValue(int dimension) {
        // Get dimension value
    }

    public GeneralValue[] getValues() {
        // Get all values
    }

    public String toString() {
        // String representation
    }
}

public class State {
    // Properties
    private Point3D leftWrist, rightWrist;
    private double time;

    // Constructor
    public State(String stateString) {
        // Initialization
    }

    public Point3D getLeftWrist() {
        return leftWrist;
    }

    public Point3D getRightWrist() {
        return rightWrist;
    }

    public double getTime() {
        return time;
    }

    public String toString() {
        // String representation
    }
}

public class Trial {
    // Properties
    private ArrayList<State> stateList;
    private String infantID;
    private int week;
    private String fileName;

    // Constructor
    public Trial(String directory, String infantID, int week) {
        // Initialization
    }

    public State getState(int index) {
        return stateList.get(index);
    }

    public String getInfantID() {
        return infantID;
    }

    public int getWeek() {
        return week;
    }

    public String getFileName() {
        return fileName;
    }

    public double getMaxLeftWrist(int dimension) {
        // Get max value
    }

    public double getMinLeftWrist(int dimension) {
        // Get min value
    }

    public double getAverageLeftWrist(int dimension) {
        // Get average value
    }
}

public class Driver {
    // Main method
    public static void main(String[] args) throws IOException {
        // Main code
    }
}
```
GeneralValue Class

Two instance variables:

• value (double): value to be represented (e.g., position of wrist along the X dimension)

• valid (boolean): indicates whether the value is valid or not

Class is immutable
Point3D Class

• Represent a location in 3-space

• Accessors for individual dimensions or all three at once
State Class

Describe the state of the suit at one instant in time

• Timestamp
• Positions for both the left and right wrists

<table>
<thead>
<tr>
<th>time</th>
<th>left_wrist_x</th>
<th>left_wrist_y</th>
<th>left_wrist_z</th>
<th>right_wrist_x</th>
<th>right_wrist_y</th>
<th>right_wrist_z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.16</td>
<td>0.171148</td>
<td>0.290173</td>
<td>-0.12378</td>
<td>0.317962</td>
<td>-0.24333</td>
<td>-0.16704</td>
</tr>
</tbody>
</table>
Trial Class

Trial-related data:

• Infant ID
• Study week
• File name
• Array of States (typically 15,000 of them)
• Statistics computations
Provided Materials

• Two CSV files
• Project specification
  • Details of the requirements, including the expected `toString()` values for the classes
  • Don’t deviate from the specification – we won’t be able to compile our tests against your code
• We provide a partial implementation of some of the required classes
Testing

Implement your own JUnit tests for all key classes:
• GeneralValue
• Point3D
• State
• Trial

• Be thorough in your testing: for full credit, your tests must touch all of your lines of code (except for the Driver)
• Derive test data from the provided CSV files or create your own
CSV Files

• Data files located in your project: project1/data/

• To prevent maintaining many copies of these large files, Web-Cat will automatically not upload the data directory
  • But: it will be there when your code executes
  • For those submitting directly: make sure that you do not include the data directory

• If you need to include your own, custom CSV files, then place them in project1/mydata/
Strategies for Success

• Work with project partner in person
• Start early
• Implement and test incrementally
• Don’t deviate from our design
• Write documentation as you go
Submission

• Due date: Wednesday, September 20\textsuperscript{th} @1:29pm (before class!)
• Submit to the Web-Cat server
Grading

• Grading criteria will be similar to what we are using in the labs
• Except ...
Code Review

Each group must come in for a code review with me or one of the TAs

• 15-minutes
• Discuss documentation, implementation and performance on tests
• Both group members must be able to answer questions about all aspects of the code
• Reserve a time or walk-in
• If you complete your code review before the deadline, you may resubmit to address issues
• Code reviews must be completed by Wednesday, September 27th. Don’t wait!
Carrots and Sticks

• Bonus: 1 point for every 12 hours submitted early (up to 5)

• Penalties:
  • 0-48 hours late: 5 points every 12 hours late
  • 48 hours late: no credit
  • 1 point for every submission beyond 30 submissions

• Hints about solving test failures:
  • Your unit tests must cover at least 90% of your code before the Web-Cat server will give you any hints
  • 24 hours before the deadline, the Web-Cat server will stop giving hints