CMPT 276

Assignment 3: Code Review

Ethan Rowat & Luna Sang

1. **High coupling: Butcher probability generation (Inappropriate intimacy)**
   The Butcher class relies on probability to ensure that the Butcher's move randomly at times. The code that was responsible for generating the probability boundaries of moving randomly was placed in the Board class, which holds lots of our GUI logic and tests conditions for how to display elements on the board. This was not an ideal location to place this code as the probability was more suited to be generated by the butcher class itself. In order to fix this: the Board class had all code for probability generation removed, and function calls to trigger the butcher class to run this logic remained. Inside the butcher class, a global static array was made, along with two static methods, one to generate the probability and set it to the global variable, and the other to retrieve probability values at the specified index. This problem was a case of inappropriate intimacy, since the Board class should not have such a strong relationship with the Butcher's fields, such as the array of probabilities. The board class can now only trigger the generation of probability values (not how they are generated), and access elements of the array when it calls the butcher class method.
   COMMITS: c0a5393a9a304302ab85a0c70c312432cf180777

2. **Unnecessary if/else statements: Dead code/Redundant code**
   Within the Butcher class, one of the move methods contained a series of if/else statements that had empty statement sets. This was because the method in the condition was performing necessary steps and returning a value if it succeeded. On top of this the conditions would result in either nothing being done in case the condition was "true" (empty block), or going to the else statement where the rest of the work would be done. The solution to this was to remove all empty if/else blocks by checking if the conditions were false instead, and then moving the "else" set of statements inside of the "if" (since it will run only when not true).
   E.g.

Initial	After change
if (condition){} else { doSomthing();}	if (!condition) {doSomething();}

   COMMITS: 6bb74da5bdd31efd38d01b5bdda3c17b062c828a

3. **Poorly structured code/Duplication: Move function Switch case (Switch Statements)**
   The structure of the switch statement in one of the move methods for the Butcher, was returning a value within the statements, as well as setting a "false" value each time a condition was not true. This was revised to set a default "false" value at the top, if any

conditions were true within the switch statement, the value would be set to “true”. In each case a “break” was added and at the bottom the value was returned, instead of returning within each case. This reduced the code duplication, made the code easier to read through, and maintained the standard structure of the switch case control.
COMMITS: f7b559305a1e86460c106fa22f40437961c0a166

4. **isCoordSame or compareTo (Coordinate class): Duplication of code/Middle man**
When we went over the Butcher’s class, we found that the compareTo method used in butcher’s tracking method was very similar to the isCoordSame method (in Coordinate class) and both dealt with the comparison of two coordinates. They have similar codes but isCoordSame returns a boolean value and the compareTo method returns an int value. At the same time, isCoordSame method is using compareTo method and the other class is using isCoordSame method. Therefore a middleman method is created. We have refactored the isCoordSame method to have the same functionality but independent of the compareTo method.
COMMITS: c08513aabb38b149371257aa600761333eed8dfd

5. **Remove compareTo and Comparable interface implementation (Coordinate class): Dead code**
After refactoring the isCoordSame method, the compareTo method is not useful as the isCoordSame method has the same functionality and therefore the compareTo method could be removed.
COMMITS: c08513aabb38b149371257aa600761333eed8dfd

6. **Change the use of CompareTo method in Butcher's tracking methods**
After refactoring the isCoordSame method and removing the compareTo method, we need to modify the use of those methods in other classes. Therefore we replaced compareTo with the refactored isCoordSame method in butcher’s tracking method.
COMMITS: c7a2ddf1f7befc3b19f80b68cba02feb10f358fa

7. **Remove unnecessary else if conditions in butcher’s tracking method (Butcher class):**
The conditions after the if else statement in the butchers tracking method are true for all the time, so we refactored the method by removing the conditions and replacing it with just else to simplify the code.
COMMITS: 4a887be8b3bff7b28ffca2451baad108a3030c95

8. **Lack of documentation: Comments**
Many of the methods within the butcher class were not documented thoroughly. This led to some code being confusing although it was simple in nature. Comments were adequately added to new methods as well as to older methods throughout the Butcher class that allow a reviewer to easily understand what each line in the class is accomplishing.
COMMITS: c637c6c06459f11148608c77ca55a67a676cf549,
148918ce86537aae919088821fc0d8e55327f1cb