Music Theory Quiz

This application asks the user randomly-chosen, and randomly generated, music theory questions.The questions themselves are constructed out of different question templates with the details filled in at random. The user's answer is checked against the correct answer and feedback is given to the user to indicate if their answer was correct.

Implementation Details

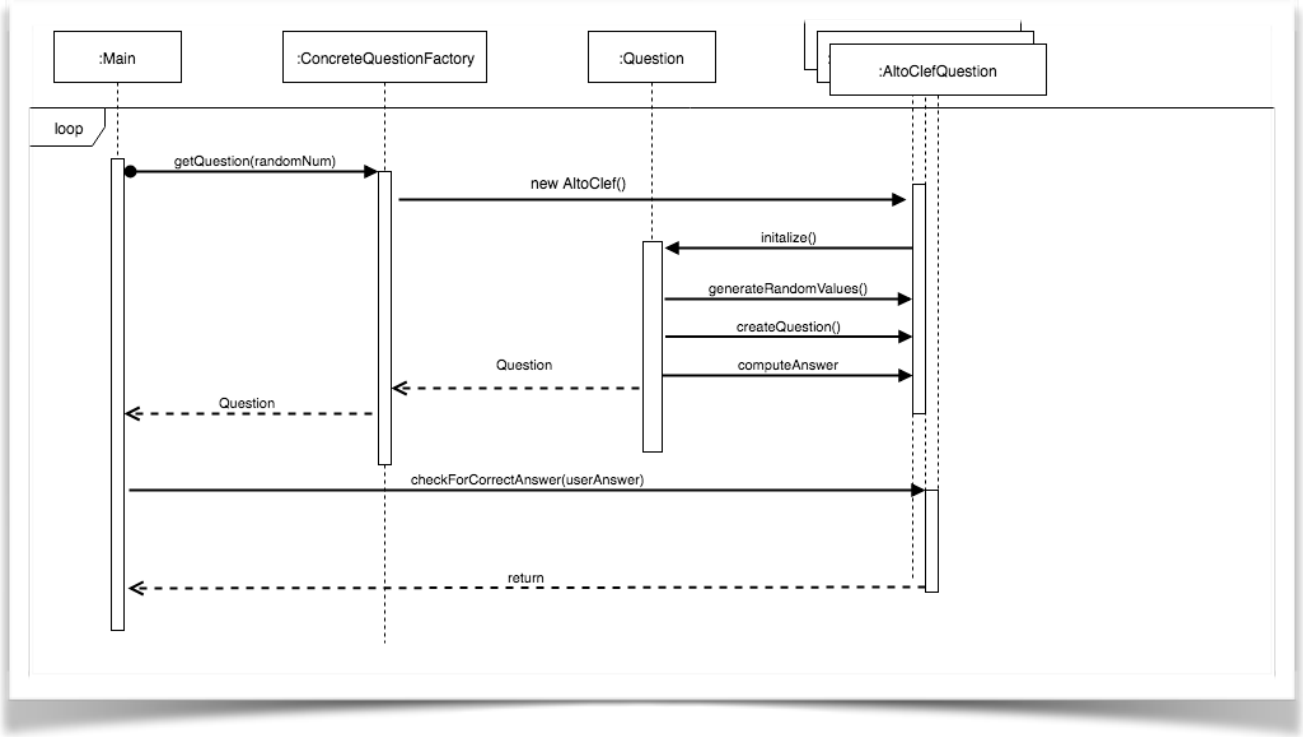
The main difficulty of this application is in generating the questions and answers with random values. I have chosen to use the template design pattern to help with the creation of these complex objects. In the case of all questions, there is a sequence of steps that occurs in the object creation. There is the generation of random values, appropriate to the type of question being generated, there is the insertion of those values into the proper place in the question template, and the computation of the answer based on the specific, randomly-generated values. The template pattern allows me to define these steps in the abstract base class and then leave the details of the question implementation to the concrete subclasses. The other issue is in randomly picking from among these concrete question subclasses. To help with this, I created a question factory that takes a number that maps to an instance of one of these concrete question classes. The flow of the generation of these questions, then, is to randomly generate a number in the main method between 0 and one less than the number of concrete question classes I have, pass that random number to my question factory, use the question factory to select a concrete question class, use the abstract question class to call the steps necessary to create the question instances, instantiate a concrete question, and return this instance back to the main.

Main Software Design Goals

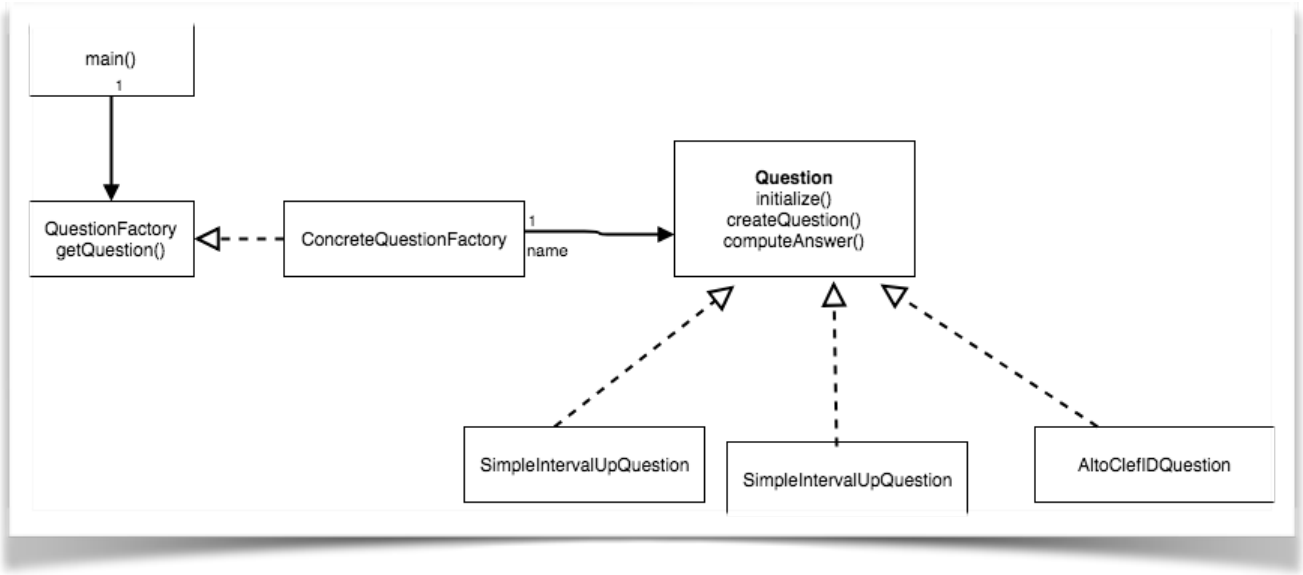
- Understandability: Because the steps to question creation are clearly defined inside the abstract base class, it is fairly easy to see how questions are constructed.

- Modularity: The question classes are highly modular and can be added and removed without affecting the other code
- Cohesion: Question functionality remains together.
- Reusability: The generation of question objects could be reused easily in other applications.

UML Diagram:



Sequence Diagram Showing Factory and Template Patterns:



Sample Output:

```
What is the note major second above Eb?
f
Answer is: f
Correct answer
Press q to quit or any other key to continue:
e
What is the note on the third line of the alto clef?
c
Answer is: c
Correct answer
Press q to quit or any other key to continue:
f
What is the note minor second above C#?
d
Answer is: d
Correct answer
Press q to quit or any other key to continue:
f
What is the note a major third below A?
c#
Answer is: c#
Incorrect.  The correct answer was: F
Press q to quit or any other key to continue:

What is the note minor second above E?
f
Answer is: f
Correct answer
Press q to quit or any other key to continue:

What is the note tritone above A?
eb
Answer is: eb
Correct answer
Press q to quit or any other key to continue:

What is the note a minor third below F?
d
Answer is: d
Correct answer
Press q to quit or any other key to continue:

What is the note perfect fourth above C#?
f#
Answer is: f#
Correct answer
Press q to quit or any other key to continue:
```