Observer Design Pattern

CSCI 2300

Announcements/Questions

- Questions about the homework?
- Submit labs 9 – 15 (this includes today's lab) by Friday, March 22.

Recall…

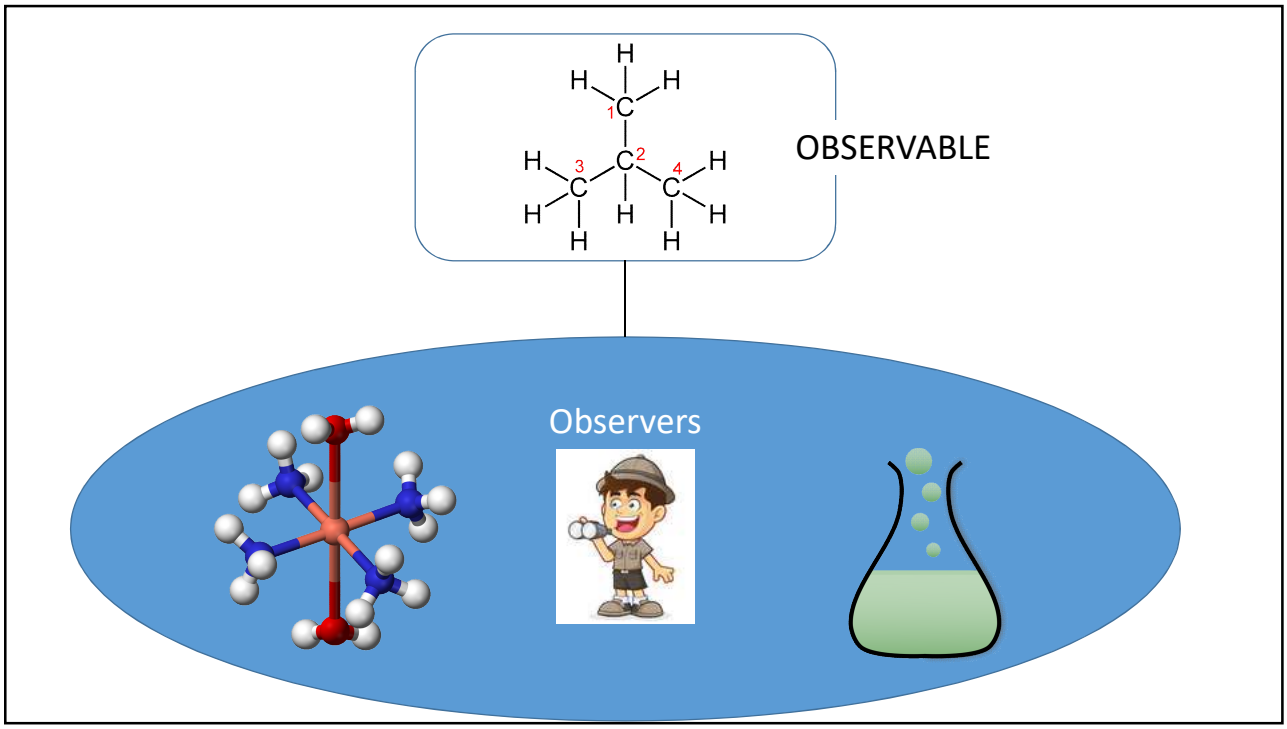
- What design pattern did we learn last time?
- What are the responsibilities of:
  - Model
  - View
  - Controller

Observer Design Pattern

Defines a one-to-many relationship between objects

One object changes state, others are notified of the change

- Model – the subject being observed (*Observable*)
- Model – notifies all *Observer(s)* if there is a change
- Observer – uses Model to update its information



Java Observable Class:

https://docs.oracle.com/javase/7/docs/api/java/util/Observable.html

Model extends **Observable** class:

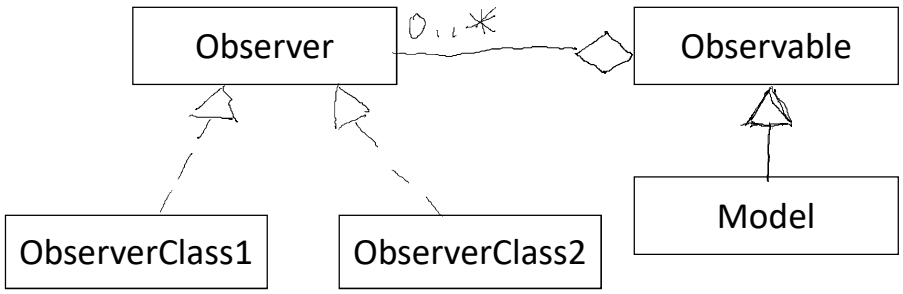
- `addObserver(Observer o)`
- `deleteObserver(Observer o)`
- `setChanged()`
- `hasChanged()`
- `clearChanged()`
- `notifyObservers()`
- `notifyObservers(Object arg)`
- `countObservers()`

Java Observer Interface:

https://docs.oracle.com/javase/7/docs/api/java/util/Observer.html

```
public void update(Observable o, Object arg)
```

Classes that want notification of Model's state changes, implement Observer interface



Call `notifyObservers()` when change happens

`notifyObservers()` will call `update()` on each observer that has been added

Math Quiz

- QuizModel has three states, implemented as enumeration type
  - NEW_QUESTION – when new question gets generated
  - CORRECT – when correct answer is submitted
  - WRONG – when wrong answer is submitted
- QuizModel extends Observable
- QuizModel has new method:

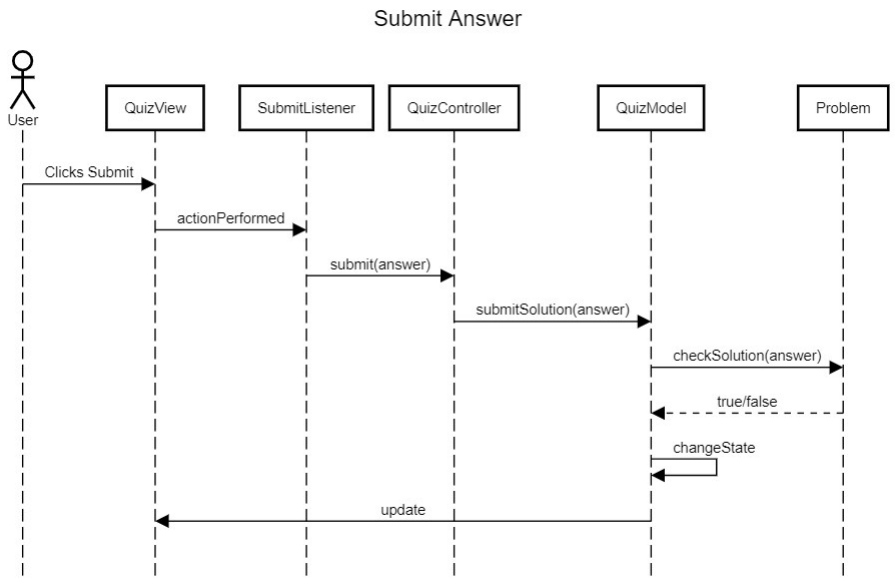
```
private void changeState(QuizState state)
{
   this.state = state;
   setChanged();
   notifyObservers(); // all observers get notified
}
```

- QuizModel has `getState()` method – returns current state

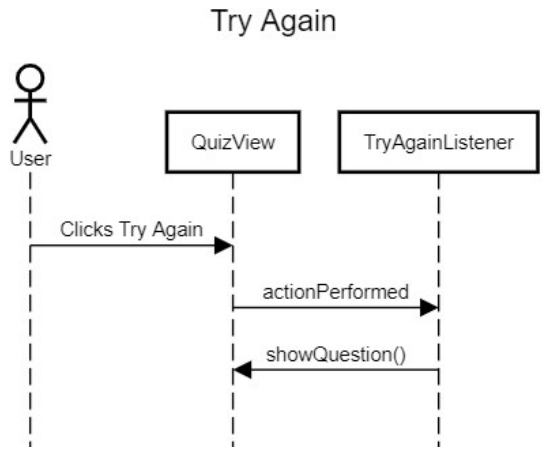
MathQuiz

- QuizView implements Observer interface

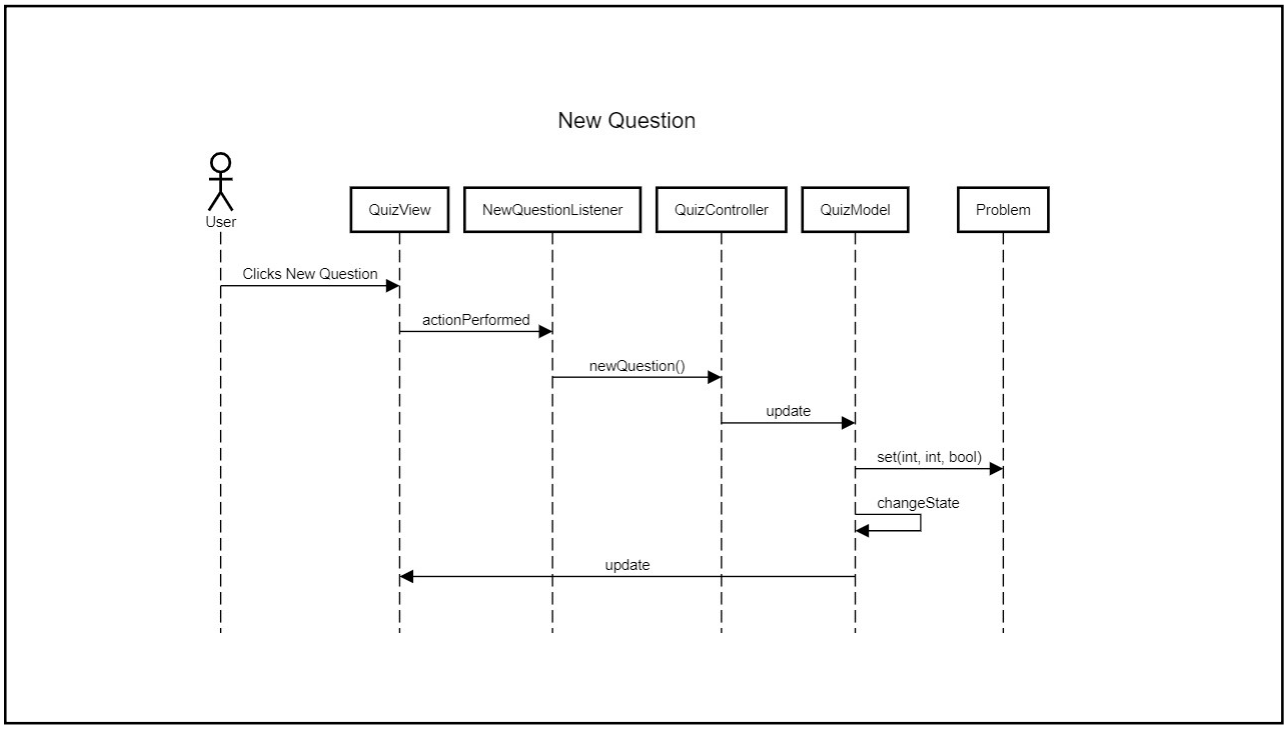
```
public void update(Observable o, Object arg)
{
   QuizState state = model.getState();
   switch (state)
   {
      case NEW_QUESTION: …
      case CORRECT: …
      case WRONG: …
   }
}
```

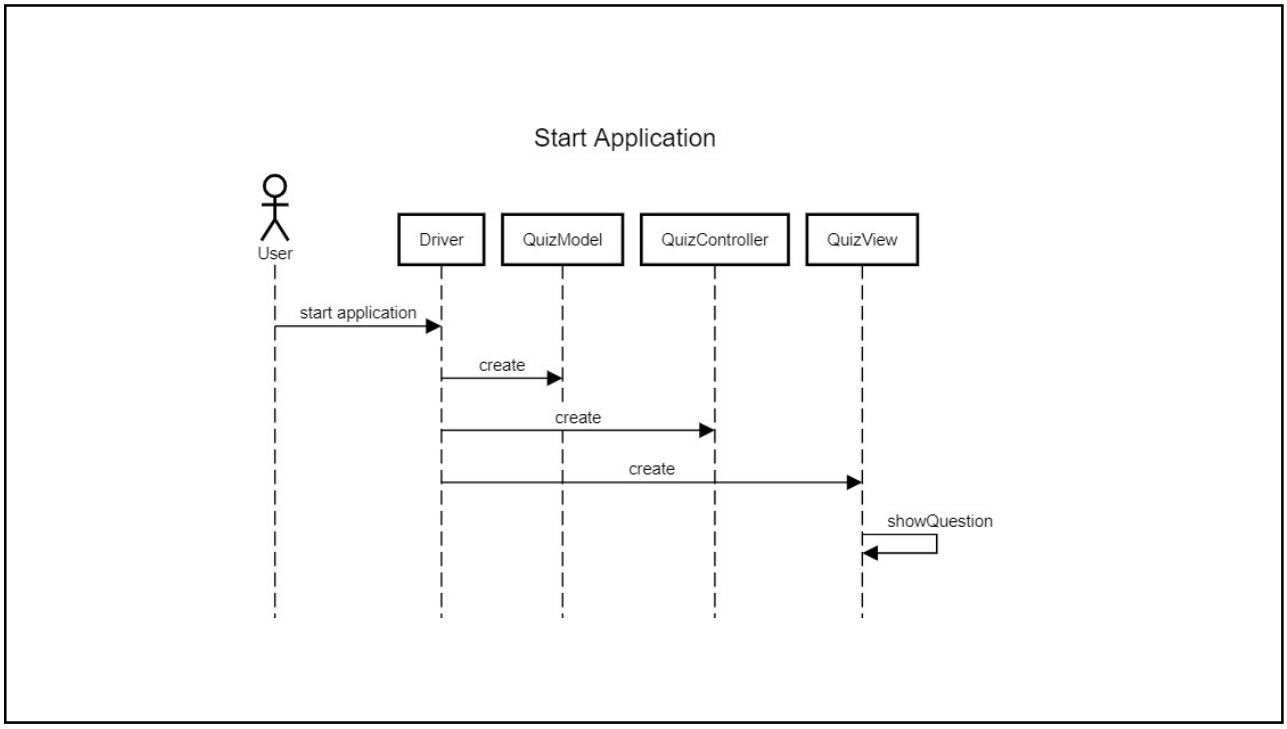


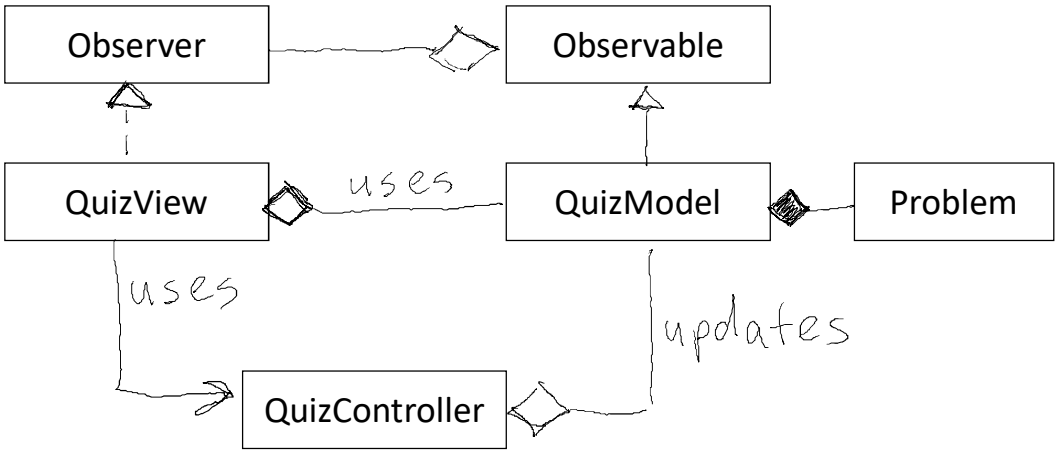
How is this different from our previous implementation of Submit Answer?



How is this different from our previous implementation of TryAgain?

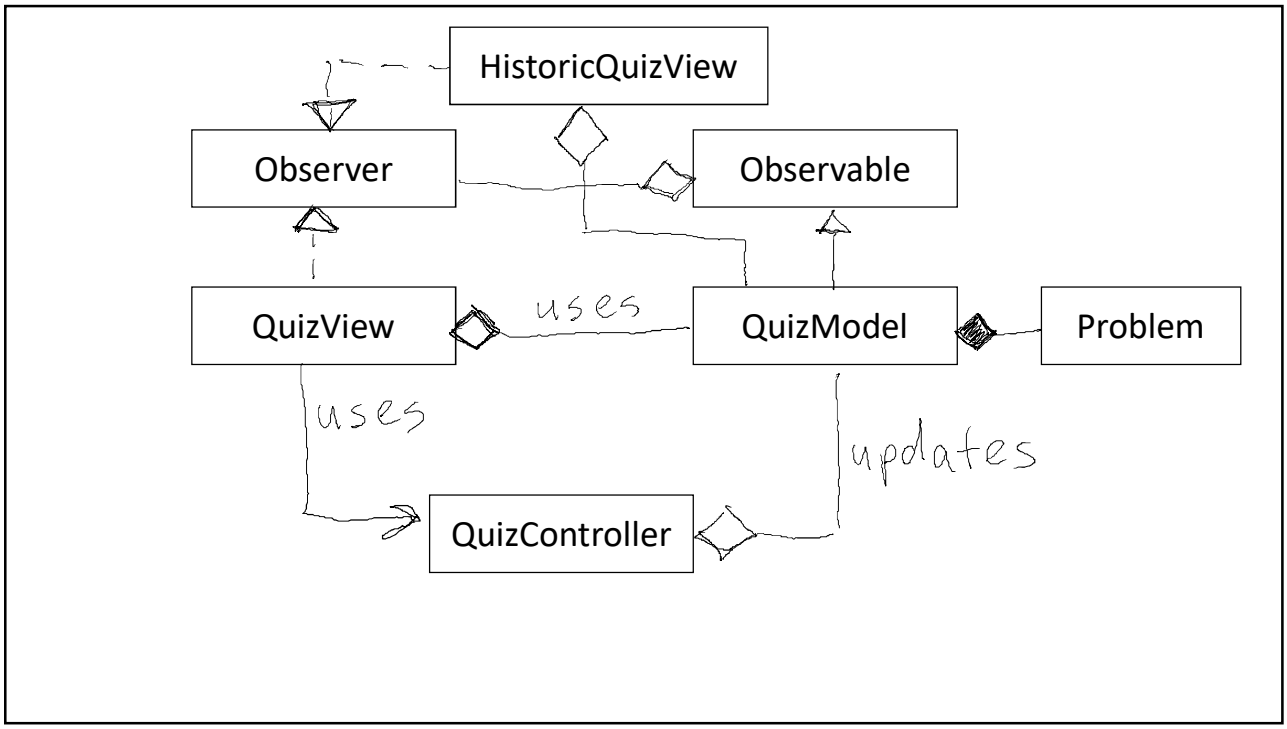

New Question
User
QuizView
NewQuestionListener
QuizController
QuizModel
Problem
Clicks New Question
actionPerformed
newQuestion()
update
set(int, int, bool)
changeState
update


Start Application
User
Driver
QuizModel
QuizController
QuizView
start application
create
create
create
showQuestion



Multiple Observers of the same Model

- Change: Update application to have a running history of all problems and solution attempts on a separate screen
- Solution:
  - Implement HistoricQuizView class as an Observer of the QuizModel
  - Add HistoricQuizView to the list of QuizModel's observers
  - Whenever QuizModel calls notifyObservers, all observers are notified



Updated Driver

```
public static void main(String []args)
{
   QuizController controller = new QuizController();
   QuizModel model = new QuizModel();
   QuizView view = new QuizView(model);
   view.addListeners(controller);
   HistoricQuizView historicView = new HistoricQuizView(model);
   model.addObserver(view);
   model.addObserver(historicView);
}
```

Lab 15

Add another observer to the application in the mathQuizObserver directory. This observer will keep calculate and display the user's current score:

- For each correct answer, the add 2 points
- For each wrong answer, subtract 1 point