

**Math City**

**Client: Dr. Irene Polycarpou**

**Team: Amanreet Bajwa, Nick Hansen, Levente Sipeki, and Joe Zeimen**

## **Abstract**

Math City is a game designed to help 5<sup>th</sup>-grade students learn the fundamentals of mathematics to improve their CSAP scores as well as learn about renewable resources. The Math City project for the client, Dr. Polycarpou, involved creating a wider variety of math questions for 5<sup>th</sup>-grade students, and an improved feedback system. The motivation for this project came from a lack of proficiency in Colorado's math scores on the CSAP. Only 63% of 5<sup>th</sup>-graders were proficient or above in mathematics on the CSAP[1]. Currently this project is focused on 5<sup>th</sup>-graders, but has the ability to expand. Using this system, 5<sup>th</sup>-grade students can improve and reinforce their basic math skills while learning in a fun environment.

There were three primary requirements for this project. First, the team needed to increase the variety of math problems that will be featured in the game to advance the user's math skills which required some changes in functionality. Second, enhancements needed to be made to the feedback that is provided when the user answers a question, not only providing a "correct" or "incorrect" indicator, but hints on how to solve the problems as well. Finally, any bugs that were residing in the code were either documented or fixed and bugs created during development were fixed.

To increase the variety of math problems we implemented a sophisticated algorithm based on the user's input and problem-solving history. To enhance the feedback system we added more visual hints as well as examples. These visual hints are given upon the user's failure to answer correctly, or by the request of the user. All program components are designed to be easily used and understood by fifth-grade students.

In the end all the objectives were completed. We refactored the problem system, added twelve new problem types, added GIFs to the feedback, and included a progress bar in the score frame.

## **Requirements Specifications**

### **Introduction**

Our client is Dr. Irene Polycarpou, a professor at Colorado School of Mines (CSM). The system that the client has provided is a game called Math City which is an ongoing graduate project at CSM. Math City is designed for students in the 5<sup>th</sup>-grade to practice math questions for the Colorado Student Assessment Program (CSAP) as well as learn about renewable energy sources. Over time the game will expand to include questions for grades K-12 but currently the focus is on 5<sup>th</sup>-grade questions. The motivation behind this program is to improve scores students are receiving on standardized tests like the CSAP. According to the Colorado Department of Education, only 63% of 5<sup>th</sup>-grade

students were proficient or above in mathematics on the CSAP. The idea of Math City is to reinforce the basic mathematical concepts required to succeed on standardized tests. Since every student is required to take the CSAP, this program could be very beneficial not only to the students but also to the educational system.

Math City allows users to create virtual cities that include houses, hospitals, police stations, and fire stations. Additionally, there are sources of power, such as coal plants and windmills, and industrial buildings that provide jobs. Power lines are required to connect power sources to buildings. The primary goal of the game is to maximize “happiness” indicators related to factors such as number of jobs, pollution, and services. The coal plants create pollution and decrease happiness whereas the windmills do not affect pollution. Everything in the city requires money to be built which, in turn, forces the students to answer CSAP practice questions in order to make more money to continue playing.

The goals for advancing the Math City project were prioritized on a functional basis. First, a wider variety of questions needed to be added to the question set from CSAP question examples. Second, more feedback needed to be given to the user when a question is answered incorrectly. Rather than just telling the user they answered incorrectly or correctly there is feedback that encourages the student to use the hints if the program finds that they are struggling.

## **Requirements**

The client asked us to focus on increasing the number and type of mathematics questions for a 5<sup>th</sup>-grade audience as well as add and improve the feedback the user receives when answering questions.

### **A. Functional requirements**

The product has these additional functionalities:

1. More questions and types of questions to the game. These questions are similar to what a student will see on a standardized exam. The user gets a reward of additional funds comparable to the difficulty of the question.
2. Improved corrective feedback on questions. If the user answers a question incorrectly he or she should see the right answer. If he or she answers the same type of question incorrectly again, the user should get an explanation of how the question can be completed correctly.
3. Fix and document any bugs we find in existing code.

Our first priority was to work on the questions and feedback in the game. The game is meant to teach children math in a fun game setting.

## **B. Non-Functional requirements**

1. Write code in Java.
2. Add and modify the code existing on the Google Code repositories to implement the new or improved features.
3. The game should be geared towards a 5<sup>th</sup>-grade audience.

The game is currently targeted at a 5<sup>th</sup>-grade audience, but we also had to keep in mind that we needed to make it easy to add questions so that other grade levels could be implemented.

## **C. Scope**

The most important aspect was to increase the number and variety of math problems. The problems we made were designed to resemble what a 5<sup>th</sup>-grader would see on a CSAP test based on released items from previous CSAP tests. This will ensure that a 5<sup>th</sup>-grader has familiarity when actually taking a standardized test like the CSAP. In broadening the scope of problems, the feedback requires more in depth explanations for difficult problems. We focused on adding those step-by-step explanations so that a 5th-grader could potentially learn from the examples without having to consult anyone for help. Creating this kind of feedback was our priority in this project.

## **D. Use Cases**

There are many different types of questions we can ask. Most of the questions will be very similar with minor differences in flow. This is a use case for a generic question.

### Question Use Case:

1. User clicks the "Get more money" button.
2. User is presented with a question. Difficulty is based on whether the user selects the "More" or "Less" money button. The "More" money corresponds to a higher level of difficulty while the "Less" money button corresponds to a lower difficulty.
3. User types or clicks an answer, the close button, hint, next or score button.
  - a. User clicks an answer the program displays whether or not the question was correct.
    - i. The answer is correct. Give user more money in the game.
    - ii. The answer is incorrect. Display the correct answer. If the user has missed this type of question before in the recent past suggest the user look at the hint of how the mathematical operation is performed.
    - iii. Enable the "Next" button to be pressed.

- b. The user clicks the "Next" button. The next question is displayed and we return to 2.
- c. The user clicks the "Score" button. The score dialog displays how many questions the user has gotten correct and incorrect and what types of problems he or she is struggling with as well as a progress bar. The score is tallied from when the game is first run to when it is closed. User must close the window to get back to the previous window.
- d. The user clicks the hint button. The program displays a new window with an animation describing how to solve the problem.
- e. The user clicks the close button. The window closes and game play resumes.

## System Design

### High-level Design

1. **Question Expansion** - The subsystem for questions was expanded to contain a wider variety of questions. The list of questions added can be seen on the attached chart. We looked at released items from CSAP as well as other educational websites to word our questions most appropriately for a 5<sup>th</sup>-grader. The order in which questions are given has also been revised to adjust to the difficulty level that the student is at.
2. **Feedback** Before we started, the user only knew if he or she answered the question correctly. We have changed the game so that they know what the correct answer was after they answer the question. It gives encouraging feedback based on how the student is doing. If the student is incorrectly answering questions it reminds them that there is a hint that they can look at.

### Design Detail

Question Expansion – We refactored the system that we inherited by generalizing it. Figure 7 illustrates the previous system design.

The manager class is responsible for getting a problem from a problem factory, and passing that problem to the appropriate GUI interface. For example, the manager asks the problem factory to generate a new problem based on the player's history. It then passes that question to the GUI component that is responsible for displaying a problem of that type. The manager then displays the GUI for the question.

The problem factory takes information from the player's history then references an enumerated type (`ProblemTypeEnum`) to create a new question. The `ProblemTypeEnum` contains the information needed to create a problem. It is a common place to store a list

of questions and information. The factory creates a problem based on this enumerated type. As you can see in the figures below, we have abstracted the current classes so they can be reused with any class for an answer.

Figure 4 shows our revised UML design. The classes whose parent is `Problem` are objects that represent a question and the classes whose parent is `QuestionGUI` are the graphical face of those problems. Previously the questions were hard coded to only take a fraction as the answer. This is obviously not desirable because other questions will not use fractions at all. To make it more abstract, we changed the class `Problem` to make it a generic class. This way we can choose anything to be the answer. We also made the problem factory an interface so that we can switch out factories easily without changing any code.

Note that the UML for the new design only has some of the problems and their graphical counterparts the `JPanels`. If a new problem needs a different graphical layout or different parameters to show for the question, new classes can be created by only implementing question text and the response area. The rest of the functionality is defined in the `QuestionGUI` class. For almost all problems, only the enumerated type and the create function in the problem factory need to be altered to add a new question. The only reason why there would need to be more coding than this is if a different layout for the GUI is needed. In this case the programmer would need to create a new subclass of `QuestionGUI`.

We have added a new main routine that is a teacher mode for displaying the questions. This way the teacher doesn't need to run the entire game to preview the questions. It also has a short description of what kind of problem the teacher is looking at in the title bar.

## **Results**

Our project included implementing new problem types and a better feedback system. We completed these tasks as well as refactored the question system so that it is easier to add different types of questions. We were able to meet the requirements by adding twelve new problem types, improving the feedback system and score system. We added the question types as well as improved the look of the questions by increasing font size and changing the color to make it more child friendly, as seen in Figure 3. The old question layout can be seen in Figure 2. The score and feedback systems were also improved by adding a progress bar and GIFs that show the user how to answer different problem types. The progression of our project included a lot of coding and research. We spent a few days researching different types of 5<sup>th</sup>-grade questions and how they were worded to portray the feel of answering CSAP questions in the game. About a week

was spent refactoring the question system and attaching our new GUI layouts to the system. Lastly, a couple days were spent adding in all the new problem types.

## **Conclusion**

### **Lessons Learned**

Throughout this project we implemented a wide variety of functionality. These advances included creating more problem types to better suit the CSAP curriculum on a 5<sup>th</sup>-grade level. Along with the functionality of this addition, we have added a progress meter to measure the level at which the user is at, as well as show the statistics of the success for answering questions. Advances have been made to make a more sophisticated algorithm for moving between the levels of problem difficulty.

We inherited a lot of code for this project. Sorting through this code was our first major problem to deal with. The code had been written by several different people before us and some of the classes were inadequately commented. The best way we found to understand this code was to meet with one of the people who worked on this project before us. It saved us a significant amount of work and let us understand why they chose the path they took. We were ready to implement the questions in a much more confusing way until we met with the person who worked on this project before us. Then we were able to understand and expand on his ideas.

Another frustrating aspect of this project was getting Subversion to merge different code the right way, when multiple people were working on the same part. This problem was only fixed by being very careful and making sure that everything works in the right manner after every commit and updating often. The second most troubling challenge that we encountered was refactoring the code. The issue with that was that the right changes had to be made in the existing code for it to work. This required us to learn the code very well and trust our changes, because there was no way to know if our changes were working until the very end. We also learned the ins-and-outs of the Eclipse IDE. It was very helpful to use the features it provides to help us refactor.

In the future this project requires improvement for everything to work completely in a classroom setting. The questions should be expanded even more to be able to encompass everything in the CSAP curriculum of the fifth grade level. Also, the tutorial feature should be implemented to provide a better game play experience for the user. With the tutorial feature, the user could quickly learn important parts of the game without having to learn the game with trial and error. Implementing these steps would provide a

smoother and more enjoyable game-play, making the 5<sup>th</sup>-grade learning experience a better process.

## **Adding new Questions**

To add most new questions there are three main places where code needs to be added. The first is adding a new type to the `ProblemTypeEnum`. The `ProblemTypeEnum` is an enumerated type that holds information for the factory and other classes to make the problem. The new problem type contains information as to what type of problem it is, the difficulty, where the image is for displaying a hint, and other useful information.

In the problem factory a new function needs to be added to hold the logic for creating the problem text and the correct answer. If it is a multiple choice question, it also needs to generate some wrong answers.

After adding the function to make the question, the factory needs to be able to call the function. This is done in the `create` function of the problem factory. It uses a switch statement based on the enumerated type and then calls the `make` function.

For example, to make a new problem that quizzes the student on finding a least common demoninator, we start with the enumerated type. First a new type needs to be created, we will call it `LCD`. In the enumerated type we will set the parameters to tell it that it is a multiple choice problem with difficulty 3. It also is expecting an integer for the answer and the location of the animated GIF we created if the user needs a hint.

In the problem factory a new function is created called `makeLCD` which takes in a `MCPProblem<Integer>` as a parameter. This function picks two random fractions then finds the least common denominator. It sets that as the question's answer with a setter function. We also can set it as one of the choices by adding it to the array list in the question with the function `getChoices().add(answer)`. Dummy answers can be added in the same way, they should look like plausible answers. Now just set the problem text with another setter function taking a string and shuffle the answers with another function in the question class.

Lastly the `create` function needs to be updated to know that there is a `makeLCD()` function. This just involves following the way that other `make` functions are called. The rest of the program will do the work for displaying the problem and checking the answers.



## Glossary

Happiness indicator - The status bars on the ToolBar window in the game. This includes pollution, police, fire, health, and buildings.



**Figure A.**

CSAP - The Colorado Student Assessment Program.

## References

[1] "2005-2009 Overall CSAP, Lectura, and Escritura Results." Colorado Department of Education.

[http://www.cde.state.co.us/cdeassess/documents/pressreleases/changing\\_conversations/2009\\_CSAP\\_TablesPRelease.pdf](http://www.cde.state.co.us/cdeassess/documents/pressreleases/changing_conversations/2009_CSAP_TablesPRelease.pdf)

[2] "Colorado Student Assessment Program (CSAP) Released Items"

[http://www.cde.state.co.us/cdeassess/released\\_items.html](http://www.cde.state.co.us/cdeassess/released_items.html)

[3] "IXL - 5<sup>th</sup> Grade Math Practice"

<http://www.ixl.com/math/grade/fifth/>

[4] "AAA Math - Fifth Grade"

<http://www.aaamath.com/grade5.html#topic1>


[5] "Everyday Mathematics and NCTM curriculum Focal Points"

[https://www.wrightgroup.com/download/em/EM3\\_Focal\\_Points\\_Brochure.pdf](https://www.wrightgroup.com/download/em/EM3_Focal_Points_Brochure.pdf)


Figure 1. More or less money

Question

# Choose an Option:



Less Money



More Money

Figure 2. The old question GUI

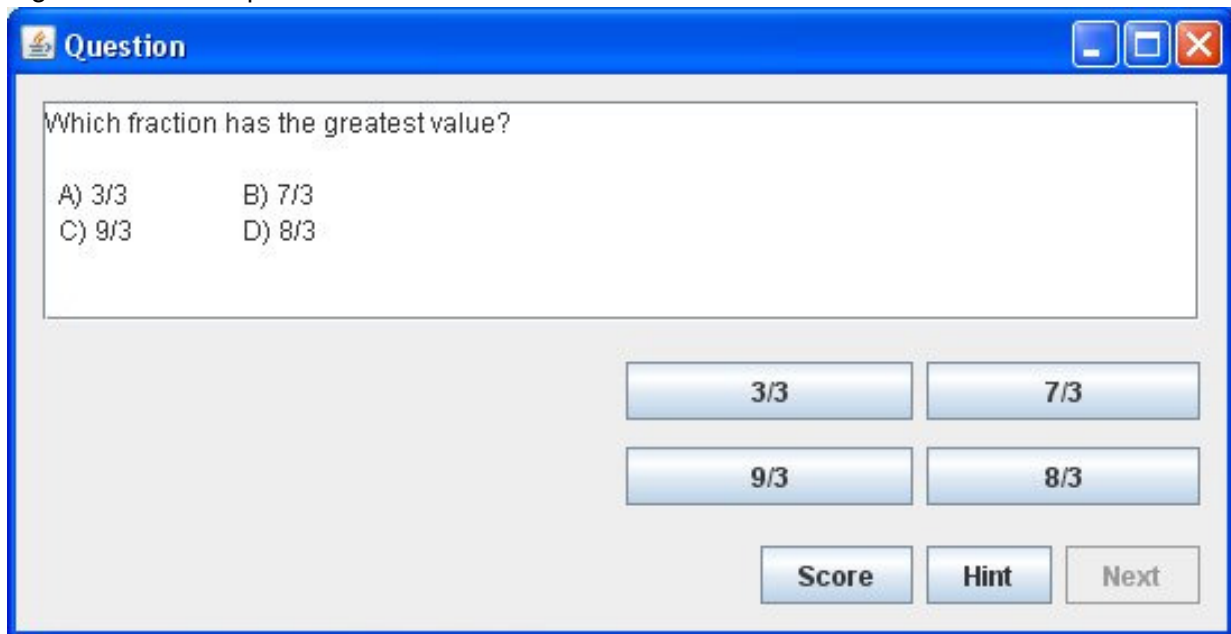


Figure 3. The new question GUI

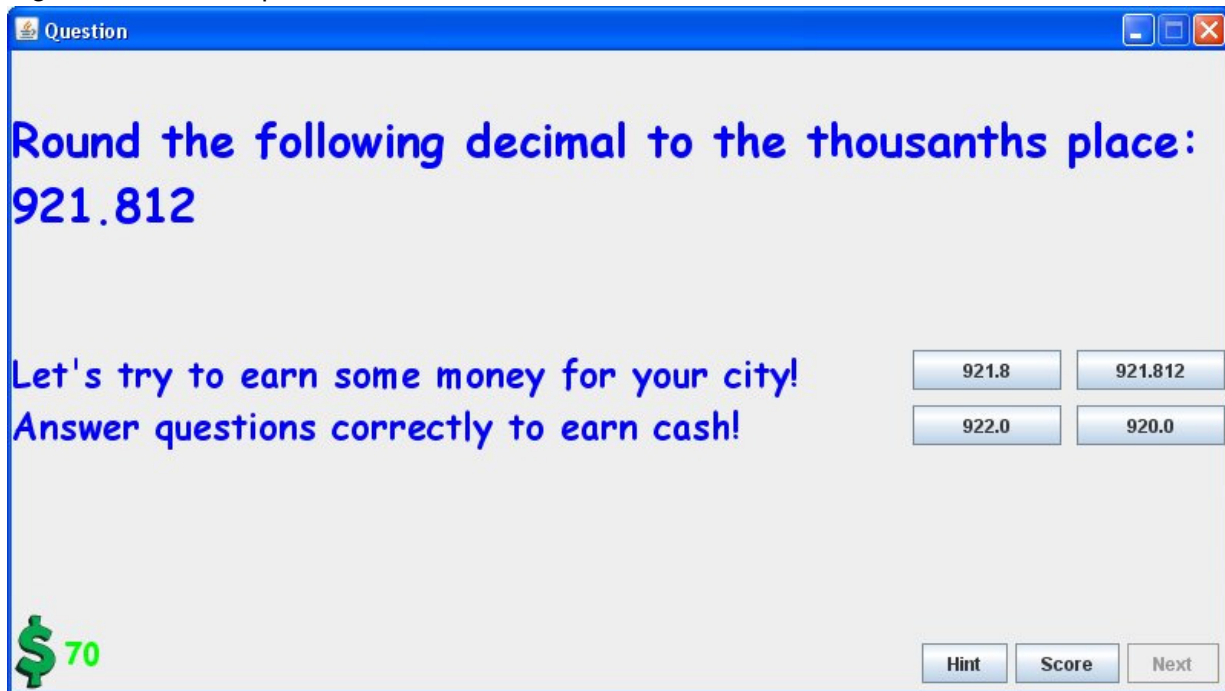


Figure 4. UML for Questions

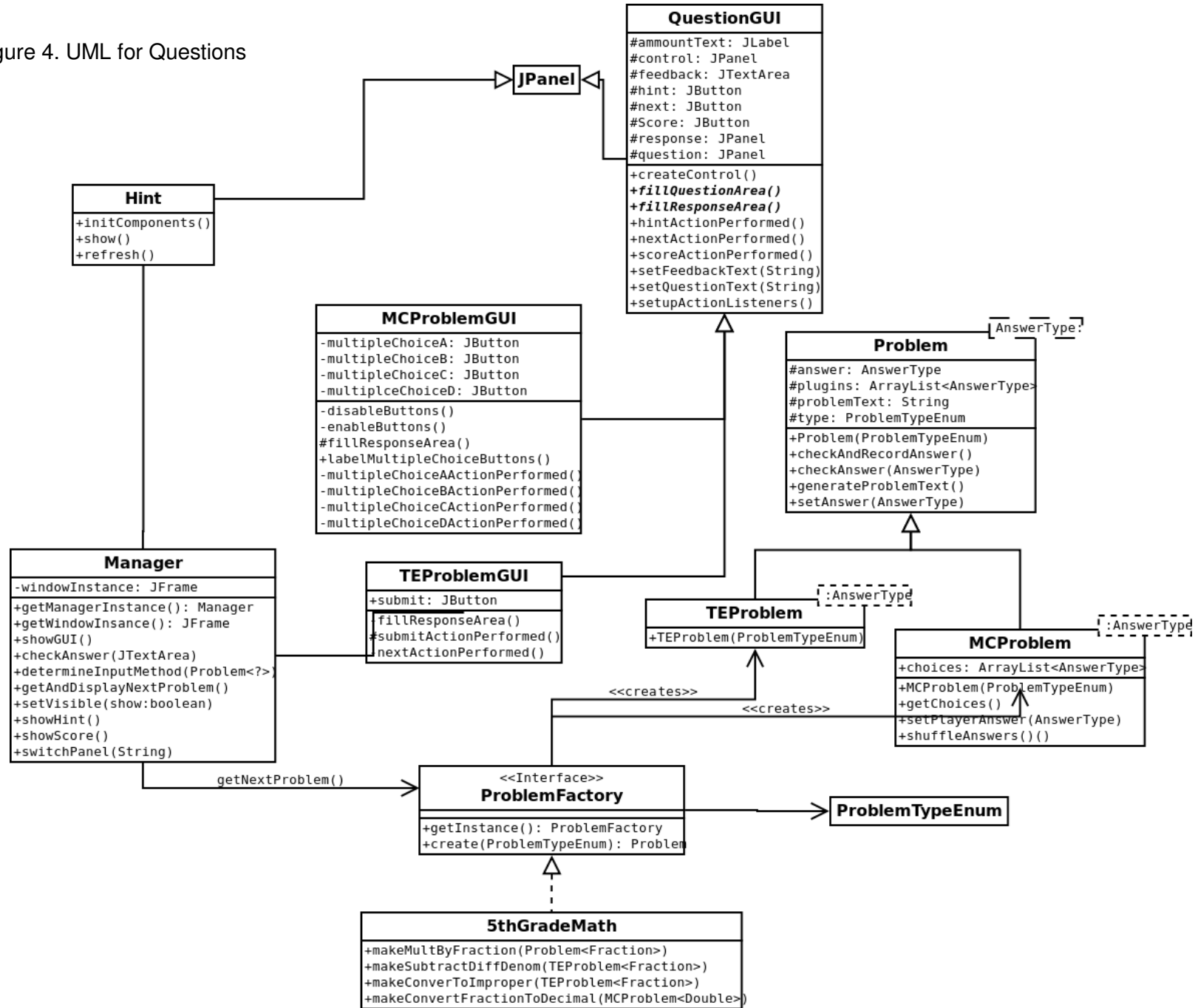


Figure 5. UML of the graphics and buildings.

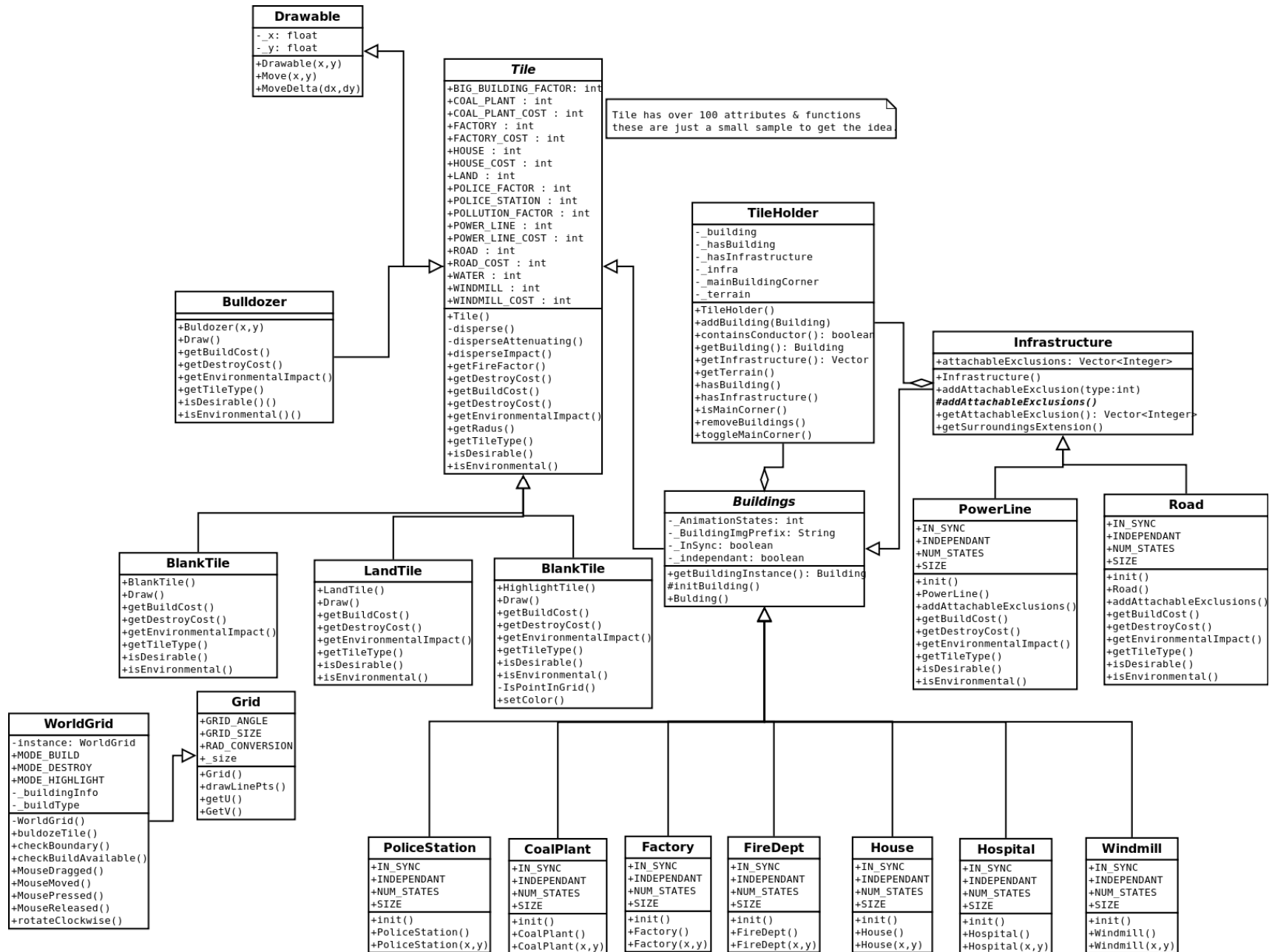


Figure 6. UML of control

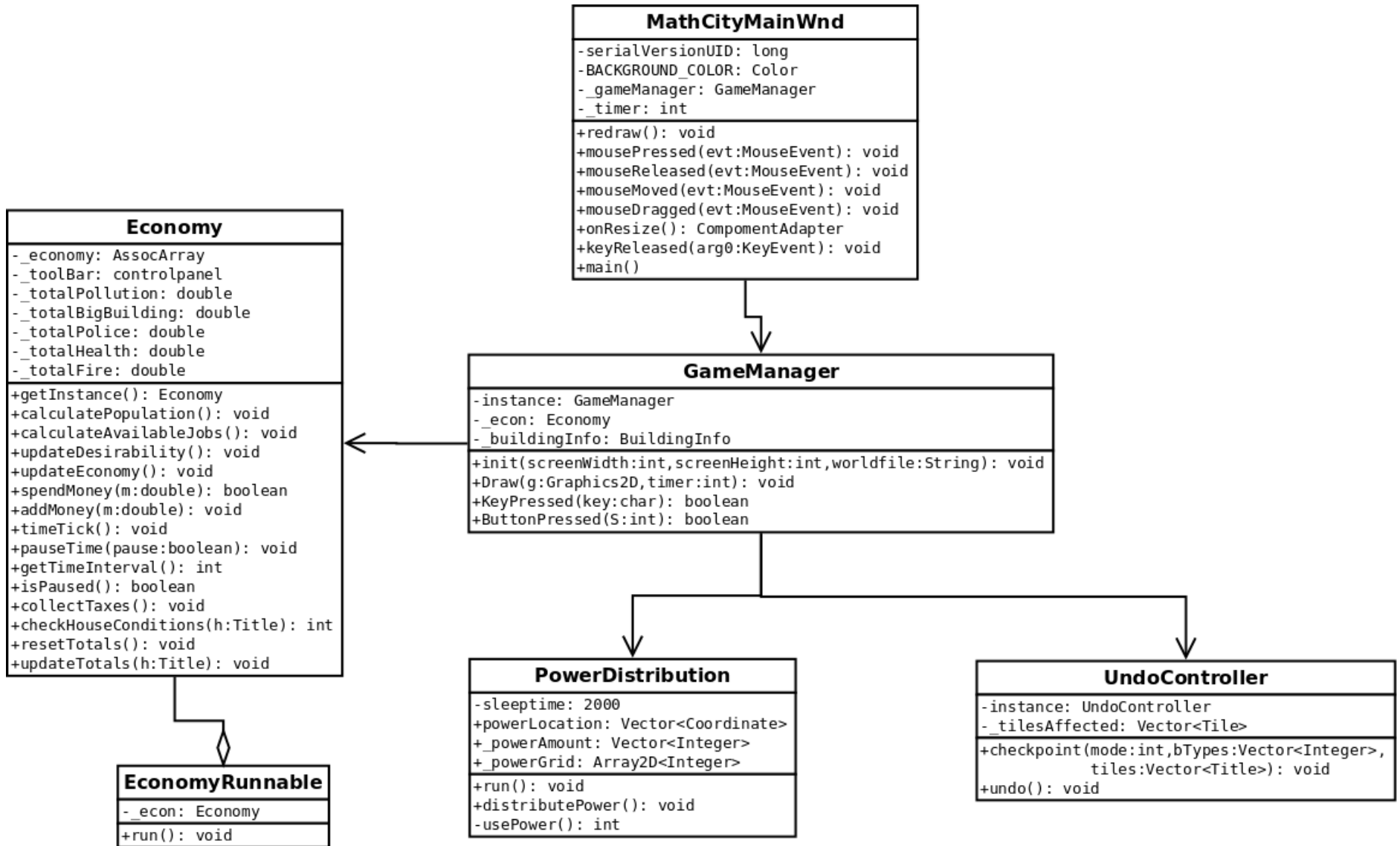
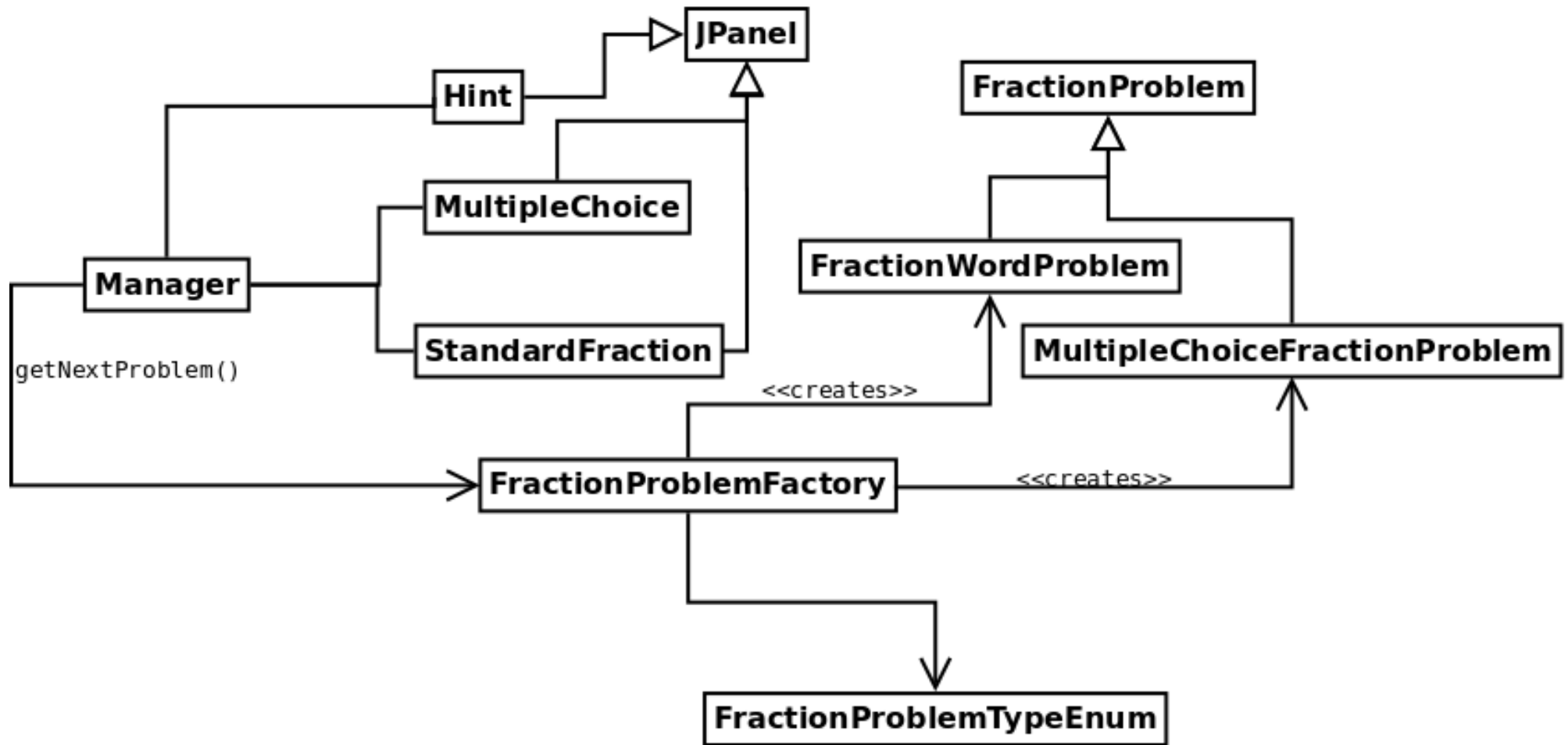


Figure 7. Previous Question System





Problem Specifications Chart

<b>Problem Type</b>	<b>Wording</b>	<b>Answer Type</b>	<b>Input/Multiple Choice</b>	<b>Difficulty level/ value</b>	<b>Notes on creating question numbers</b>
Reduce to Simplest form	Reduce the following fraction to the simplest form : $\frac{5}{25}$	Fraction	Input	3/\$30	Question number is created from set answers ( $\frac{1}{2}$ , $\frac{1}{3}$ , $\frac{1}{4}$ , $\frac{1}{5}$ , $\frac{3}{4}$ , $\frac{2}{5}$ ) then multiplied by a random number
Equivalent Fraction	Choose the equivalent fraction to $\frac{1}{4}$	Fraction	Multiple Choice	1/\$10	Answers are generated by random fractions except for the answer when a random number is multiplied by the question
Convert to improper fraction	Convert the following mixed fraction to an improper fraction $1 \frac{1}{5}$	Fraction	Input	2/\$20	Random number generation for the question, answer is calculated from question
Least common denominator	Find the least common denominator for the two fractions: $\frac{1}{4}$ and $\frac{1}{3}$	Number (only looking for denominator)	Input	3/\$30	Random number generation for the question on a level of (1-10). Answer is calculated from question. Random number generation is from 10-50
Compare fractions and mixed numbers	Find the equivalent to the following mixed number: $3 \frac{3}{4}$	Fraction	Multiple Choice	5/\$50	Random number generation 1-10 for mixed number and calculated form question. Then random number generation from for numerator only. Denominator pulled from question.
Round mixed numbers	Round the following mixed number to the nearest whole number $2 \frac{3}{4}$	Whole number	input	4/\$40	Random number generation 1-10 for mixed numbers and fraction number generation is 1-20. input calculated form generated question
Adding different denominators	$\frac{4}{9} + \frac{3}{7}$	Fraction	input	9/\$90	Random number generation of 1-10 on the fractions. Answer calculated.
Rounding decimals	Round the following decimal number the nearest whole number: 1.332	Whole number	Multiple choice	7/\$70	Random number generation 1-50 for all individual numbers on the question. Answer calculated. Fake answers include one answer rounded the wrong way and two decimal numbers with close answers
Convert fractions to decimals	Convert the following fraction to a decimal $\frac{3}{4}$	Decimal number	Multiple choice	8/ \$80	Random number generation 1-10 on question fraction. Answers calculated and random number generation with same whole number answer as answer (i.e. Answer: 2.5 fake answers: 2.4 2.75 )
Place values in decimal numbers	What digit in the tenth place 123.456	Whole number	Multiple Choice	6/ \$60	Random decimal number generation. Answer calculated from question. Fake answers include other individual numbers in question
Subtracting unlike denominators	$\frac{4}{9} - \frac{1}{4}$	Fraction	Multiple Choice	10/\$100	Same process as adding unlike denominators