Match Witch and Colorful kingdom

Team7

201221834 Minju Lee

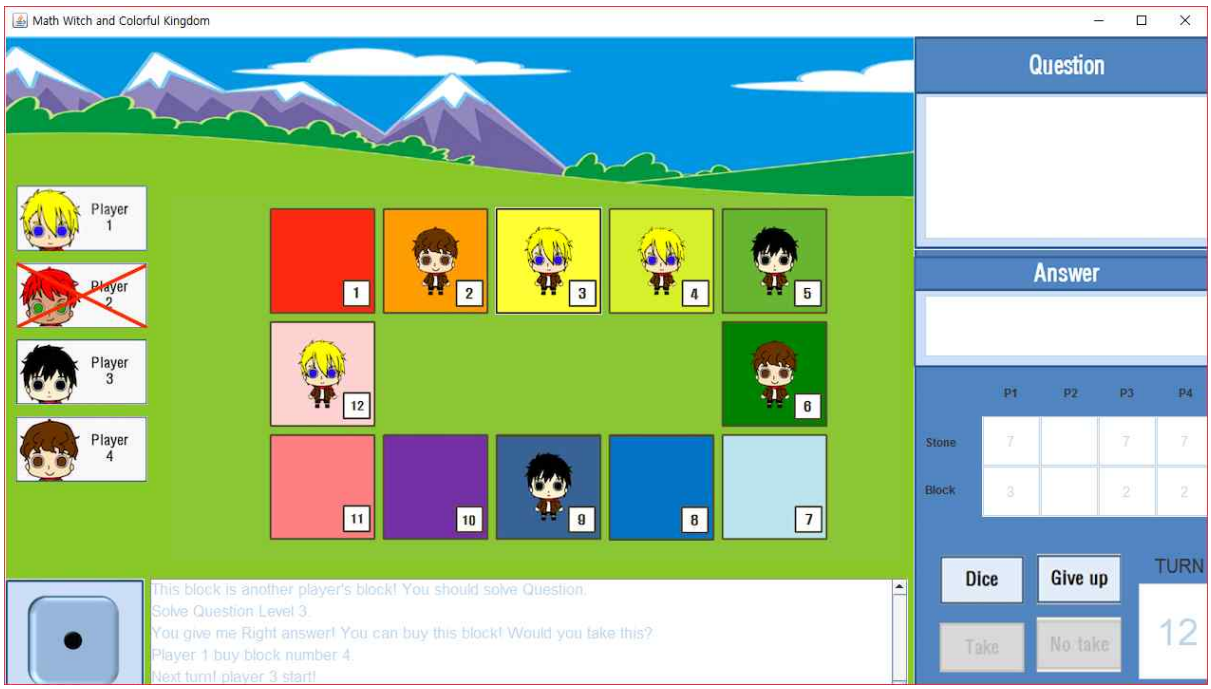
201220987 Donghun Kang

201221764 Jungyong Choi

201323152 Seonggwon Son

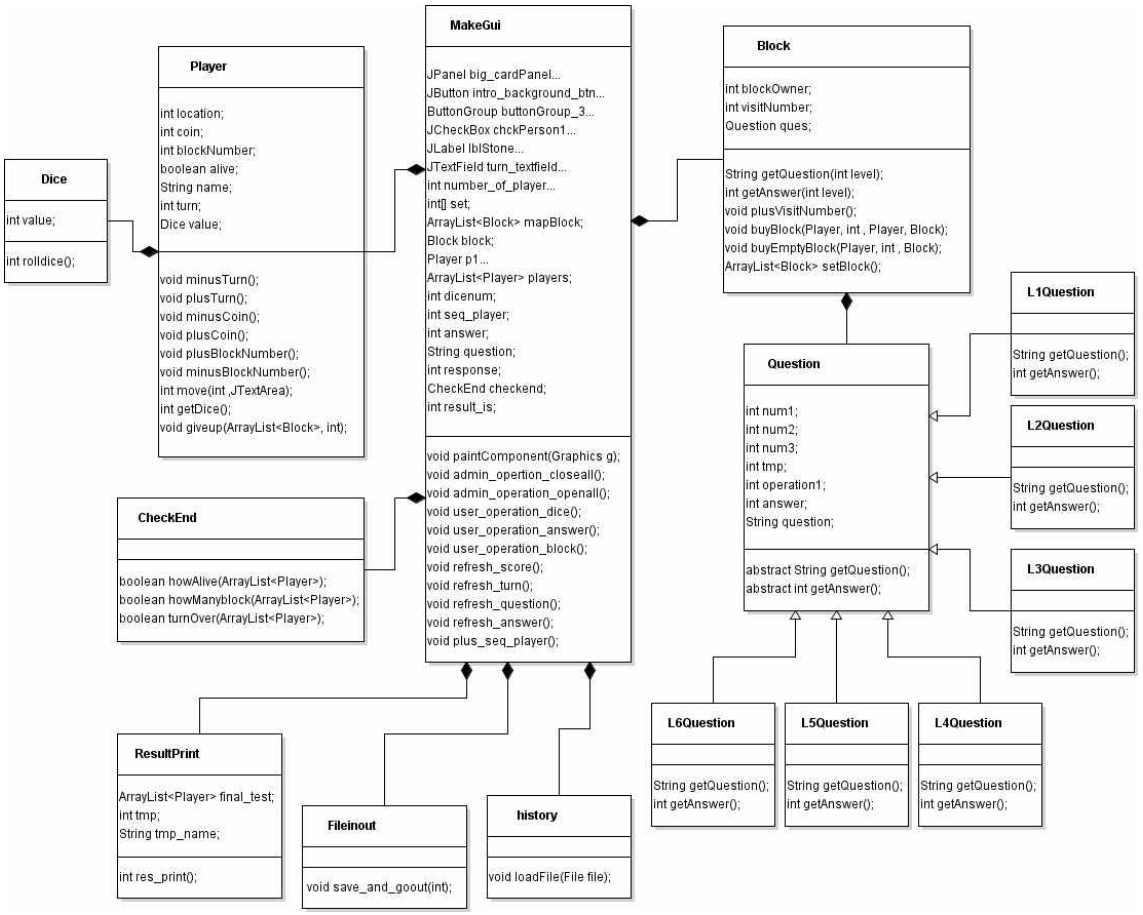
- The purpose and features of the application

This application is for lower grade elementary school student.
Purpose of it is for progressing their mathematical ability.
The features of application are that it is based on monopoly game.
And it has story, so students can be interested in this application by the story, but aslo students can play this game and study math with their friends up to 4 players.



- System architecture

This program runs base on MakeGui class. This class has Player, CheckEnd, ResultPrint, Rileinout, History and Block. Player has Dice class. Block class has Question class. Question class inherits from L1Question to L6Question classes.



- Explain important classes and methods

•MakeGui
MakeGui is a class that processes class and organizes frame. The whole program executes through this class.

•Startgame button
The startgame button produces the things you need for the game. When you press this button, the players and blocks will be generated for the game to start. And then the game tells the first player to start.

•Dice button
The dice button moves the player to the specific block and the block asks the player a math problem.

•Giveup button
The Giveup button is a button that allows player to give up the game. When the player presses this button the player gives up the game and gives back the blocks they have. When there is only one player left or the turn is over then you get the game score. Neither of these, then the next player will role the dice.

•Answer text field
Answer text field is where you get the answers from the user. The player should insert the answer as an integer. When the player gets right he or she gets to buy the block. And when the player is wrong, will lose one stone. When you don't have any stone left the player dies.

•Take button
Take button allows the player to buy the block. Player can buy a block using one stone. When there is a owner for every block or when the turn is over the result will come out. When Neither, then next player will role the dice.

•4 Game progress methods
admin_operation_openall
user_operation_dice
user_operation_answer
user_operation_block
admin_operation_closeall
are the methods that restrict game screen's element up to the progress of game. If users have to choose Dice or giveup button, methods are vitalize only Dice and giveup buttons. And if users have to write the answer, answer textfield is only vitalized. And if users have to choose buy block or not, Take and noTake buttons are vitalized.

•Block class
Block class do everything about blocks. This class use setBlock, it makes blocks and use buyBlock, and buyEmptyBlock, it decides the owner of block. And it has visit number of each block. depend on it, Question level is decided, next call getQuestion method that is for showing qeustion.

•Dice class
Dice class is make a number how move players. Rolldice method makes a random number and it is dice number.

•Fileinout class
Fileinout class is save the Result of Game into text file. Using Save_and_gogout method, It saves Play time and Result into text file.

•CheckEnd class
CheckEnd class checks when to end the game. There is a three way to end the game. The three way is When there is one player left, or when there is an owner for every block, or when the turn is over. How alive method calculates the player alive on the game. How many block method calculates blocks that has an owner. turnover method calculates the turns.

•Histoy class
History class shows the result of game by using saved textfile and gui. loadFile method opens text file and print it into screen. If press History reset button, all of game histories will be cleared.

•Resultprint class
Resultprint class shows the winner. res_print method tells who won the game.

•Player class
This class has the information of the player's location, number of stones, number of block that player has, life and death, name, the information about the turns that player has. It can also make the player move. Move method will move the player as much as the dice rolled.
Giveup method can make the player dead. Dead method returns the block of the dead player. getDice method call rolldice method in dice class.

•Question class
Question class is an abstract class that has a convention for its subclasses.

•L1~L6Question class
These classes are extend from Question class. These make question from level1 to level 6 and make a correct answer. getQuestion method make question and getAnswer method calculate the answer.

-Used OOP concepts

•Polymorphism

When make L1~L6Question objects, Used Question class as type. For example, Question class is upper class of L1Question. So we can use L1Question object's type as Question.

```
private Question ques;

public String getQuestion(int level){
     if(level==1){
          ques = new L1Question();
          return ques.getQuestion();
     }
     else if(level==2){
          ques = new L2Question();
          return ques.getQuestion();
     }else if(level==3){
          ques = new L3Question();
          return ques.getQuestion();
     }else if (level==4){
          ques = new L4Question();
          return ques.getQuestion();
     }else if(level==5){
          ques = new L5Question();
          return ques.getQuestion();
     }else{
          ques = new L6Question();
          return ques.getQuestion();
     }

}
```

•abstract classes

It uses question class as abstract class. So we can't make Question object. But we can use Question's method to inherit classes from L1Question to L6Question.

```
package kr.ac.ajou.group_Seven.Question;


abstract public class Question {

     int num1;
     int num2;
     int num3;
     int tmp;
     int operation1;
     int answer;
     String question;

     abstract public String getQuestion();
     abstract public int getAnswer();
}
```

•encapsulation

Except the question class and the subclasses of question class This make the whole class instance variable to private and it only allows method to change the value.

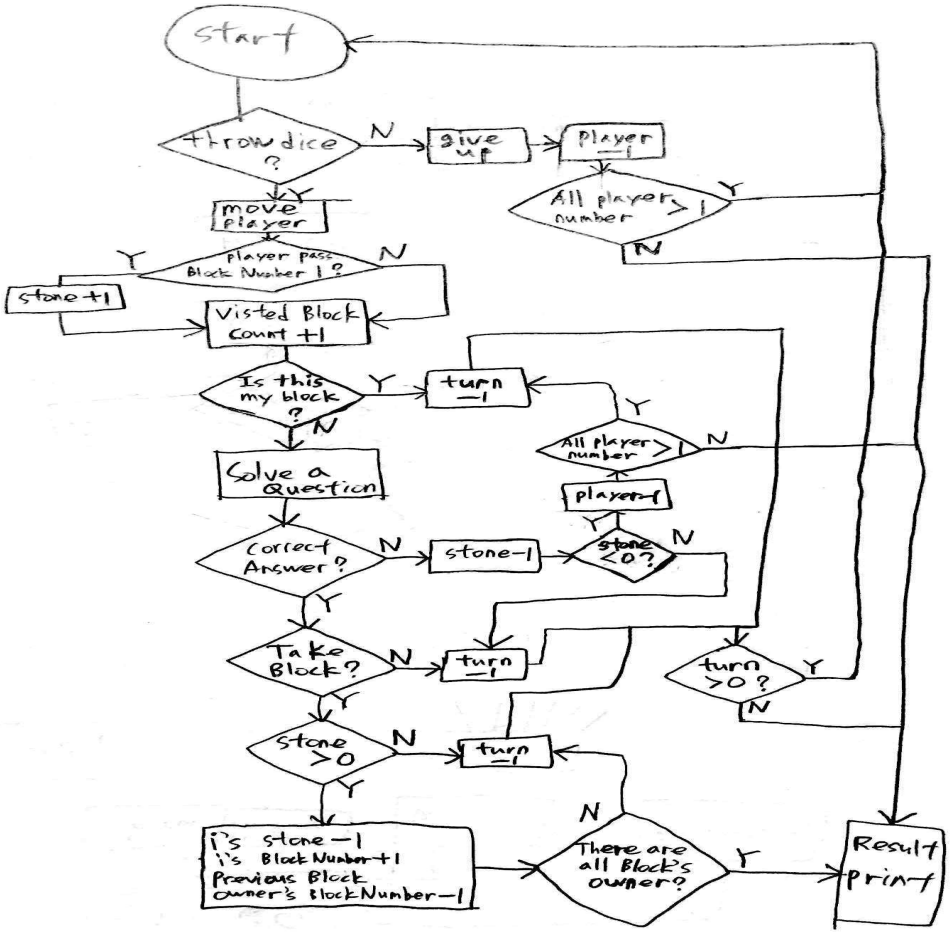
```
package kr.ac.ajou.group_Seven.play;
import java.util.ArrayList;

public class Player {
    private int location; // player's current location
    private int coin; // player's current cloin
    private int blockNumber; // player's current own block number
    private boolean alive; // player's current state
    private String name; // player's name
    private int turn; // player's have turn.

    private Dice value;

    public Player(String name, int location, int coin, int blockNumber, int turn, boolean alive) {
        this.name = name;
        this.location = location;
        this.coin = coin;
        this.blockNumber = blockNumber;
        this.turn = turn;
        this.alive = alive;
    }

    public int getTurn() {
        return turn;
    }

    public void setTurn(int turn) {
        this.turn = turn;
    }

    public void minusTurn() {
        this.turn -= 1;
    }

    public void plusTurn() {
        this.turn += 1;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getLocation() {
        return location;
    }

    public void setLocation(int location) {
        this.location = location;
    }
```

•try&catch

We use try and catch to find an error if we enter String not integer. If we enter integer in answer text field, NumberFormatException is catched. And stateFlow_text area write "You have to give me Integer! Write again!"

```
public void actionPerformed(ActionEvent e) {


    try{
        response = Integer.parseInt(answer_text.getText());

        admin_operation_openall();

        if(response != answer){
            stateFlow_text.append("You give me Wrong answer! Real answer is " + answer + ".\n");
            players.get(seq_player).minusCoin();
            stateFlow_text.append("Your Stone minus 1.\n");
            if(players.get(seq_player).getCoin()<0)
            {
                players.get(seq_player).giveup(mapBlock, seq_player);
                stateFlow_text.append("Player " + (seq_player+1) + " is dead.\n");
                if(checkend.howAlive(players)){
                    cl.next(big_cardPanel);
                    repaint();
                }
            }
            players.get(seq_player).minusTurn();
            if(checkend.turnOver(players)){
                cl.next(big_cardPanel);
                repaint();
            }
            refresh_question();
            refresh_answer();
            refresh_score();
            refresh_turn();
            repaint();
            plus_seq_player();
            stateFlow_text.append("Next turn! player " + (seq_player+1) + " start!\n");
            admin_opertion_closeall();
            user_operation_dice();

        }else{
            stateFlow_text.append("You give me Right answer! You can buy this block! Would you take this?\n");
            admin_opertion_closeall();
            user_operation_block();
        }


    }catch(NumberFormatException ex){
        admin_operation_openall();
        stateFlow_text.append("You have to give me Integer! Write again!\n");
        answer_text.setText("");
        admin_opertion_closeall();
        user_operation_answer();
    }
```

-Running instructions



1) Start game with pushing Start button.
2) If push history button, game history screen is shown.
3) Game introduction and prologue is printed.
4) Game setting screen is printed.
5) Using checkbox, determine player, coin, and turn number.
6) Game strart with pushing Startgame button.
7) On someone's turn, dice button and giveup button is vitalized.
8) If player push dice button, dice is rolled and player move, and have to solve problem shown on screen.
9) If player input correct answer, Take or no button are vitalized, else player's coin is -1 and turn over.
10) If player buy block, players's coin is -1, and player's location block show player.
11) next player's turn start. (arrow goes to Start Box, next player start)
12) In end, The winner is the man who has most blocks.

-Declaration of used libraries and borrowed code, if you use anything that is not 100% yours

***Code**

1)CardLayoutDemo of eclass 12_GUI_2-code.big_cardPanel.add(menubar, "zero")
•big_cardPanel.add(intro, "first");
•big_cardPanel.add(prologue, "second");
•big_cardPanel.add(game_rule1, "third");
•big_cardPanel.add(game_rule2, "fourth");
•big_cardPanel.add(game_setting, "fifth");
•big_cardPanel.add(game_playing, "sixth");
•big_cardPanel.add(result_panel, "seventh");
•CardLayout cl = (CardLayout) (big_cardPanel.getLayout());
•cl.next(cardPanel);
•cl.next(big_cardPanel);

2)http://stackoverflow.com/questions/9020522/how-to-show-automatically-caret-at-the-end-of-new-added-text-to-the-textarea-in
stateFlow_text.setCaretPosition(stateFlow_text.getDocument().getLength());

***Picture**

1) background.jpg
https://pixabay.com/ko/%EB%B0%B0%EA%B2%BD-%EC%9E%94%EB%94%94-%EB%85%B9%EC%83%89-%EC%96%B8%EB%8D%95-%EC%B4%88%EC%9B%90-%EC%82%B0-%EC%9E%90%EC%97%B0-%EB%B4%89%EC%9A%B0%EB%A6%AC-%ED%95%98%EB%8A%98-%EB%88%88-985947/

2) game characters
http://www.icongenerators.net/chibidot_low.html

- Team member roles

Donghun Kang: Synthesize claasses and make MakeGui class.
Seonggwon Son: Make Block, Question, and L1~L6Question classes.
Minju Lee: Make Fileinout, history classes and all image.
Jungyong Choi: Make Plyaer, Dice, CheckEnd and Resultprint classes.