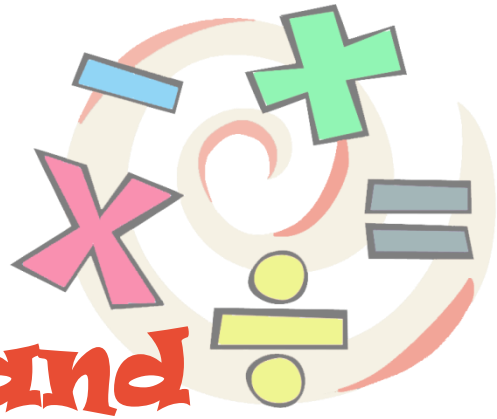


# Math Witch and Colorful Kingdom


Team 7

201220987 DongHun Kang

201221764 Jeongyong Choi

201221834 Minju Lee

201323152 Seonggwon Son

# Contents

- ✓ Key features of the application
- ✓ Team member roles
- ✓ Architectural design of the system
- ✓ Object-oriented programming concepts
- ✓ Difficult in the project work

# Key features of the application

This application is for lower grade elementary school student

- ✓ Purpose of it is for progressing their mathematical ability.
- ✓ This application are based monopoly game.
- ✓ students can be interested in this application by the story.
- ✓ students can play this game and study math with their friends.

Math Witch and Colorful Kingdom

# Team member roles

The roles of which the team members

- ✓ DongHun :

  Synthesize claasses and make MakeGui class.
- ✓ Jeongyong :

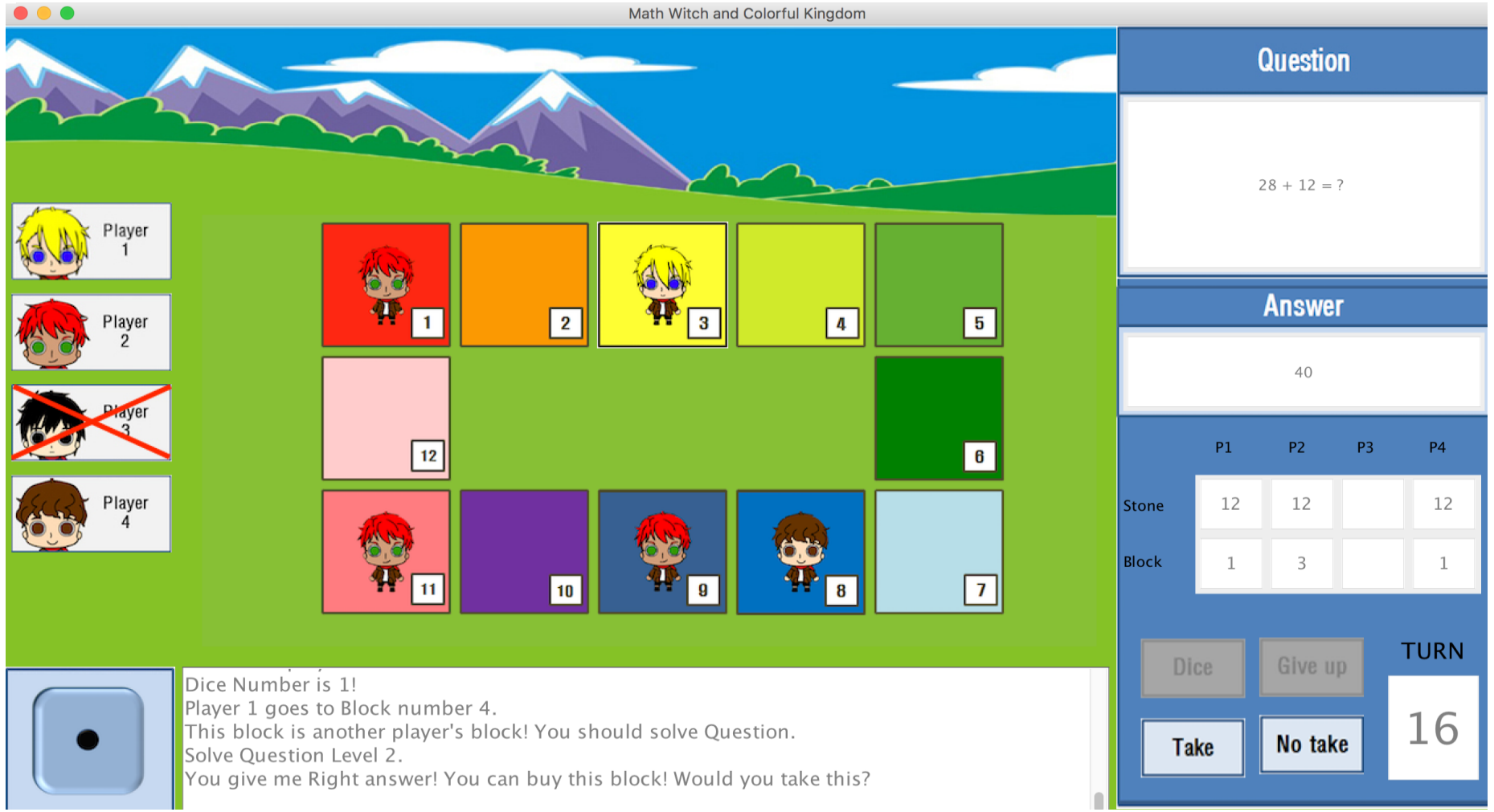
  Make Player, Dice, CheckEnd and Resultprint classes.
- ✓ Minju :

  Make Fileinout, history classes and all image.
- ✓ Seonggwon :

  Make Block, Question and L1~6 Question classes.

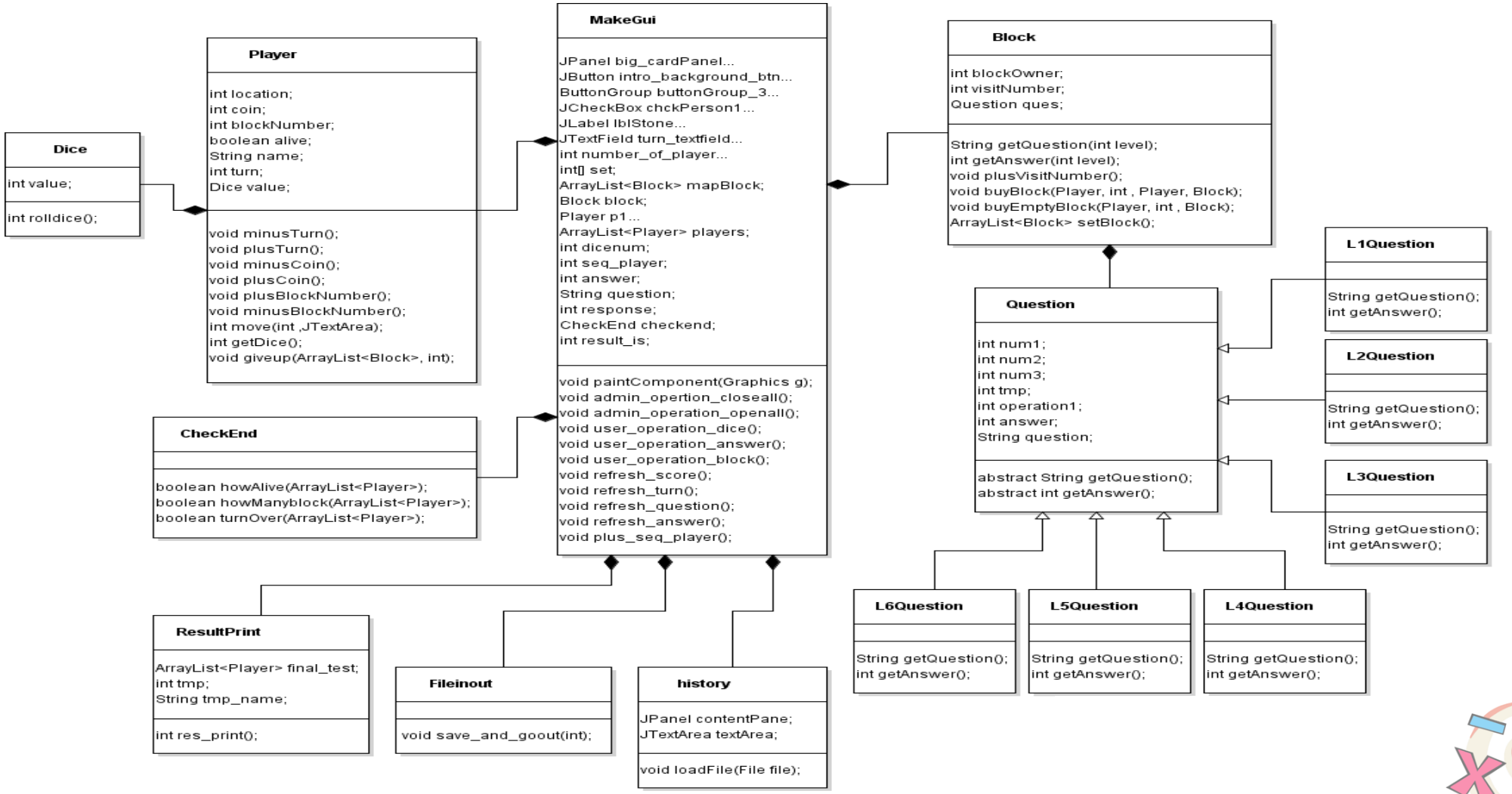
Math Witch and Colorful Kingdom

# Architectural design of the system




Math Witch and Colorful Kingdom

# Architectural design of the system




Math Witch and Colorful Kingdom

# Architectural design of the system

| MakeGui |
| --- |
| JPanel big_cardPanel...<br>JButton intro_background_btn...<br>ButtonGroup buttonGroup_3...<br>JCheckBox chckPerson1...<br>JLabel lblStone...<br>JTextField turn_textfield...<br>int number_of_player...<br>int[] set;<br>ArrayList<Block> mapBlock;<br>Block block;<br>Player p1...<br>ArrayList<Player> players;<br>int dicenum;<br>int seq_player;<br>int answer;<br>String question;<br>int response;<br>CheckEnd checkend;<br>int result_is; |
| void paintComponent(Graphics g);<br>void admin_opertion_closeall();<br>void admin_operation_openall();<br>void user_operation_dice();<br>void user_operation_answer();<br>void user_operation_block();<br>void refresh_score();<br>void refresh_turn();<br>void refresh_question();<br>void refresh_answer();<br>void plus_seq_player(); |

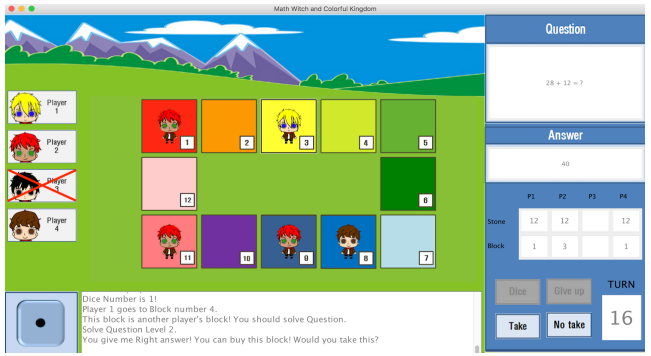
## MakeGui

- game processes and organizes frame.
  The whole program executes through this class.

## 4 Button

- This game is played with four buttons.

- Dice button : player move to block
- Give up buttons : player remove
- Take buttons : player occupy the block
- No Take buttons : player no occupy the block




Math Witch and Colorful Kingdom

# Architectural design of the system

Ex.

```
notake_btn = new JButton("");
notake_btn.addActionListener(new ActionListener(){
    @Override
    public void actionPerformed(ActionEvent e){
        stateFlow_text.append("You were not buy Block.\n");

        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();
    }
});
Image notake_btn_img = new ImageIcon(this.getClass().getResource("/notake_btn.png")).getImage();
notake_btn.setIcon(new ImageIcon(notake_btn_img));
```

Math Witch and Colorful Kingdom

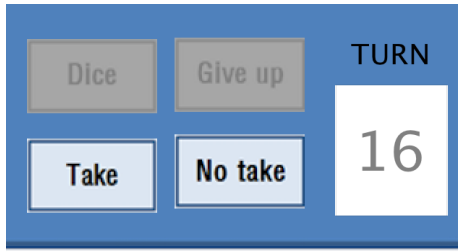
# Architectural design of the system

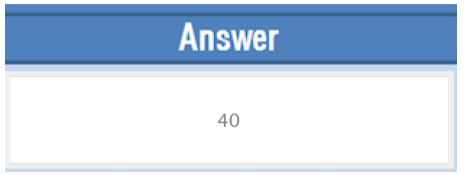
**MakeGui**

```
JPanel big_cardPanel...
JButton intro_background_btn...
ButtonGroup buttonGroup_3...
JCheckBox chckPerson1...
JLabel lblStone...
JTextField turn_textfield...
int number_of_player...
int[] set;
ArrayList<Block> mapBlock;
Block block;
Player p1...
ArrayList<Player> players;
int dicenum;
int seq_player;
int answer;
String question;
int response;
CheckEnd checkend;
int result_is;

void paintComponent(Graphics g);
void admin_opertion_closeall();
void admin_operation_openall();
void user_operation_dice();
void user_operation_answer();
void user_operation_block();
void refresh_score();
void refresh_turn();
void refresh_question();
void refresh_answer();
void plus_seq_player();
```

## Game progress 4 methods

- the methods that restrict game screen's element according to the progress of game.
- If users have to choose Dice or giveup button, only Dice and giveup buttons are vitalized.
- If users have to write the answer, answer textfield is only vitalized.
- If users have to choose buy block or not, Take and noTake buttons are vitalized.




Math Witch and Colorful Kingdom

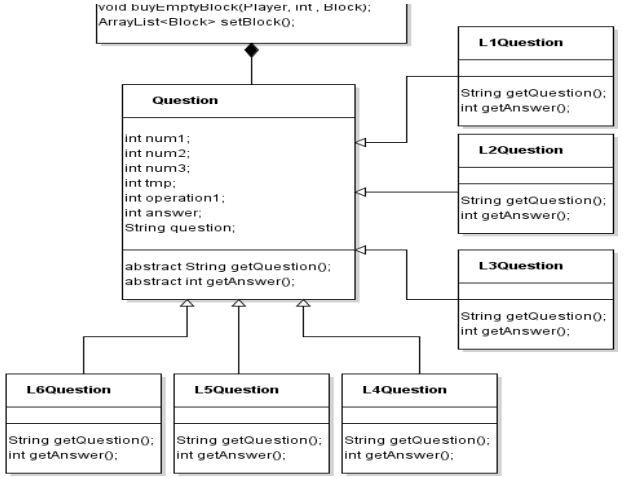
# Object-oriented programming concepts

```
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();
    }

}
```

## Polymorphism

- Question class is upper class of L1Question..

When make L1~L6Question object,
Used Question class as type.



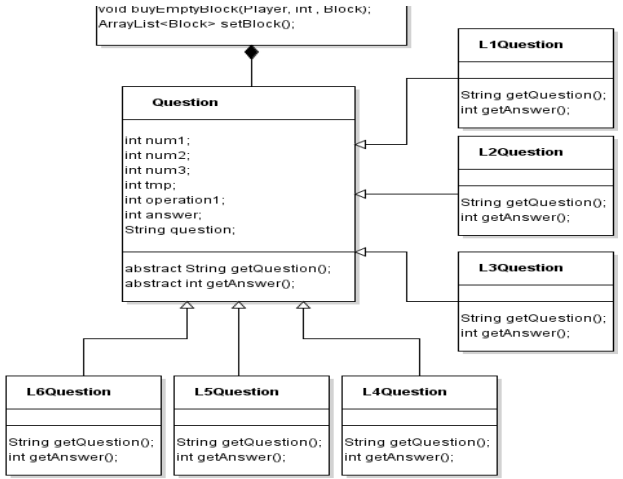

Math Witch and Colorful Kingdom

# Object-oriented programming concepts

```
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();
}
```

## abstract classes

- It uses question class as abstract class.




Math Witch and Colorful Kingdom

# Object-oriented programming concepts

```
public class Player {
    private int location; // player's current location
    private int coin; // player's current coin
    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.

    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;
    }
```

## Encapsulation

- the most of class instance variable to private and it only allows method to change the value.

Math Witch and Colorful Kingdom

# Difficult in the project work

Game :
progresses sequentially

Design :
Object-oriented

Math Witch and Colorful Kingdom

# Demonstration of the system


Math Witch and
Colorful Kingdom

# Q & A