UNIVERSITA' DEGLI STUDI DI NAPOLI
PARTHENOPE

RELAZIONE PROGETTO

TRACCIA: GIOCO DEL MONOPOLY

CANDIDATO: Giosuè Orefice
MATRICOLA: 0124001942
ANNO: 2020/2021
MATERIA: PROGRAMMAZIONE 3

PROFESSORI
ANGELO CIARAMELLA
RAFFAELE MONTELLA

SOMMARIO

1. Analisi del Problema

Il problema richiede di dover sviluppare un programma per la simulazione (semplificata) del gioco da tavolo Monopoly. La simulazione del gioco deve prevedere la visualizzazione delle varie fasi del gioco, la possibilità di salvare e ripristinare una partita e la visualizzazione della classifica dei vincitori in diverse partite effettuate.

2. Analisi dei Requisiti

Dall'analisi della traccia vengono riscontrati i seguenti requisiti, suddivisi in requisiti funzionali, requisiti non funzionali o vincoli.

2.1. Requisiti funzionali

- ✓ Un giocatore può lanciare i dadi;
- ✓ Un giocatore può decidere di acquistare una proprietà
- ✓ Un giocatore che possiede una proprietà ne gode la rendita, la quale è costituita dall'affitto che ogni altro giocatore, fermandosi sulla proprietà, è tenuto a pagargli. Le rendite o gli affitti aumentano in ragione delle case e degli alberghi che vengono eretti sul terreno.
- ✓ Un giocatore può costruire case su una proprietà quando possiede tutte le proprietà dello stesso colore
- ✓ Un giocatore può costruire un albergo solo dopo aver costruito quattro case su una proprietà le quali verranno poi restituite alla banca;
- ✓ Un giocatore può partecipare all'asta di una proprietà quando un giocatore atterrato su una determinata proprietà libera decide di non acquistarla
- ✓ Una partita può essere salvata
- ✓ Una partita può essere ripristinata se salvata precedentemente
- ✓ È possibile visionare la lista dei vincitori alle precedenti partite
- ✓ Un giocatore può finire in prigione
- ✓ Un giocatore può uscire di prigione mediante il pagamento di una somma di denaro o una volta terminati i due turni
- ✓ La banca gestisce giocatori e terreni

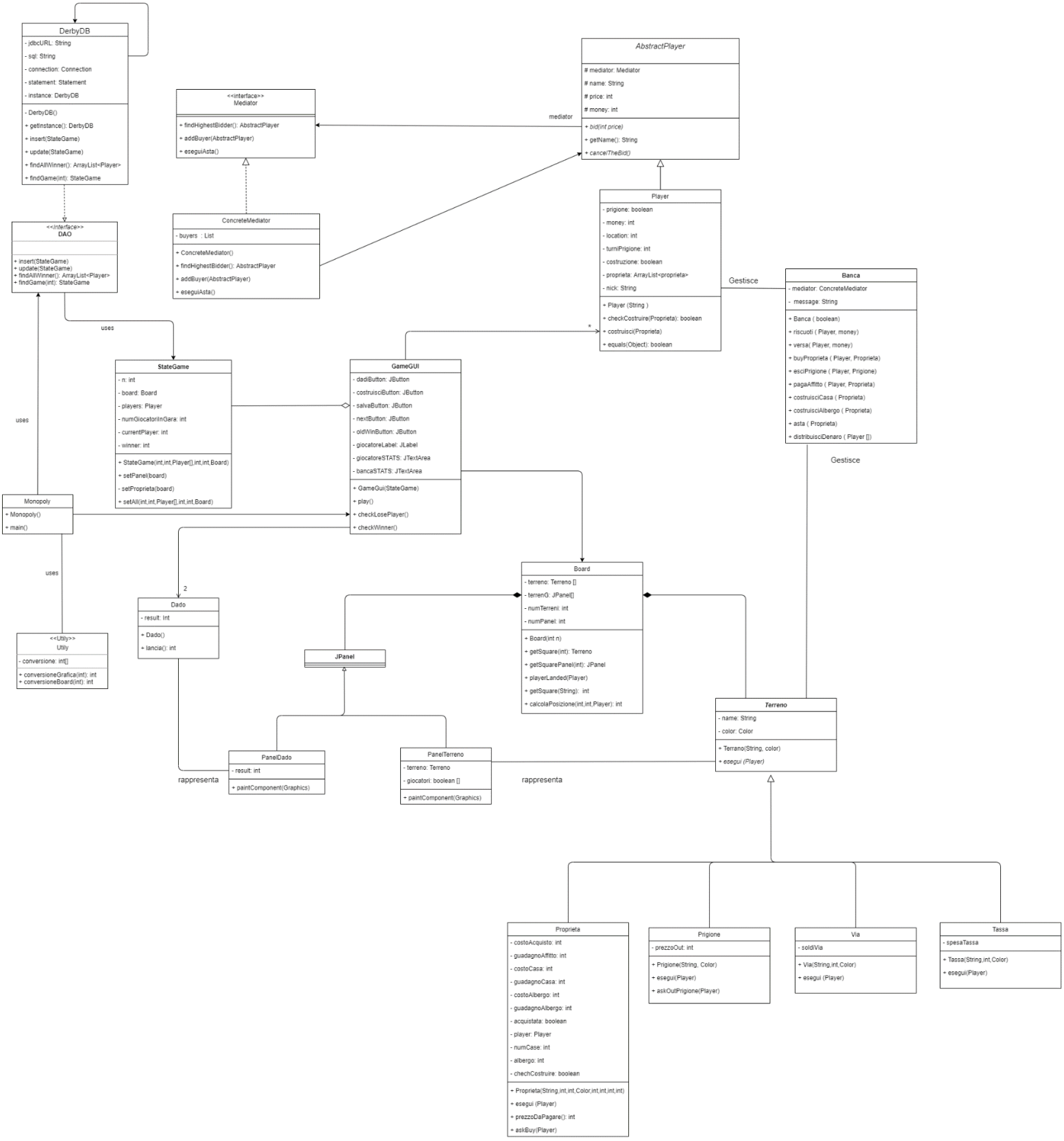
2.2. Requisiti non funzionali

- ✓ Interfaccia utente: Visualizzazione delle varie fasi del gioco;
- ✓ Robustezza: Il programma dovrebbe comportarsi in modo ragionevole in situazioni impreviste come dati di input scorretti

2.3. Vincoli o pseudo-requisiti

- ✓ Il programma deve essere sviluppato in java

3. Diagramma UML delle Classi

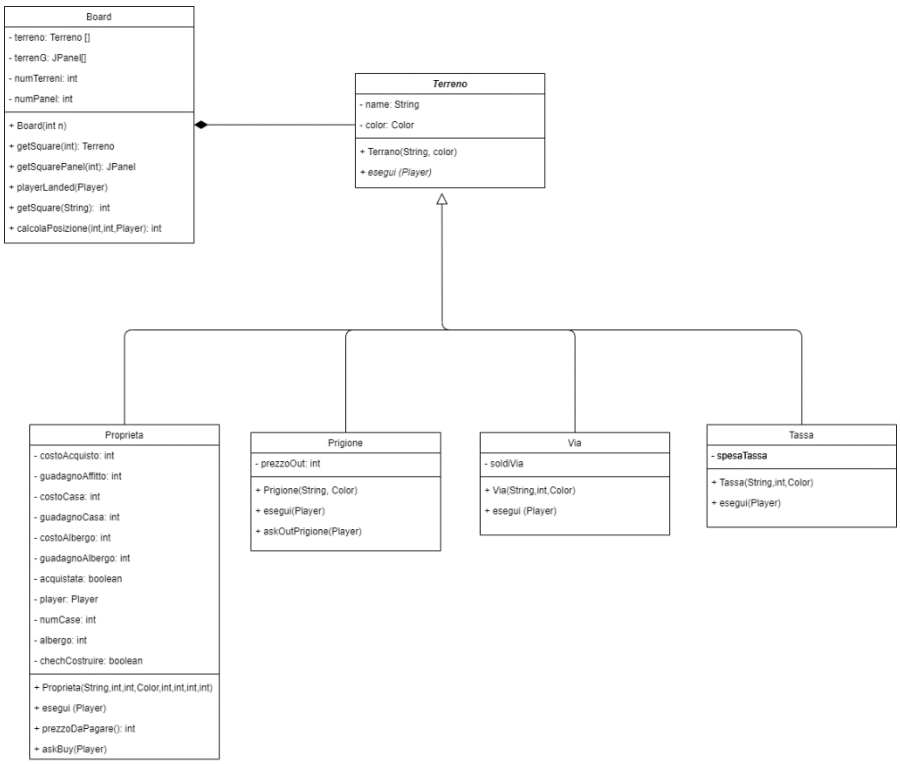


3.1. Design Pattern

I design pattern utilizzati sono:

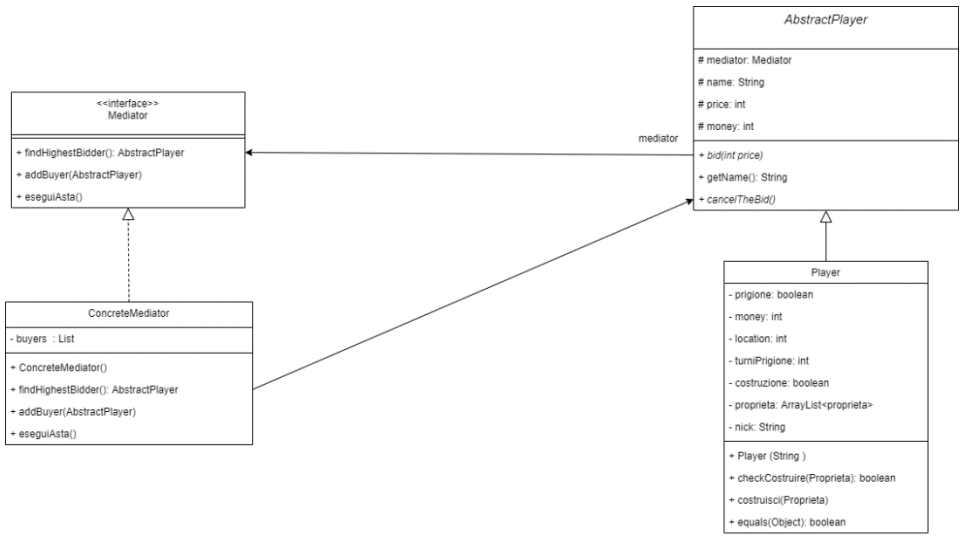
- Strategy
- Mediator
- Singleton

3.1.1. Strategy



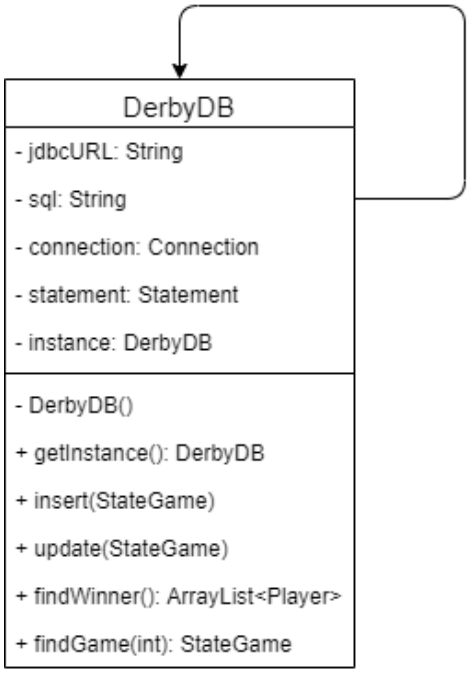
Il pattern Strategy è stato utilizzati per gestire l'atterramento di un giocatore su un determinato terreno dopo aver lanciato i dadi.

3.1.2 Mediator

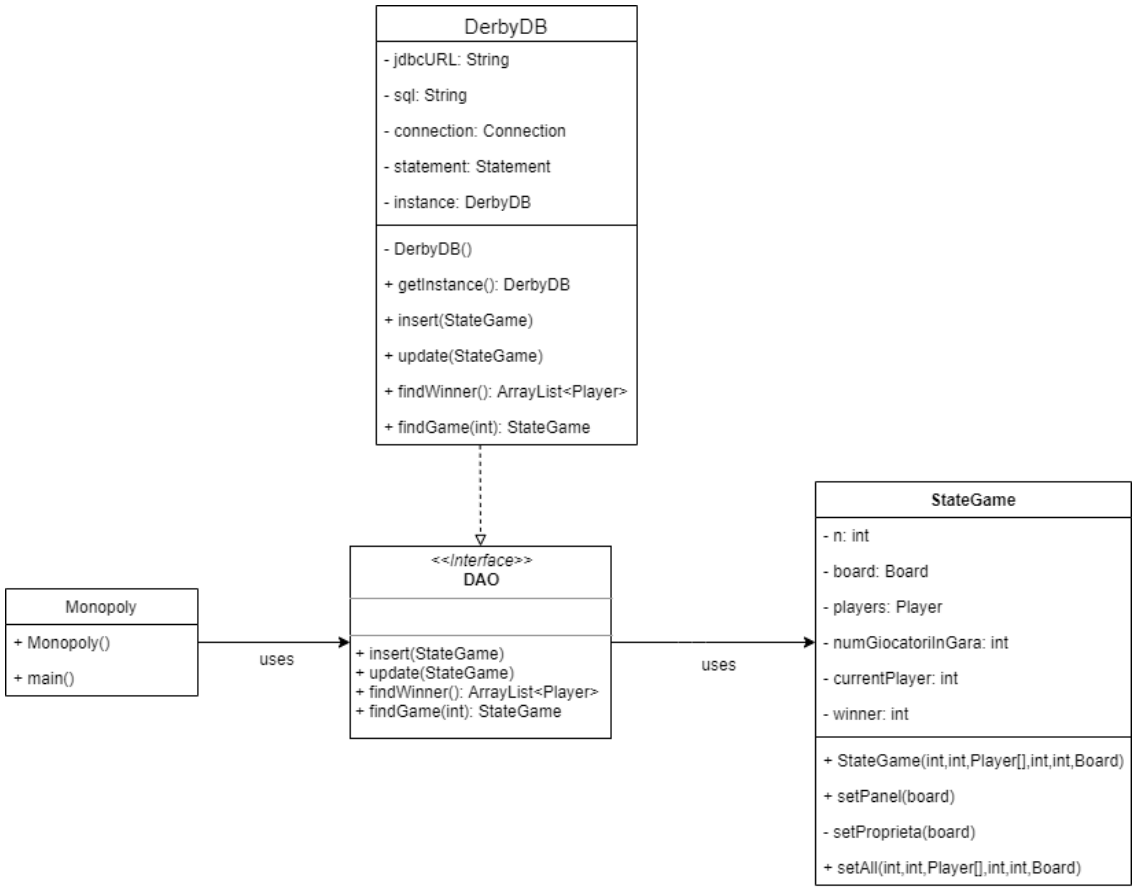


Il pattern Mediator è stato utilizzato per implementare l'asta tra i giocatori quando un giocatore atterra su una proprietà libera e decide di non acquistarla.

3.1.3 Singleton



3.2. Pattern Architetturale DAO



4. Parti rilevanti del codice sviluppato

Gestione dell'evento "Click sul bottone costruisci" con tutti i relativi controlli riguardo l'input immesso dal giocatore e i controlli sulla possibilità di costruire su una determina proprieta

```
costruisciButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String input=JOptionPane.showInputDialog(null,"Digita il nome della proprieta", "Costruzione",
                JOptionPane.QUESTION_MESSAGE);
        try {
            int posizione = board.getSquare(input);

            if(posizione == -1) {
                JOptionPane.showMessageDialog (null,
                        "INSERISCI IL NOME CORRETTO RISPETTANDO MAIUSCOLE E MINUSCOLE",
                        "ERRORE",
                        JOptionPane.ERROR_MESSAGE);
            }
            else if (board.getSquare(posizione) instanceof Proprieta) {

                Proprieta p = (Proprieta) board.getSquare(posizione);
                if(p.isCheckCostruire() && players[currentPlayer].equals(p.getPlayer())) {

                    players[currentPlayer].costruisci(p);
                    PanelTerreno k2 =  (PanelTerreno) board.getSquarePanel(Utily.conversioneGrafica(posizione));
                    k2.repaint();
                    bancaSTATS.setText("OPERAZIONI\n" +Banca.getMessage());
                    giocatoreSTATS.setText("STATISTICHE\n"+ players[currentPlayer].getStatistiche() + board.getSquare(players[currentPlayer].getLocation()).getNa
                }
                else {
                    JOptionPane.showMessageDialog (null,
                            "Non puoi costruire qui",
                            "ERRORE",
                            JOptionPane.ERROR_MESSAGE);
                }

            }

            else {
                JOptionPane.showMessageDialog (null,
                        "Non puoi costruire qui",
                       "ERRORE",
                       JOptionPane.ERROR_MESSAGE);
            }
        } catch (NullPointerException ex) {

        }
    }
});
```

Gestione dell'evento "Click sul bottone Lancia Dadi"

```
dadiButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        PanelTerreno k1 =  (PanelTerreno) board.getSquarePanel(Utily.conversioneGrafica(players[currentPlayer].getLocation()));
        if(!players[currentPlayer].isPrigione()){
            int result1 = uno.lancia();
            dadoPanelUno.setResult(result1);
            int result2 = due.lancia();
            dadoPanelDue.setResult(result2);
            pos= board.calcolaPosizione(result1, result2, players[currentPlayer]);

            PanelTerreno k2 =  (PanelTerreno) board.getSquarePanel(Utily.conversioneGrafica(players[currentPlayer].getLocation()));

            k1.setGiocatori(currentPlayer, false);
            k2.setGiocatori(currentPlayer, true);

            dadoPanelUno.repaint();
            dadoPanelDue.repaint();
            k2.repaint();
            k1.repaint();

        }
        else
            pos=players[currentPlayer].getLocation();

        board.playerLanded(pos,players[currentPlayer]);
        giocatoreSTATS.setText("STATISTICHE\n"+ players[currentPlayer].getStatistiche() + board.getSquare(players[currentPlayer].getLocation()).getName());
        nextButton.setEnabled(true);
        dadiButton.setEnabled(false);
        bancaSTATS.setText("OPERAZIONI\n" +Banca.getMessage());
        if(players[currentPlayer].isCostruzione())
            costruisciButton.setEnabled(true);
        checkLosePlayer();
    }
});
```

Asta tra i giocatori con tutti i relativi controlli sulla puntata e sull'input in generale

```
public void eseguiAsta() {
    int index,price=0,number;
    ArrayList<AbstractPlayer> playersAsta = new ArrayList<AbstractPlayer>();
    for(index=0; index<numBuyers;index++) {
        buyers.get(index).price=0;
        if(buyers.get(index).money>0)
            playersAsta.add(buyers.get(index));
    }

    int numPartecipanti = playersAsta.size();
    index = 0;
    while (numPartecipanti >1) {
        try {
            String input=JOptionPane.showInputDialog(null," " + playersAsta.get(index).getName() +
                    " quanto offri?", "ASTA",JOptionPane.QUESTION_MESSAGE);

            if(input == null) {
                numPartecipanti--;
                playersAsta.get(index).cancelTheBid();
                playersAsta.remove(index);
                if(index==numPartecipanti)
                    index=0;
                continue;
            }
            number = Integer.parseInt(input);

            if (price>= number) {
                JOptionPane.showMessageDialog (null,
                        "IL VALORE DELLA PUNTATA DEVE ESSERE MAGGIORE DI " + price,
                        "Attenzione",
                        JOptionPane.ERROR_MESSAGE);
                continue;
            }

            if(number>playersAsta.get(index).money) {
                JOptionPane.showMessageDialog (null,
                        "NON HAI SUFFICIENTEMENTE DENARO PER FARE QUESTA PUNTATA",
                        "Attenzione",
                        JOptionPane.ERROR_MESSAGE);
                continue;
            }
            playersAsta.get(index).bid(number);
            price = number;


            if(++index >= numPartecipanti)
                index = 0;

        } catch (NumberFormatException e) {}
    }


    if(playersAsta.get(0).price <= 0)
        playersAsta.get(0).price = 1;

    JOptionPane.showMessageDialog (null, " " + playersAsta.get(0).getName() + " vince l'asta  con il prezzo di "
            +  playersAsta.get(0).price + "€",
            "ASTA TERMINATA", JOptionPane.INFORMATION_MESSAGE);

}
```