Файл Game.h:

```
#ifndef GAME_H
#define GAME_H

#include "interface.h"
#include "qtypes.h"
#include <vector>
#include "bank.h"
#include "player.h"
#include "question.h"
#include "QKeyEvent"
#include <qfile.h>
#include <QTimer>
#include <QTime>
#include <QFile>
#include <QRandomGenerator>
#include <qdebug.h>

class Game : public QObject {
    Q_OBJECT
    Interface* interface;
    qint32 roundTimeLeft, roundDuration, currentRound = 0,
currentQuestionIndex = 0, currentPlayerIndex = 0, newSize = 6;
    qint32 randomIndex, time, minPlayerAnswered,
totalBankScoreInt;
    std::vector<Player*> players;
    Bank* bank = new Bank();
    std::vector<Question*> questions;
    std::vector<quint32> answeredQuestions;
    QString filePath = "D:/qtFiles/sl_zv.txt";
    QString filePathAnswers = "D:/qtFiles/sl_zv_otv.txt";
    QTimer *delayedTimer, *timer;
    int intScoreLabels[10];

public:
    Game(Ui::MainWindow* ui, qint32 roundDuration);

    void setRandomQuestion();

    void addQuestionsFromFile();

    void startNewRound(int newTime);

    void answerButtonClicked();

    void BankButtonClicked();

    void toNextPlayer();

    void passButtonClicked();

    void rightAnswersBackground_2();
```

```
    void timerSlot();

    void checkAnswer(QString userAnswer);

    void showDelayedWindow();

    void showFinishWindow();

    void minPlayerDelete();

    int findMinScorePlayer();

};

#endif // GAME_H
```

Файл Game.cpp:

```
#include "game.h"
#include "afterroundwindow.h"
#include "finishwindow.h"

void Game::setRandomQuestion() {
    if (questions.size() == answeredQuestions.size()) {
        qDebug() << "All questions were answered";
        return;
    }

    do {
        currentQuestionIndex =
QRandomGenerator::securelySeeded().bounded(questions.size());
    } while(std::find(answeredQuestions.begin(),
answeredQuestions.end(), currentQuestionIndex) !=
answeredQuestions.end());

    interface->SetQuestion(questions[currentQuestionIndex]-
>getQuestion());
}

void Game::addQuestionsFromFile(){
    const QString questionsFilePath = "D:/qtFiles/sl_zv.txt";
    const QString answersFilePath ="D:/qtFiles/sl_zv_otv.txt";
    QFile questionsFile(questionsFilePath);
    QFile answersFile(answersFilePath);

    if (!questionsFile.open(QIODevice::ReadOnly |
QIODevice::Text) || !answersFile.open(QIODevice::ReadOnly |
QIODevice::Text)) {
        qDebug() << "Failed to open question or answer file";
        return;
    }

    QTextStream inQuestions(&questionsFile);
    QTextStream inAnswers(&answersFile);
```

```
    while (!inQuestions.atEnd()) {
        QString q = inQuestions.readLine();
        QString a = inAnswers.readLine();
        Question* newQuestion = new Question(q, a);
        //Question newQuestion(q, a);
        questions.push_back(newQuestion);
    }
}

Game::Game(Ui::MainWindow *ui, qint32 roundDuration) :
roundDuration(roundDuration){
    interface = new Interface(ui);
    players.push_back(new Player("player1"));
    players.push_back(new Player("player2"));
    players.push_back(new Player("player3"));
    players.push_back(new Player("player4"));
    players.push_back(new Player("player5"));
    //interface->DataOnScreen(players, newSize);
    qDebug() << players.size()<< "huev vo rty";
    currentQuestionIndex = 0;
    interface->DataOnScreen(players, newSize);
    addQuestionsFromFile();
    startNewRound(60);
}

void Game::startNewRound(int newTime){
    interface->clearUserInput();
    newSize--;
    if (newSize == 1){
        showFinishWindow();
    }
    interface->deletePlayersFromScreen();
    interface->playersOnScreen(players, newSize);
    interface->setBankZero();
    roundTimeLeft = roundDuration;
    currentPlayerIndex = 0;
    bank->newRound();
    totalBankScoreInt = bank->getTotal();
    interface->totalBankUpdate(totalBankScoreInt);
    currentRound++;
    setRandomQuestion();
    interface->NextPlayerAnswerColor(currentPlayerIndex,
newSize);
    interface->bankColor(bank->getRightAnswerInARow());

    QTimer::singleShot(60000, this, &Game::minPlayerDelete);
    //   QTimer::singleShot(60000, this,
&Game::showDelayedWindow);

    time = newTime;
    timer = new QTimer(this);
    connect(timer, &QTimer::timeout, this, &Game::timerSlot);
```

```
    timer->start(1000);
}

void Game::answerButtonClicked(){
    checkAnswer(interface->getUserInput());
    if(currentPlayerIndex >= players.size() - 1){
        currentPlayerIndex = 0;
    } else {
        currentPlayerIndex++;
    }
    answeredQuestions.push_back(currentQuestionIndex);
    toNextPlayer();
    interface->clearUserInput();
}

void Game::BankButtonClicked(){
    qint32 a = bank->getRightAnswerInARow();
    qDebug() << a << "rrrar";
    bank->currentRound += Bank::scores[a];
    bank->setRightAnswerInARow(0);
    interface->setBank(QString::number(bank->currentRound));
    interface->bankColor(bank->getRightAnswerInARow());
}

void Game::toNextPlayer(){
    QTimer::singleShot(1500, this, &Game::setRandomQuestion);
    QTimer::singleShot(1500, this,
&Game::rightAnswersBackground_2);
    interface->disableButtons();
    QTimer::singleShot(1500, interface,
&Interface::activateButtons);
    interface->NextPlayerAnswerColor(currentPlayerIndex,
newSize);
    interface->bankColor(bank->getRightAnswerInARow());
    QTimer::singleShot(4000, interface,
&Interface::disableBankButton);
}

void Game::passButtonClicked(){
    bank->checkAnswer(false);
    interface->falseAnswerOn();
    interface->setRightAnswerOn(questions[currentQuestionIndex]-
>getAnswer());

    if(currentPlayerIndex >= players.size() - 1){
        currentPlayerIndex = 0;
    } else {
        currentPlayerIndex++;
    }
    toNextPlayer();
    interface->clearUserInput();
   // timerSlot();
}
```

```
void Game::rightAnswersBackground_2(){
    interface->rightAnswersBackground();
}

void Game::timerSlot() {
    time--;
    interface->setTime(QString::number(time));
    if (time <= 0) {
        if (timer) timer->stop();
    }
}

void Game::checkAnswer(QString userAnswer){
    qDebug() << "Game got it";
    if (questions[currentQuestionIndex]-
>checkUserAnswer(userAnswer)){
        players[currentPlayerIndex]->incRightAnswers();
        bank->checkAnswer(true);
        interface->rightAnswerOn();

    }
    else {
        bank->checkAnswer(false);
        interface->falseAnswerOn();
    }
    interface->setRightAnswerOn(questions[currentQuestionIndex]-
>getAnswer());
}

void Game::showDelayedWindow(){
    afterRoundWindow window(this, players[findMinScorePlayer()]-
>getName());
    window.playerKick();
    window.setModal(true);
    window.exec();
}

void Game::showFinishWindow(){
    finishWindow window(this, players[findMinScorePlayer()]-
>getName());
    window.playerWin();
    window.setModal(true);
    window.exec();
}

void Game::minPlayerDelete(){
    int index = findMinScorePlayer();

    showDelayedWindow();
    if (index >= 0 && index < players.size()) {
        players.erase(players.begin() + index);
    }
```

```
}

int Game::findMinScorePlayer(){
    int index = 0;
    for (int i = 1; i < newSize; i++){
        if (players[index]->getRightAnswersInThisRound() >=
players[i]->getRightAnswersInThisRound()){
            index = i;
        }
    }

    return index;
}
```

Файл Interface.h:

```
#ifndef INTERFACE_H
#define INTERFACE_H

#include "./ui_mainwindow.h"
#include "player.h"
#include "bank.h"
#include <QTime>
#include <vector>
#include <QLabel>

class Interface : public QObject{
    Q_OBJECT
    Ui::MainWindow* ui;
    QPushButton* answerButton, *passButton, *bankButton;
    QLineEdit* userInput;
    QHBoxLayout *playersLayout, *rightAnswerLayout,
*totalBankLayout;
    QLabel* bankLabels[10], *questionLabel, *rightAnswerLabel,
*timeLabel, *totalBankScore, *totalBankLabel;
    QVBoxLayout *questionLayout, *bankLayout;
    std::vector<QLabel*> playersLabels;

public:
    Interface(Ui::MainWindow* ui);;

    void setBank(QString newValue);

    void setBankZero();

    void setTime(QString newValue);

    void addPlayerWidget(Player* player);

    void totalBankUpdate(qint32 bank);

    QString getUserInput();

    void SetQuestion(const QString& string);
```

```
    void DataOnScreen(std::vector<Player*>& players, qint32
newSize);

    void playersOnScreen(std::vector<Player*>& players, qint32
newSize);

    void deletePlayersFromScreen();

    void rightAnswersBackground();

    void setRightAnswerOn(QString answer);

    void NextPlayerAnswerColor(int index, qint32 newSize);

    void bankColor(int index);

    void disableButtons();

    void activateButtons();

    void disableBankButton();

    void activateBankButton();

    void rightAnswerOn();

    void falseAnswerOn();

    void clearUserInput();

};

#endif // INTERFACE_H
```

Файл Interface.cpp:

```
#include "interface.h"

void Interface::setBank(QString newValue){
    ui->bankScore->setText(newValue);
}

void Interface::setBankZero(){
    ui->bankScore->setText("0");
}

void Interface::setTime(QString newValue){
    ui->timeLabel->setText(newValue);
}

Interface::Interface(Ui::MainWindow *ui): ui(ui) {
    userInput = ui->userInput;
    answerButton = ui->button_answer;
```

```
    passButton = ui->button_pass;
    bankButton = ui->button_bank;
    playersLayout = new QHBoxLayout(ui->playerGroupBox);
    bankLayout = new QVBoxLayout(ui->bankGroupBox);
    playersLabels.resize(5);
}

void Interface::addPlayerWidget(Player *player) {
    QLabel* label = new QLabel(ui->playerGroupBox);
    label->setText(player->getName());
    label->setStyleSheet("background-color: #0f0f0f");  //
устанавливаем начальный стиль
    playersLayout->addWidget(label);
    playersLabels.push_back(label);
}

void Interface::totalBankUpdate(qint32 bank){
    ui->totalBankScore->setText(QString::number(bank));
}

QString Interface::getUserInput() {
    return userInput->text();
}

void Interface::SetQuestion(const QString &string) {
    questionLabel->setText(string);
}

void Interface::DataOnScreen(std::vector<Player *> &players,
qint32 newSize){
    questionLayout = new QVBoxLayout(ui->questionGroupBox);
    questionLabel = new QLabel(ui->questionGroupBox);
    questionLabel->setAlignment(Qt::AlignCenter);
    questionLayout->addWidget(questionLabel);
    questionLabel->setWordWrap(true);

    //  QString imagePath = "D:/qtFiles/images/image2.jpg";
    //  QString styleSheet = "background-image: url(" +
imagePath + "); background-position: center; background-repeat:
no-repeat; background-attachment: fixed; background-size:
cover;";
    //  ui->centralwidget->setStyleSheet(styleSheet);
    // ui->centralwidget->setStyleSheet("background-image:
url(D:/qtFiles/images/image2.jpg)");

    for (int i = 0; i < 10; i++){
        bankLabels[i] = new QLabel(ui->bankGroupBox);
        bankLabels[i]->setAlignment(Qt::AlignHCenter);
        bankLabels[i]-
>setText(QString::number(Bank::scores[i]));
        bankLayout->addWidget(bankLabels[i]);
    }
```

```
    rightAnswerLayout = new QHBoxLayout(ui-
>rightAnswerGroupBox);
    rightAnswerLabel = new QLabel(ui->rightAnswerGroupBox);
    rightAnswerLabel->setAlignment(Qt::AlignCenter);
    rightAnswerLayout->addWidget(rightAnswerLabel);

    totalBankLayout = new QHBoxLayout(ui->groupBox_10);
    totalBankLabel = new QLabel(ui->groupBox_10);
    totalBankLabel->setAlignment(Qt::AlignHCenter);
    totalBankLayout->addWidget(totalBankScore);
}

void Interface::playersOnScreen(std::vector<Player *> &players,
qint32 newSize){
    for (int i = 0; i < newSize; i++){
        playersLabels[i] = new QLabel(ui->playerGroupBox);
        playersLabels[i]->setText(players[i]->getName());
        playersLabels[i]->setAlignment(Qt::AlignCenter);
        playersLayout->addWidget(playersLabels[i]);
    }
}

void Interface::deletePlayersFromScreen(){
    if (playersLayout) {
        QLayoutItem *item;
        while ((item = playersLayout->takeAt(0)) != nullptr) {
//очистка
            delete item->widget();
            delete item;
        }
    }
}

void Interface::rightAnswersBackground(){
    rightAnswerLabel->setStyleSheet("background-color: 0f0f0f");
    rightAnswerLabel->clear();
}

void Interface::setRightAnswerOn(QString answer){
    rightAnswerLabel->setText(answer);
}

void Interface::NextPlayerAnswerColor(int index, qint32
newSize){
    for (int i = 0; i < newSize; i++){
        playersLabels[i]->setStyleSheet("background-color:
0f0f0f"); }

    playersLabels[index]->setStyleSheet("background-color:
lightyellow");
}

void Interface::bankColor(int index){
```

```
    for (int i = 0; i < 10; i++){
        bankLabels[i]->setStyleSheet("background-color:
0f0f0f"); }
    bankLabels[index]->setStyleSheet("background-color:
lightyellow");
}

void Interface::disableButtons(){
    ui->button_answer->setDisabled(true);
    ui->button_bank->setDisabled(true);
    ui->button_pass->setDisabled(true);
}

void Interface::activateButtons(){
    ui->button_answer->setDisabled(false);
    ui->button_bank->setDisabled(false);
    ui->button_pass->setDisabled(false);
}

void Interface::disableBankButton(){
    ui->button_bank->setDisabled(true);
}

void Interface::activateBankButton(){
    ui->button_bank->setDisabled(false);
}

void Interface::rightAnswerOn(){
    rightAnswerLabel->setStyleSheet("background-color:
lightgreen");
}

void Interface::falseAnswerOn(){
    rightAnswerLabel->setStyleSheet("background-color: red");
}

void Interface::clearUserInput() {
    userInput->clear();
}
```

Файл MainWindow.h:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include "game.h"
#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
```

```
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

    void keyPressEvent(QKeyEvent *event) override;

    Game* getGame();

private slots:

    void on_button_answer_clicked();

    void on_button_bank_clicked();

    void on_button_pass_clicked();

private:
    Ui::MainWindow *ui;
    Game *g;
};
#endif // MAINWINDOW_H
```

Файл mainWindow.cpp:

```
#include "mainwindow.h"
#include "./ui_mainwindow.h"
#include "game.h"
#include "bank.h"

const std::vector<qint32> Bank::scores = {0, 100, 200, 500,
1000, 2000, 5000, 10000, 20000, 50000};

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)

{
    ui->setupUi(this);
    g = new Game(ui, 90);
}

MainWindow::~MainWindow()
{
    delete g;
    delete ui;
}

void MainWindow::keyPressEvent(QKeyEvent *event) {
    if (event->key() == Qt::Key_Return) {
        g->answerButtonClicked();
    }
    QWidget::keyPressEvent(event);
```

```
}

Game *MainWindow::getGame(){
    return g;
}

void MainWindow::on_button_answer_clicked()
{
    g->answerButtonClicked();
}

void MainWindow::on_button_bank_clicked()
{
    g->BankButtonClicked();
}

void MainWindow::on_button_pass_clicked()
{
     g->passButtonClicked();
}
```

Файл afterRoundWindow.h:

```
#ifndef AFTERROUNDWINDOW_H
#define AFTERROUNDWINDOW_H

#include "mainwindow.h"
#include "ui_afterroundwindow.h"
#include <QDialog>

namespace Ui {
class afterRoundWindow;
}

class afterRoundWindow : public QDialog
{
    Q_OBJECT

public:
    afterRoundWindow(Game* g, QString weakestLink);
    ~afterRoundWindow();

    void playerKick();

private slots:
    void on_pushButton_clicked();

private:
    Ui::afterRoundWindow *ui;
    Game* g;
    QString weakestLink;
};

#endif // AFTERROUNDWINDOW_H
```

Файл afterRoundWindow.cpp:

```
#include "afterroundwindow.h"
#include "ui_afterroundwindow.h"

afterRoundWindow::afterRoundWindow(Game *g, QString weakestLink)
:
    ui(new Ui::afterRoundWindow), g(g), weakestLink(weakestLink)
{
    ui->setupUi(this);

   /* QString imagePath = "D:/qtFiles/images/image3.jpg";
    QString styleSheet = "background-image: url(" + imagePath +
");"
    "background-position: center; "
    "background-repeat: no-repeat; "
    "background-attachment: fixed; "
    "background-size: cover; "
    "background-color: rgba(255, 255, 255, 100);";

    ui->groupBox->setStyleSheet(styleSheet);*/

}

afterRoundWindow::~afterRoundWindow()
{
    delete ui;
}

void afterRoundWindow::playerKick(){
    ui->textToQuit->setText(weakestLink + " самое слабое звено.
Прощайте");

}

void afterRoundWindow::on_pushButton_clicked()
{
    if (g) {
        g->startNewRound(60);
    }
    close();
}
```

Файл finishWindow.h

```
#ifndef FINISHWINDOW_H
#define FINISHWINDOW_H

#include "mainwindow.h"
#include "ui_finishwindow.h"
#include <QDialog>

namespace Ui {
class finishWindow;
```

```
}

class finishWindow : public QDialog
{
    Q_OBJECT

public:
    finishWindow(Game* g, QString weakestLink);

    ~finishWindow();

    void playerWin();

private slots:
    void on_finishButton_clicked();

private:
    Ui::finishWindow *ui;
    Game* g;
    QString weakestLink;
};

#endif // FINISHWINDOW_H
```

Файл finishWindow.cpp:

```
#include "finishwindow.h"
#include "ui_finishwindow.h"

finishWindow::finishWindow(Game *g, QString weakestLink) :
    ui(new Ui::finishWindow), g(g), weakestLink(weakestLink)
{
    ui->setupUi(this);
}

finishWindow::~finishWindow()
{
    delete ui;
}

void finishWindow::playerWin(){
    ui->finishLabel->setText("На удивление " + weakestLink + "
самое сильное звено. Поздравляю");

}

void finishWindow::on_finishButton_clicked()
{
    qApp->quit();
}
```

Файл bank.h:

```
#ifndef BANK_H
#define BANK_H
```

```
#include "qtypes.h"
#include <vector>

class Bank{
    qint32 rightAnswersInARow = 0;

public:
    qint32 currentRound = 0, total = 0;
    static const std::vector<qint32> scores;

    void newRound();

    qint32 getTotal();

    void checkAnswer(bool isRight);

    qint32 getRightAnswerInARow();

    void setRightAnswerInARow(qint32 value);

    qint32 getScore(int index);
};

#endif // BANK_H
```

Файл bank.cpp:

```
#include "qtypes.h"
#include <vector>
#include "bank.h"

void Bank::newRound(){
    total += currentRound;
    currentRound = 0;
    rightAnswersInARow = 0;
}

qint32 Bank::getTotal(){
    return total;
}

void Bank::checkAnswer(bool isRight){
    if (isRight){
        rightAnswersInARow++;
    }
    else {
        rightAnswersInARow = 0;
    }
}

qint32 Bank::getRightAnswerInARow() {
    return rightAnswersInARow;
}
```

```
void Bank::setRightAnswerInARow(qint32 value){
    rightAnswersInARow = value;
}

qint32 Bank::getScore(int index){
    return scores[index];
}
```

Файл player.h:

```
#ifndef PLAYER_H
#define PLAYER_H

#include "qwidget.h"
#include <QString>

class Player: public QWidget{
    Q_OBJECT

    int rightAnswersInThisRound;
    QString name;

public:
    Player(QString name, int rightAnswersInThisRound = 0);

    void incRightAnswers();

    QString getName();

    int getRightAnswersInThisRound();

};

#endif // PLAYER_H
```

Файл player.cpp:

```
#include "player.h"

Player::Player(QString name, int rightAnswersInThisRound):
name(name), rightAnswersInThisRound(rightAnswersInThisRound){}

void Player::incRightAnswers(){
    rightAnswersInThisRound++;
}

QString Player::getName(){
    return name;
}

int Player::getRightAnswersInThisRound(){
    return rightAnswersInThisRound;
}
```

Файл question.h:

```
#ifndef QUESTION_H
#define QUESTION_H

#include <QString>

class Question{
    QString question;
    QString answer;
public:
    Question(QString question, QString answer);

    bool checkUserAnswer(QString userAnswer);

    const QString& getQuestion();

    const QString& getAnswer();
};

#endif // QUESTION_H
```

Файл question.cpp:

```
#include "question.h"

Question::Question(QString question, QString answer):
question(question), answer(answer){}

bool Question::checkUserAnswer(QString userAnswer){
    return userAnswer.toLower() == answer.toLower();
}

const QString &Question::getQuestion() {
    return question;
}

const QString &Question::getAnswer() {
    return answer;
}
```

Файл main.cpp:

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```