**Задание:**

Реализовать настольное приложение по мотивам телевизионного шоу «Кто хочет стать миллионером?»

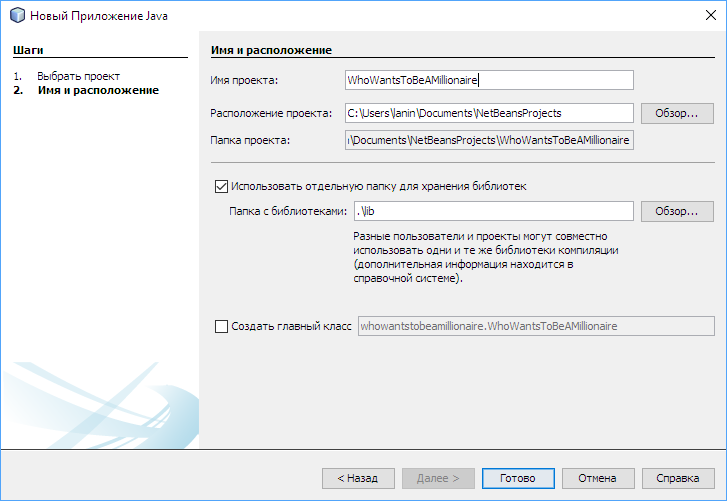
**Правила игры:** Для победы в игре игроку необходимо верно ответить на 15 вопросов из различных областей знаний. Каждый вопрос имеет 4 варианта ответа, из которых только один является верным. Сложность вопросов постоянно возрастает. Время на раздумье над каждым вопросом у игрока не ограничено.

Каждый из пятнадцати вопросов имеет конкретную денежную стоимость: 3 000 000, 1 500 000, 800 000, 400 000, 200 000, 100 000, 50 000, 25 000, 15 000, 10 000, 5 000, 3 000, 2 000, 1 000, 500.

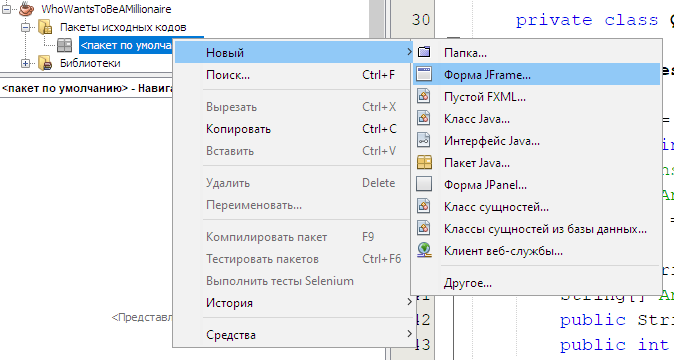
Все суммы являются заменяемыми, то есть, ответив на следующий вопрос не суммируются с суммой за ответ на предыдущий. В игре существует одна несгораемая сумма - её выбирают сами участники перед началом игры. Эта сумма остаётся у игроков даже при неправильном ответе на один из последующих вопросов.

Игроку предлагаются 5 подсказок. Участники могут использовать только четыре из пяти подсказок по ходу игры.

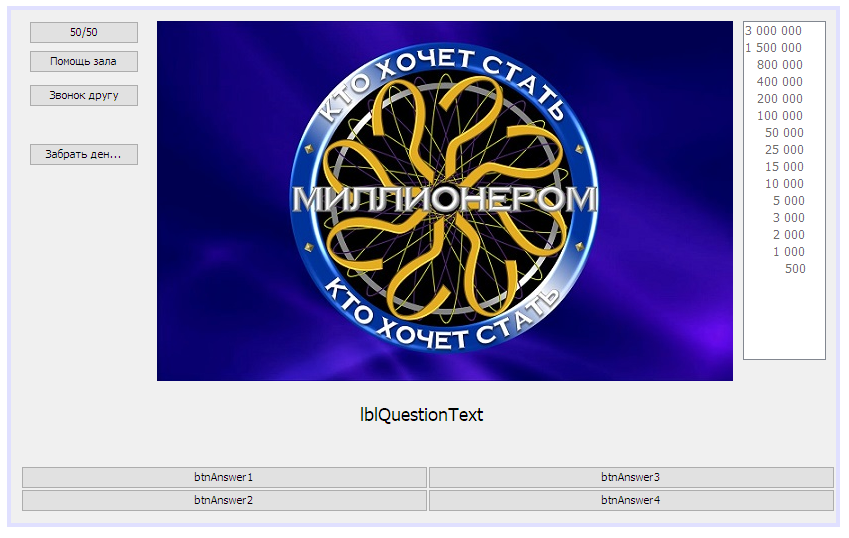
1. «Помощь зала» – каждый зритель в студии голосует за правильный, на его взгляд, ответ, а игроку предоставляется статистика голосования.
2. «50 на 50» – компьютер убирает два неправильных ответа.
3. «Звонок другу» – в течение 30 секунд игрок может посоветоваться с одним из пяти друзей, заявленных заранее.
4. «Право на ошибку» – Игрок может дать два ответа на заданный вопрос. Если первый ответ игрока оказался правильным, подсказка всё равно считается использованной.
5. «Замена вопроса» – игрок может заменить вопрос на другой того же уровня и той же стоимости.
6. Создайте новый проект с именем «WhoWantsToBeAMillionaire».



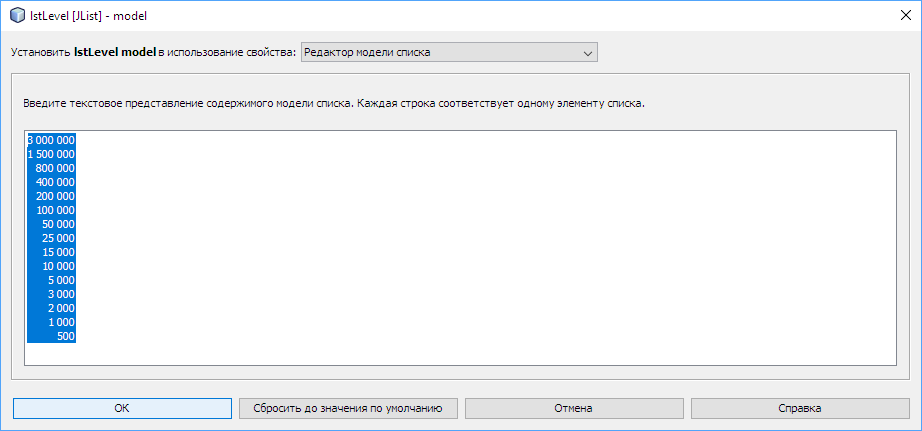
1. Добавьте новую форму JFrame и назовите ее «GameFrame».



1. Разместите на форме элементы управления как показано на рисунке ниже.



Установите свойство «model» списка lstLevel.



3 000 000

1 500 000

800 000

400 000

200 000

100 000

50 000

25 000

15 000

10 000

5 000

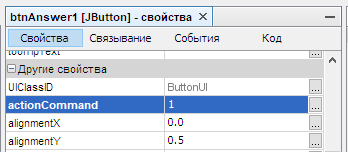
3 000

2 000

1 000

500

1. У кнопок ответов установите свойства ActionCommand значениями «1», «2», «3», «4» соответственно.



1. Добавьте к проекту описание класса Question.

public class Question

{

public Question(String[] s)

{

Text = s[0];

for (int i=0; i<4; i++)

Answers[i]= s[i+1];

RightAnswer=s[5];

Level = Integer.parseInt(s[6]);

}

public String Text;

String[] Answers = new String[4];

public String RightAnswer;

public int Level;

}

1. Добавьте в класс формы поля.

ArrayList<Question> questions = new ArrayList<Question>();

private Random rnd = new Random();

int Level =0;

Question currentQuestion;

1. Добавьте к коду формы метод чтения вопросов из файла.

private void ReadFile()

{

try{

FileInputStream fstream = new FileInputStream("Вопросы.txt");

BufferedReader br = new BufferedReader(new InputStreamReader(fstream));

String strLine;

while ((strLine = br.readLine()) != null) {

String[] s = strLine.split("\t");

questions.add(new Question(s));

}

} catch (IOException e) {

System.out.println("Ошибка");

}

}

1. Добавьте к коду формы метод отображения вопроса и ответов.

private void ShowQuestion(Question q)

{

lblQuestionText.setText(q.Text);

btnAnswer1.setText(q.Answers[0]);

btnAnswer2.setText(q.Answers[1]);

btnAnswer3.setText(q.Answers[2]);

btnAnswer4.setText(q.Answers[3]);

}

1. Добавьте к коду формы метод получения вопроса с заданным уровнем сложности.

private Question GetQuestion(int level)

{

List<Question> list =

questions.stream().filter(q->q.Level==level).collect(Collectors.toList());

return list.get(rnd.nextInt(list.size()));

}

1. Добавьте к коду формы метод получения вопроса с заданным уровнем сложности.

private Question GetQuestion(int level)

{

List<Question> list =

questions.stream().filter(q->q.Level==level).collect(Collectors.toList());

return list.get(rnd.nextInt(list.size()));

}

1. Добавьте к коду формы метод NextStep.

private void NextStep()

{

JButton[] btns = new JButton[]{btnAnswer1, btnAnswer2,

btnAnswer3, btnAnswer4};

for(JButton btn: btns)

btn.setEnabled(true);

Level++;

currentQuestion = GetQuestion(Level);

ShowQuestion(currentQuestion);

lstLevel.setSelectedIndex(lstLevel.getModel().getSize()-Level);

}

1. Добавьте к коду формы метод startGame.

private void startGame()

{

Level = 0;

NextStep();

}

1. В конструктор формы добавьте вызов методов.

ReadFile();

startGame();

1. Каждой из четырех кнопок ответов добавьте обработчик.

private void bntAnswerPerformed(java.awt.event.ActionEvent evt) {

if (currentQuestion.RightAnswer.equals(evt.getActionCommand()))

NextStep();

else

{

JOptionPane.showMessageDialog(this, "Неверный ответ!");

startGame();

}

}

1. Добавьте обработчик кнопки подсказки «50/50».

private void bntFiftyFiftyActionPerformed(java.awt.event.ActionEvent evt) {

JButton[] btns = new JButton[]{btnAnswer1, btnAnswer2,

btnAnswer3, btnAnswer4};

int count = 0;

while (count<2)

{

int n = rnd.nextInt(4);

String ac = btns[n].getActionCommand();

if (!ac.equals(currentQuestion.RightAnswer)

&& btns[n].isEnabled())

{

btns[n].setEnabled(false);

count++;

}

}

}

1. Реализовать хранение вопросов в БД Sqlite. Пример подключения и запроса представлен ниже.

try{

Class.forName("org.sqlite.JDBC");

Connection conn = DriverManager.getConnection("jdbc:sqlite:C:\\Temp\\db1.db");

Statement statmt = conn.createStatement();

String query = "select p.Name as 'Имя', CT.Name 'Тип контакта', C.Value 'Контакт' \n" +

" from Person P \n" +

" join Contact C on P.ID = C.PersonId\n" +

" join ContactType CT on CT.ID = C.ContactTypeId";

ResultSet resSet = statmt.executeQuery(query);

while (resSet.next()) {

System.out.print(resSet.getString(1)+"\t");

System.out.print(resSet.getString(2)+"\t");

System.out.println(resSet.getString(3)+"\t");

}

resSet.close();

conn.close();

} catch (Exception ex) {

System.out.println(ex.toString());

}