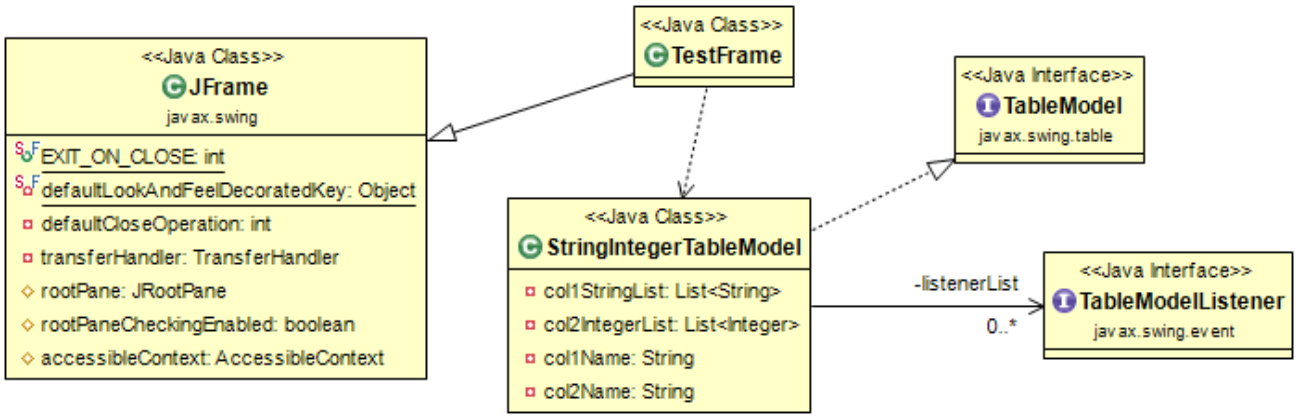
Question 1.1



Class diagram showing the relationships between StringIntegerTableModel, TableModel, TestFrame, JFrame and TableModelListener.

Question 1.2

Step1: User clicks the button.

Step2: Button fires ActionEvent to its listeners.

Step3: Activates actionPerformed method.

Step4: The nameField and the scoreField values that the user inserted are passed to the addRow method in the StringIntegerTableModel class, where they are then added to the col1StringList and the col2IntegerList arraylists respectively.

Step5: Activates fireRowAddedEvent.

Step6: The entry appears on the screen.

Question 1.3

Inversion of control allows us to reuse the code that calls into the custom code.

Question 2.1

This game follows the interface design principle of **user familiarity**.

The arrow keys to move the spaceship and the spacebar to shoot may be familiar to most players who have played this type of game before, as these controls are usually default for this type of game. The score at the top right hand corner and the 'GAME OVER' message are also common.

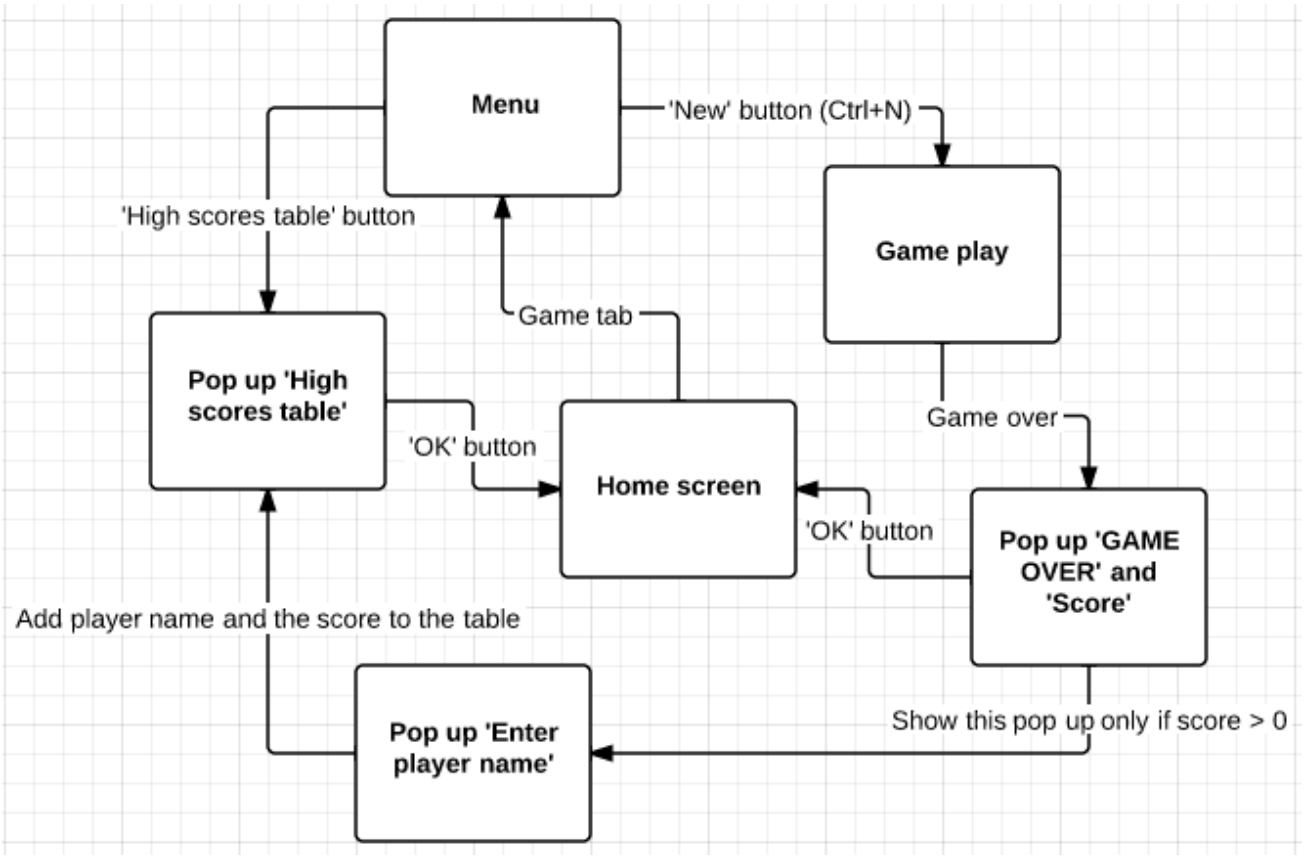
This game follows the interface design principle of **minimal surprise**.

A new player would simply try to control the spaceship by pressing the arrow keys and the spacebar and predict the outcome for each command. The menu clearly shows the buttons to start a new game, pause/unpause and end the game. However, the 'Esc' key is used for pause/unpause in this game. A new player would normally expect the game to close with a 'Esc' key but instead it would actually pause the game and may possibly surprise the player.

This game does not follow the interface design principle of **user diversity**

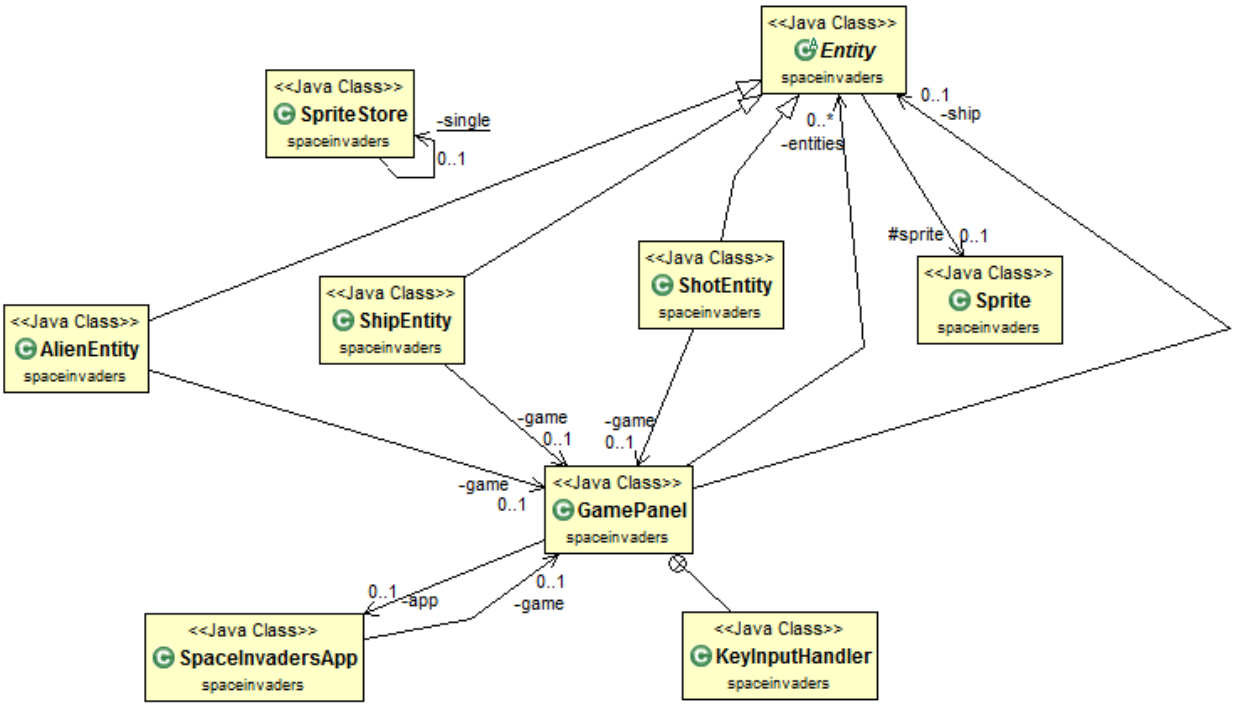
It does not support different levels for different types of users (from beginners to experts). However, there is a score based on time (decreasing over time) which allows the players to notice how well they are doing. The quicker they kill the aliens, the higher the score.

Question 2.2



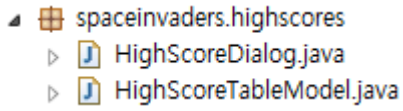
Screen flow diagram modeling the process of getting a new entry into the high score table and accessing the high score table itself.

Question 3.1



Class diagram showing the relationships between the classes present in the spaceinvaders package.

Question 3.2



Question 3.3 & 3.4 & 3.5 & 3.6 & 3.7

```
package spaceinvaders.highscores;

import java.awt.*;

@SuppressWarnings("serial")

public class HighScoreDialog extends JDialog {

    public HighScoreDialog(HighScoreTableModel model) {

        setTitle("High Scores table");
        setPreferredSize(new Dimension(200, 400));
        setLocationRelativeTo(null);

        Container cp = getContentPane();
        final HighScoreTableModel tableModel = model;
        JTable table = new JTable(tableModel);
        cp.add(table.getTableHeader(), BorderLayout.NORTH);
        cp.add(table, BorderLayout.CENTER);

        final JTextField nameField = new JTextField("A string");
        final JTextField scoreField = new JTextField("0");

        JButton button = new JButton("OK");
        button.setVisible(false);
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                tableModel.addRow(nameField.getText(), Integer.valueOf(scoreField.getText()));
            }
        });

        JPanel panel = new JPanel(new GridLayout(1, 3));
        panel.add(nameField);
        panel.add(scoreField);
        panel.add(button);
        cp.add(panel, BorderLayout.SOUTH);
        pack();
    }

}
```

Full listing of the completed **HighScoreDialog**.

```
public class SpaceInvadersApp extends JFrame {

    private final GamePanel game;
    private HighScoreTableModel model;

    final private JMenuItem menuItemGamePause;
    private HighScoreDialog dialog;

    /**
     * Create new Space Invaders application.
     * @throws HeadlessException
     */
    public SpaceInvadersApp() throws HeadlessException {
        model = new HighScoreTableModel ("Name", "Score");
        dialog = new HighScoreDialog (model);
```

```
JMenuItem highscorestable = new JMenuItem("High scores table", KeyEvent.VK_H);
highscorestable.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_H, ActionEvent.CTRL_MASK));
menuGame.add(highscorestable);
```

```
highscorestable.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        HighScoreDialog dialog = new HighScoreDialog (model);
        dialog.setVisible(true);
    }
});
```

Partial listing of the modified **SpaceInvadersApp** only showing the additional code.

Question 4.1 & 4.2

```
JOptionPane.showMessageDialog(this,
        message, "Game Over",
        JOptionPane.INFORMATION_MESSAGE);

if (game.getScore()>0){
    String s = (String)JOptionPane.showInputDialog(this,
            "You have achieved a high score!",
            "Congratulations!",
            JOptionPane.PLAIN_MESSAGE,
            null,
            null,
            "Enter your name here");

    model.addRow(s, game.getScore());
}
```

Partial listing of the modified gameEnded() method of the SpaceInvadersApp class.

Question 4.3

Test Case 1

Test procedure : User clicks Cancel button when asked to add name to high score table.

Expected behaviour : Closes the dialog and returns to the home screen.

Observed behaviour : Closes the dialog and returns to the home screen.

Test Case 2

Test procedure : User clicks OK button when asked to add name to high score table.

Expected behaviour : Closes the dialog and returns to the home screen. The player's name and the score are stored in the table.

Observed behaviour : Closes the dialog and returns to the home screen. The player's name and the score are stored in the table.