Crear códigos necesarios para guardar puntaje.

Clase HighScore (Aplicación)

```
import javafx.application.Application;

import javafx.event.ActionEvent;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextArea;

import javafx.scene.control.TextField;

import javafx.scene.layout.GridPane;

import javafx.stage.Stage;


public class HighScore extends Application {


    @Override

    public void start(Stage primaryStage) {


        GridPane gridPane = new GridPane();

        Scene scene = new Scene(gridPane, 300, 250);

        Label labelName = new Label("Nombre:");

        TextField textFieldName = new TextField();

        Label labelPoints = new Label("Puntuación:");

        TextField textFieldPoints = new TextField();

        TextArea textAreaResults = new TextArea();

        Button btn = new Button("Guardar");

        Label labelPosition = new Label("Posición:");

        TextField textFieldPosition = new TextField();
```

```
textAreaResults.setEditable(false);
textFieldPosition.setEditable(false);
gridPane.setHgap(10);
gridPane.setVgap(10);
gridPane.setPadding(new Insets(10));
gridPane.add(labelName, 0, 0);
gridPane.add(textFieldName, 1, 0);
gridPane.add(labelPoints, 0, 1);
gridPane.add(textFieldPoints, 1, 1);
gridPane.add(btn, 1, 2);
gridPane.add(labelPosition, 0, 3);
gridPane.add(textFieldPosition, 1, 3);
gridPane.add(textAreaResults, 0, 4, 2, 1);

primaryStage.setTitle("High Scores");
primaryStage.setScene(scene);
primaryStage.show();

// Creación de objetos para almacenar máximas puntuaciones
Scores scores = new Scores();
ScoresFile scoresFile = new ScoresFile();
// Cargar la lista inicial de máximas puntuaciones
scoresFile.load(scores);
// Mostrar la lista inicial de máximas puntuaciones
textAreaResults.setText(scores.toString());

btn.setOnAction((ActionEvent event) -> {
    // Recoger datos de nueva puntuación desde la ventana
    String playerName = textFieldName.getText();
```

```
            int value = Integer.valueOf(textFieldPoints.getText());
            // Crear una nueva puntuación
            Score score = new Score(playerName, value);
            // Añadirla a la lista de puntuaciones
            scores.addScore(score);
            // Mostrar la posición correspondiente a la puntuación en la lista
            //  o -1 si no está entre los primeros
            textFieldPosition.setText(String.valueOf(scores.getPosition(score) + 1));
            // Mostrar la lista de máximas puntuaciones
            textAreaResults.setText(scores.toString());
            // Almacenar la lista de máximas puntuaciones
            scoresFile.save(scores);
        });

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }

}
```

Clase ScoresFile (Almacenamiento en fichero)

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ScoresFile {

    private final File highScoreFile;

    public ScoresFile() {
        highScoreFile = new File("highscores.dat");
    }

    public void load(Scores scores) {
        FileInputStream fis = null;
        try {
            fis = new FileInputStream(highScoreFile);
            ObjectInputStream ois = new ObjectInputStream(fis);
            scores.setScoresList((ArrayList<Score>)ois.readObject());
        } catch (FileNotFoundException ex) {
            // No existe el fichero. Se creará posteriormente al guardar
        } catch (IOException | ClassNotFoundException ex) {
            Logger.getLogger(ScoresFile.class.getName()).log(Level.SEVERE, null,
ex);
        } finally {
            try {
                if(fis != null) {
                    fis.close();
                }
            } catch (IOException ex) {
                Logger.getLogger(ScoresFile.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }
    }

    public void save(Scores scores) {
        FileOutputStream fos = null;
        try {
            fos = new FileOutputStream(highScoreFile);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject((ArrayList<Score>)scores.getScoresList());
            oos.close();
        } catch (IOException ex) {
            Logger.getLogger(ScoresFile.class.getName()).log(Level.SEVERE, null,
ex);
        } finally {
            try {
                if(fos != null) {
                    fos.close();
                }
            } catch (IOException ex) {
                Logger.getLogger(ScoresFile.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }
    }
}
```

Clase Scores (Lista de puntuaciones)

```
import java.util.ArrayList;
import java.util.Collections;

public class Scores  {

    public static final int MAX_SCORES = 5;
    private ArrayList<Score> scoresList = new ArrayList();

    public ArrayList<Score> getScoresList() {
        return scoresList;
    }

    public void setScoresList(ArrayList<Score> scoresList) {
        this.scoresList = scoresList;
    }

    public void addScore(Score score) {
        scoresList.add(score);
        Collections.sort(scoresList);
        if(scoresList.size() > MAX_SCORES) {
            scoresList.remove(scoresList.size() - 1);
        }
    }

    public int getPosition(Score score) {
        return scoresList.indexOf(score);
    }
    @Override
    public String toString() {
        String result = "";
        for(int i=0; i<scoresList.size(); i++) {
            Score score = scoresList.get(i);
            result += (i+1) + "º: " + score.getName() + ": " + score.getPoints() + "\n";
        }
        return result;
    }

}
```

Clase Score (Una puntuación)

```
import java.io.Serializable;

public class Score implements Comparable<Score>, Serializable {

    private String name;
    private int points;

    public Score(String name, int score) {
        this.name = name;
        this.points = score;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getPoints() {
        return points;
    }

    public void setPoints(int points) {
        this.points = points;
    }
    @Override
    public int compareTo(Score o) {
        if (this.points < o.points) {
            return 1;
        } else if (this.points > o.points) {
            return -1;
        } else {
            return 0;
        }
    }

}
```