```java
/**
 *Elliot Duncan
 *Horton 7th
 *5/11/24
 *
 *@(#)Main.java
 *
 * Main Driver code, manages window creation.
 */
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import javax.swing.*;
import javax.swing.filechooser.*;

public class Main {
  private static JFrame mainPanel;
  /**
   *Entry point to the program
   */
  public static void main(String[] args) {

    askForFile();
    stepDisplay();
  }

  /**
   *Opens a JFileChooser and asks the user where to save the current world, and
   *then saves the current world as a *.rle file, which can the be loaded again.
   */
  public static void saveWorld() {
    JFileChooser c = new JFileChooser(new File("Patterns"));
    int returnVal = c.showSaveDialog(new JPanel());
    if (returnVal == JFileChooser.APPROVE_OPTION)
      try {
        FileWriter fw = new FileWriter(c.getSelectedFile());
        System.out.println(c.getSelectedFile());
        fw.write(World.saveWorldEncoding());
        fw.close();
        JOptionPane.showMessageDialog(new JPanel(),
                                      "The file has successfully been saved!");
      } catch (IOException e) {
        JOptionPane.showMessageDialog(new JPanel(),
                                      "File Write Error, try again.");
        System.err.println(e + ": File Write Error");
      }
  }
```

```
/**
 *Opens a new JFileChooser and has the user choose a file, which will then be
 *loaded into the world and displayed.
 */
public static void askForFile() {
  JFileChooser c = new JFileChooser(new File("Patterns"));
  c.setFileFilter(new FileNameExtensionFilter("RLE Files", "rle"));
  int rv = c.showOpenDialog(new JPanel());
  if (rv == JFileChooser.APPROVE_OPTION) {
    World.setWorld(new Seed("Patterns/" + c.getSelectedFile().getName()));
    stepDisplay();

  } else if (mainPanel == null)
    System.exit(0);
}

/**
 *Returns the first n seeds in the Patterns folder.
 *@param n the number of seeds to process. (Many seeds take a long time)
 *@return a list of seeds created from the files.
 */
public static ArrayList<Seed> getPatterns(int n) {
  return Arrays.asList(new File("Patterns").list())
      .stream()
      .limit(n)
      .map(s -> new Seed("Patterns/" + s))
      .collect(ArrayList::new, ArrayList::add, ArrayList::addAll);
}

/**
 *Given a list of seeds, returns the list of seeds but sorted by whatever the
 *Seed.compareto() method defines.
 *@param list the list to sort
 *@return the sorted list
 */
public static ArrayList<Seed> sortList(ArrayList<Seed> list) {
  ArrayList<Seed> newList = new ArrayList<Seed>();

  for (int i = 0; i < list.size() + newList.size(); i++) {
    int maxIndex = 0;
    for (int j = 0; j < list.size(); j++) {
      if (list.get(j).compareTo(list.get(maxIndex)) < 0) {
        maxIndex = j;
      }
    }
    newList.add(list.remove(maxIndex));
  }
  return newList;
}
```

```
/**
 *When I is pressed in the worldview, a menu with the list of seeds appears,
 *navigable with the left and right arrow keys. upon pressing s, the arry is
 *sorted by number of alive cells.
 *@param list the list of seeds to display
 */
public static void displayAllSeeds(ArrayList<Seed> list) {
  int[] currentPanel = {0};
  JPanel panel = new JPanel();
  JFrame j = new JFrame("Seeds Display");
  CardLayout c = new CardLayout(1, 1);
  panel.setLayout(c);

  for (int i = 0; i < list.size(); i++) {
    Seed s = list.get(i);
    panel.add(s.getName(), new SeedPanel(s, 600));
  }

  j.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  j.add(panel);
  j.addKeyListener(new KeyListener() {
    @Override
    public void keyPressed(KeyEvent e) {
      if (e.getKeyCode() == KeyEvent.VK_S) {
        ArrayList<Seed> tempList = new ArrayList<Seed>(list);

        displayAllSeeds(sortList(tempList));
        j.dispose();
      }
      if (e.getKeyCode() == KeyEvent.VK_ENTER) {
        mainPanel.dispose();
        World.setWorld(
            ((SeedPanel)panel.getComponent(currentPanel[0])).getSeed());
        stepDisplay();
        j.dispose();
      }
      if (e.getKeyCode() == KeyEvent.VK_ESCAPE) {
        j.dispose();
      }
      if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
        currentPanel[0]++;
        currentPanel[0] += list.size();
        currentPanel[0] %= list.size();
        c.next(panel);
      }
      if (e.getKeyCode() == KeyEvent.VK_LEFT) {
        currentPanel[0]--;
        currentPanel[0] += list.size();
        currentPanel[0] %= list.size();
        c.previous(panel);
        }
      }

      @Override
      public void keyTyped(KeyEvent e) {}

      @Override
      public void keyReleased(KeyEvent e) {}
    });
    j.pack();
    j.setVisible(true);
  }

  /**
   *Displays a single seed (currently unused)
   */
  public static void displaySeed(Seed s) {
    mainPanel = new JFrame("GameOfLife");
    mainPanel.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    mainPanel.add(new SeedPanel(s, 1080));
    mainPanel.pack();
    mainPanel.setVisible(true);
  }

  /**
   *Draws the window of the world, and creates the window context for which the
   *keybinds are working and timer is running.
   */
  public static void stepDisplay() {
    if (mainPanel != null)
      mainPanel.dispose();
    WorldPanel p = new WorldPanel(1024);
    mainPanel = new JFrame("GameOfLife");
    mainPanel.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    mainPanel.add(p);
    mainPanel.addKeyListener(p.getKeyListener());
    mainPanel.pack();
    mainPanel.setVisible(true);
  }
}
```