1. Classes:

- SmhgzdNotecards meets this requirement in the following places:
    - AboutViewController.java
    - AllNoteCardSets.java
    - LoadSetViewController.java
    - NewSetViewController.java
    - NoteCard.java
    - NoteCardSet.java
    - SetViewController.java
    - SmhgzdNotecards.java
    - WelcomeViewController.java

2. Subclasses:

- SmhgzdNotecards meets this requirement in the following places:
    - AboutViewController.java = extending the abstract class Switchable
    - LoadSetViewController.java = extending the abstract class Switchable
    - NewSetViewController.java = extending the abstract class Switchable
    - SetViewController.java = extending the abstract class Switchable
    - WelcomeViewController.java = extending the abstract class Switchable
    - AllNoteCardSets.java = extending the abstract class Set

3. Abstract Classes:

- SmhgzdNotecards meets this requirement in the following places:
    - Set.java (implemented by AllNoteCardSets.java)
        - The purpose of this abstract class is to ensure that AllNoteCardSets.java implements the methods addNoteCardSetToAllSets() and removeNoteCardSet();
    - Switchable.java (implemented from lecture #37, credit to Prof. Wergeles)
        - The purpose of this abstract class is to make it a lot easier to switch between scenes, with only one copy of each controller.
    - **Note**: I created the abstract class Set.java as a backup in case Switchable.java does not count towards the requirement. Although Switchable.java was not created by me, it was added to in lecture and we went over how it worked.

4. Interface:

- SmhgzdNotecards meets this requirement in the following places:
    - Serializer.java (implemented by LoadSetViewController.java and NewSetViewController.java)
        - The purpose of this class is to require these controllers to implement the method serialize(), in order to ensure serialization occurs in these controllers.

5. Collection Classes:

- SmhgzdNotecards meets this requirement in the following places:
    - AllNoteCardSets.java = line 17. (ArrayList)
    - NoteCardSet.java = line 17. (ArrayList)
    - Switchable.java = line 25. (HashMap)

6. Exception Handling:

- SmhgzdNotecards meets this requirement in the following places:
    - LoadSetViewController.java = lines 59, 121

- NewSetViewController.java = line 123
- Switchable.java = line 37
- WelcomeViewController.java = line 64

7. Model

- SmhgzdNotecards meets this requirement in the following places:
  - AllNoteCardSets.java
    - Stores all of the note card sets in an array list of <NoteCardSet>, with getters/setters
  - NoteCardSet.java
    - Stores one individual set of notecards in an array list of <NoteCard>, with getters/setters
  - NoteCard.java
    - Stores the individual note cards (front and back) within each notecard set, with getters/setters

8. Multiple Scenes and contents of scene graph changed depending on application state

- SmhgzdNotecards meets this requirement in the following places:
  - In WelcomeView.java (Multiple Scenes)
    - Clicking on the "New Set" button will change the scene from WelcomeView.fxml to NewSetView.fxml (line 32)
    - Clicking on the "Load Set" button will change the scene from WelcomeView.fxml to LoadSetView.fxml (line 46)
    - Clicking on the "About" button will change the scene from WelcomeView.fxml to AboutView.fxml (line 86)
    - Each respective view has a button that will take them back to WelcomeView.fxml
  - In LoadSetViewController.java (Change in scene graph)
    - Upon right click of set and selecting "delete" that set will be removed from the ListView that is seen on that screen (line 85)
  - In SetViewController.java (Change in scene graph)
    - Upon double click of a notecard to reveal the front/back, the content displayed in the "notecard" will change depending on what the user loaded the set with. (line 93)

9. About Page

- SmhgzdNotecards meets this requirement in the following places:
  - In WelcomeViewController.java (line 86)
    - There is a large orange button "about" on the welcome screen, this will take you to the about page
    - In the about page you will find information on how to use the application, who developed it, and why.

10. Saving and Loading Data (serialization)

- SmhgzdNotecards meets this requirement in the following places:
  - WelcomeViewController.java (line 62)
    - WelcomeViewController.java is responsible for loading (deserializing) saved notecard sets upon entry to the application
    - The method deSerialize() (line 62 and called in line 28) will handle this upon entry to the application

    - LoadSetViewController.java (line 119)
        - LoadSetViewController.java is responsible for saving (serializing) saved notecard sets upon deletion of a set, since that is the only time this class would change the model.
        - The method serialize() implemented from the Serializer.java interface in LoadSetViewController.java is on line 119 and called on line 87
    - NewSetViewController.java (line 124)
        - NewSetViewController.java is responsible for saving (serializing) new notecard sets that are created in NewSetView.java.

11. Others Code and Ideas

- I used the Switchable.java class from lecture #37, Prof. Wergeles wrote some of this code, while we added to it and went over it in class. After reading Wergeles email, I think that this can and should count towards my abstract class requirement (although I also made another back up abstract class).
- I used an idea from Raghunandan (stackoverflow) on how to display the names of my objects in my ListView in LoadSetViewController.java. I had to override the toString() method in NoteCardSet.java (line 51). I did not come up with this myself and had to look it up to come up with it.

12. Using code to create Visual elements

- In LoadSetViewController.java (line 88, 89) I create ContextMenu and a MenuItem. I did this because had I added it in scene builder to the fxml, it would have displayed the right click message I wanted anywhere on my ListView, even if it was empty. I did not want this. I only wanted to display the right click menu if the click was on a valid list item, thus doing so in code allowed for greater flexibility and accomplished this task.

13. UML Notes

- In SmhgzdUML.jpg I ordered the classes such that any class dependent on the existence of another class was below it. I also included aggregation arrows from classes containing fields of another class as shown in http://www.cs.bsu.edu/homepages/pvg/misc/uml/