Changes

	Architectural Problem	Proposed Solution	Reasoning
1	BitMapItem, SlideItem, TextItem have unused constructors	Remove unused code	Improve code readability, declutter code
2	setSlideNumber() method in Presentation class for out of bounds slide number values. If the value input is higher than the number of slides, it will update the counter, but keep the same slide on. If the value input is <1 the slide becomes blank.	Introduce an out of bounds slide number check before updating currentSlideNumber	Fix a bug
3	getCurrentSLide() in Presentation accesses the currentSlideNumber field directcly instead of using a getter.	Use a getter method getSlideNumber()	Proper usage of access levels improves security of the class.
4	Accessor is an abstract class with unused fields and only two methods. Its inherited classes do not obey the Liskov Substitution Principle. It makes more sense for Accessor to be an interface.	Make Accessor an interface and remove unused fields. Update its implementation inside XMLAccessor and DemoPresentation.	The relationship will be more understandable now. It will be easier to add any new accessor classes, improving adaptability of the application.
5	Style class has a field “styles”, which is an array that contains instances of the Style class itself.	Move the styles array, and the static getStyle() method into JabberPoint class as styles of slides are application wide and it makes sense to initialize them when the application starts up. Also, all presentations use the same styles.	Remove circular dependency. Make architecture less confusing. Improves testability as styles can be changed easier and tested with users.
6	SlideViewerComponent and Presentation classes have a circular dependency. SlideViewerFrame and SlideViewerCompnent both have an instance of Presentation. There is no clear relationship present, and any changes would prove difficult.	Create a clear relationship, where: SlideViewerFrame **has** SlideViewerComponent SlideViewerComponent **has** Presentation. Make SlideViewerComponent responsible for drawing the slide and its contents.	Create a clear and more understandable relationship between elements. Lower the number of parameters that have to be passed to methods in mentioned classes. Improve Maintainability of application
7	MenuController and KeyController are attached in different ways to SlideViewerFrame in different ways manually	Force both controllers to implement an interface (ControllerInterface). Make the said interface enforce a method that Connects a controller to SlideViewerFrame. This also means creating a class (ControllerManager) that will contain all Classes with that interface so that they can connect to SlideViewerFrame	Improves adaptability because it is now easier to add a new controller in the future. Improves testability as clear interfaces between components make it easier to design and execute tests. Additionally, according to previous feedback: “They [Controllers] should probably just talk to the SlideViewerFrame”

8	Remove the use of ImageObserver in the getBoundingBox and draw methods of an abstract class SlideItem. It is not required in all subclasses of SlideItem. Thus, passing it every time as a parameter is not required.	Move the use of ImageObserver inside the BitmapItem, where its actually needed to load pictures properly.	Lower the number of parameters that have to be passed to methods in mentioned classes. Abide by the Open-Close principle as new behaviour specific to one action is added in a subclass
9	Style class has no getters	Add getters to the necessary fields. Implement the use of those getters in Bitmapitem and TextItem	Proper usage of access levels improves security of the class.
10	Bloated code in AboutBox class. The message is constructed my concatenating strings. It is difficult to read	Use StringBuilder class to create the message. Reformat text to make it easier to change and easier to read shorter lines.	Improves code readability.