# Object-Oriented Design and Coding Review

Objectives:

- To review and assess the team's code base with respect to OO design principles
- To provide feedback to the team for further improvement of their application

Process:

- The team will walk through the complete code base with a qualified reviewer (e.g. instructor, professional developer, etc.)
- The reviewer will use a checklist and rating rubric to provide an assessment with respect to OO design principles compliance.

Assessment:

- This is not a graded activity (i.e. you will not receive a grade for this--just some recommendations)
- The team is expected to address and improve their OO design based on the recommendations made by the reviewers no later than the next Sprint review session

We will use the following Checklist & Assessment Rubric (rate the following using a scale of 1-5)

ARCHITECTURAL DESIGN

1. UML Class diagram reflects/maps to the code well (e.g. total number of class files in the code should be the same with the number of classes in the class diagram)
2. Classes are well organized in a *layered/tiered architecture* (e.g. GUI Forms, Model, Database, etc.)

OO SOFTWARE DESIGN

3. Classes are *cohesive* (i.e. they are responsible for doing only one thing)
4. Classes are *low coupled* (i.e. there is no major connectivity/communication bottlenecks and congestion in the class diagram, e.g. avoid a single large/popular/god class that everyone wants to connect to)
5. Classes support *encapsulation* (e.g. does each class have all the necessary data to accommodate all methods and functionality?)
6. Classes support *information hiding* (e.g. no public class attributes/data--all client code access only via class methods i.e. get, set properties)

CODE COMPLEXITY METRICS

7. Class size (e.g. range of 10-100 lines of code)
8. Number of attributes/data in a class (e.g. range of 1-15 attributes)
9. Number of methods in a class (e.g. range of 1-7 attributes)
10. Method size in a class (e.g. range of 1-30 lines of code)
11. Number of public methods (e.g. range of 1-7)
12. Number of private/helper methods (e.g. range 0-5)
13. Number of method parameters/signature (e.g. range of 0-5)