OOA, OOD and OOP

	OOA	OOD	OOP
Objective	What? • What classes will be part of the system? • What will each class be responsible for?	How? • How will each class fulfill its assigned responsibilities? • How will classes communicate with each other?	How in a given OO language? • How best to use the available language features to code attributes and services of classes and links/relationships between classes? • How language features accommodate and/or constrain design elements?
Activities	• Create a list of classes that will be part of your system. • The functionality of the system is distributed to classes as responsibilities. • Each class has two kinds of responsibilities: ▪ Knowledge responsibilities – what a class will need to know? ▪ Behavior responsibilities – what a class will need to do? • Types of classes ▪ Boundary, control, and entity	• For each class, convert assigned responsibilities into attributes and services. • Attributes represent knowledge responsibilities (what objects of a class know) • Services represent behavior responsibilities (what objects of a class know how to do) • Identify appropriate relationships/links between classes to enable object collaboration – dependencies, generalization, association, and realization.	For each class, • Convert attributes into static/instance fields. • Convert services into public methods. • Create helper (private) methods to support public methods. • Map relationships/links defined between classes into language specific features to realize them.
Techniques / Tools	• Brainstorming problem domain • requirements specifications • Noun extraction method • CRC method • …	• CRC method • Class diagrams • Design principles and patterns • …	• Language and IDE experience • Coding style and conventions • Test cases and automated testing • Test-driven development • …