What are Object Oriented Design Patterns:

Designing Object Oriented software systems is hard and designing reusable object oriented software is even harder. One must find pertinent objects, factor them into classes at right granularity, define class interfaces and inheritance hierarchies and establish key relationships among them.

Experienced Object Oriented Designers, over a period of software design have noticed while designing any software system (irrespective of domain) some of the design scenarios recur multiple times. So they have come up with standardized object oriented solutions for these recurring design scenarios and called them Design Patterns(Elements of reusable Object Oriented Software).

Why Design Patterns:

1) Coding is not enough: Many of the times code fixes are not going to yield solutions when there is fundamental problem in software design.
2) Design helps us to get big picture of software system: No matter how good are we in programming language(C++, Java, C#, Python, Ruby..), unless we have design skills it is impossible to understand whole software system in relatively short duration of time.
3) This is the way ahead for software developers in career: Any engineer passed out of engineering college, starts career as developer or programmer for 2…..3 years Then what is next? Next step is to take part in software system design process.
4) More than 80% of the modern software systems are object oriented: Therefore “design patterns” is very widely applicable and heavily valuable skill to have for any software developer.
5) Design Patterns are domain agnostic: Same across different application and system domains(Telecom, Finance, Banking, Insurance, Transport, Online Shops, Networks, ……)
6) Design Patterns questions have been asked in almost all product and Technology companies for 2+ years experience candidates.

Topics Covered:
1) UML Basics: Class Diagram, Sequence Diagram, Package Diagram, Use Case Diagram
2) Creational Patterns:
    Singleton, Factory Method, Abstract Factory
    Builder, Object Pool, Prototype

3) Structural Patterns:
    Adapter, Decorator, Composite
    Proxy, Façade, Bridge, Composite

4) Behavioural Patterns:
    Observer, Command, Strategy,
    Mediator, State, Template, Visitor,
    Chain of Responsibility

5) Discussion on some Design Problems and Solutions.