UML Class Diagram - Static view of the software system - doesn't change over time while the system is running
Classes, Relationships (collaborations), Attributes, Operations
Relationships - represent the strongest relationship that exists between 2 classes
Dependency—>**Association**—>*Aggregation*—>Composition—->Inheritance
Weakest—>Strongest (Slide #35)
Class design loosely coupled

UML Sequence or Communication diagrams - Dynamic view - interactions over time - when the application is running
Objects - Messages/Method calls between objects
Ordering of methods is indicated using numbered prefixes in a communication diagram
Time is indicated from top to bottom in a

sequence diagram
Frame operators (alt loop etc.) to represent complex code constructs

UML State diagrams - to represent State machines
Pseudo states - markers for starting point and final state
From the final state you will not see any arrows going out
For the initial state there are no arrows going into it
States and transitions/triggers are labeled
A trigger (event) may or may not cause a state change; it may also be an invalid transition (no arrow represented in the diagram);
Actions in a state - on entry, during, and on exit
======
PROCESS - Comparing methodologies (Waterfall, RUP, Agile - XP, Scrum, Kanban)

TESTING mechanisms (unit tests)
DESIGN - how to express design using a standard representation(UML)
DESIGN - how to come up with a good design - CRC, SOLID, GRASP, Design patterns
==========
I: interface segregation

```
public interface DBConnection {
    // public void  method1(..);
    // public void  method2(..);

.....
}
```

=======
Singleton - private constructor

Lazy evaluation/loading - create when needed - the examples in the slide
We are creating the Singleton object when it is accessed

Another approach:
Singleton - Declare and initialize at the same time
private static Singleton instance = new Singleton();
private static Singleton getInstance()
{ return instance;}

// if the construction is expensive - then lazy loading is preferred
==========
```
 public class Test<T> {


}
```

new Test<String>();

========
Adapter code is also referred to as wrapper