Notes

Java (OOP) Basics

```
public class Car {

    String color; // Attributes
    String brand; // Attributes

    public Car(String color, String brand) { // Constructor
        this.color = color;
        this.brand = brand;
    }
    void drive() {
            System.out.println("This " + brand + " is driving.");
    }
    }



// Calling the function
Car myCar = new Car("Blue", "BMW");
myOtherCar.drive();

// OUTPUT
This BMW is driving.
```

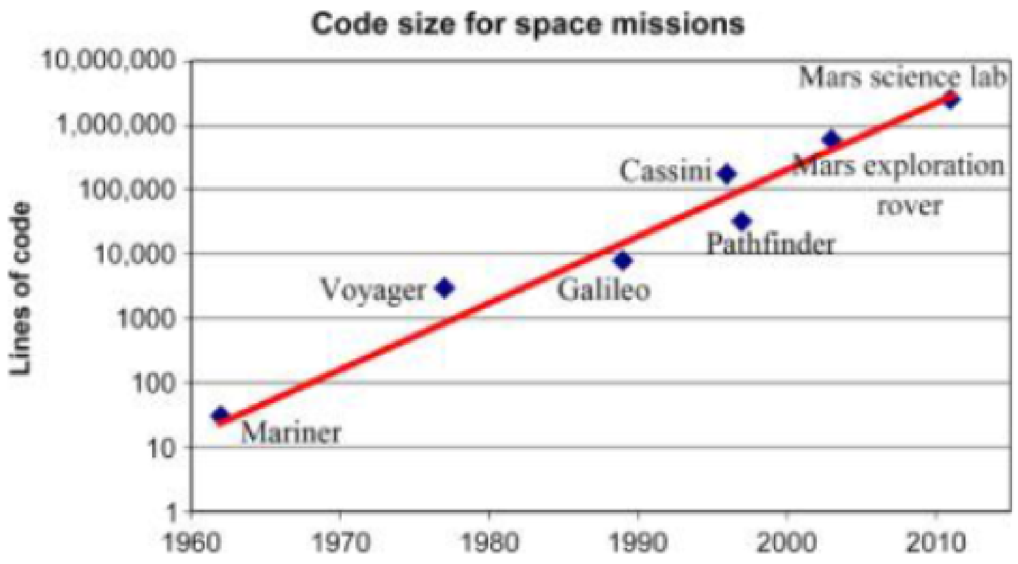
Steps:

- Class → Car.
    - Class → It is a BluePrint or Template for creating Objects.
        - It defines a set of properties (attributes) and methods (behaviors) thatthe objects created from the class will have.

        - The class doesn'trepresent an actual entity but rather a definition or a concept.
    - Color and Brand → attributes of the class Car.
- Constructor (Special Method)
    - It is special method that runs automatically when you create an object new.
    - Here it has 2 parametres Color and Brand.
    - We use the ‘this’ keyword under the Constructor.
        - `this.color` → means "the color attribute of this object".
        - Without `this`, Java would confuse between the parameter `color` and the attribute `color`.
- Method → drive()
    - The action or the Fucntion the car can do.

Complexity

- No two software parts are alike
- Complexity grow non-linearly with size.
    - It is impossible to enumerate all the states of progeram.
    - Except perhaps “toy” programs.



Changeability

- Change originates with
  - New application,users,machines,standards,laws.
  - Hardware problems
- Software is viewed as easiest to change.

World Wide Web

- This is type of Web for
  - Hyperlinked Documetns
  - Phisycal machines
  - integration between machines
- WWW's Architecture
  - Architecture of Web is seperate from the code.
  - There is no single peices fo code but rather mutiple lines of code to implement the various architecture.

Software Design Patterns

- A software design pattern is a general, reusable solution to a commonly occurring problem during software development. [Repeatable/Reusable Solutions]
    - They are not blueprints of templates, but rather GUIDELINES FOR TRACKLING PARTICULAR ISSUES.

Architecture vs patterns

Architecture is like a blueprint for a building. Patterns are like the designs for the furniture and fixtures within the rooms.

Level of Abstraction:

- Architecture is the big-picture view of the entire system's structure.
- Patterns are smaller-scale solutions that solve particular design problems within a system.

Scope of Impact:

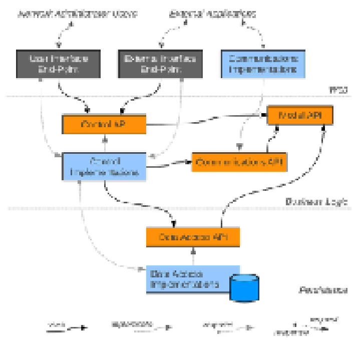
- Architecture impacts the overall structure and communication of the system's components.
- Patterns impact specific parts of the system, like how an object communicates with another or how data flows within a specific module.

Purpose

- Architecture is about defining the structure and foundation of a system, addressing global concerns like performance, scalability, and reliability.
- Patterns are about finding solutions to recurring problems within that structure, improving flexibility and maintainability.

Software Design Patterns for Web Apps (MVC)

User Interface
View (Presentation Layer)



Business Logic
Controller (Control Layer)

Data Storage
Model (Data Layer)

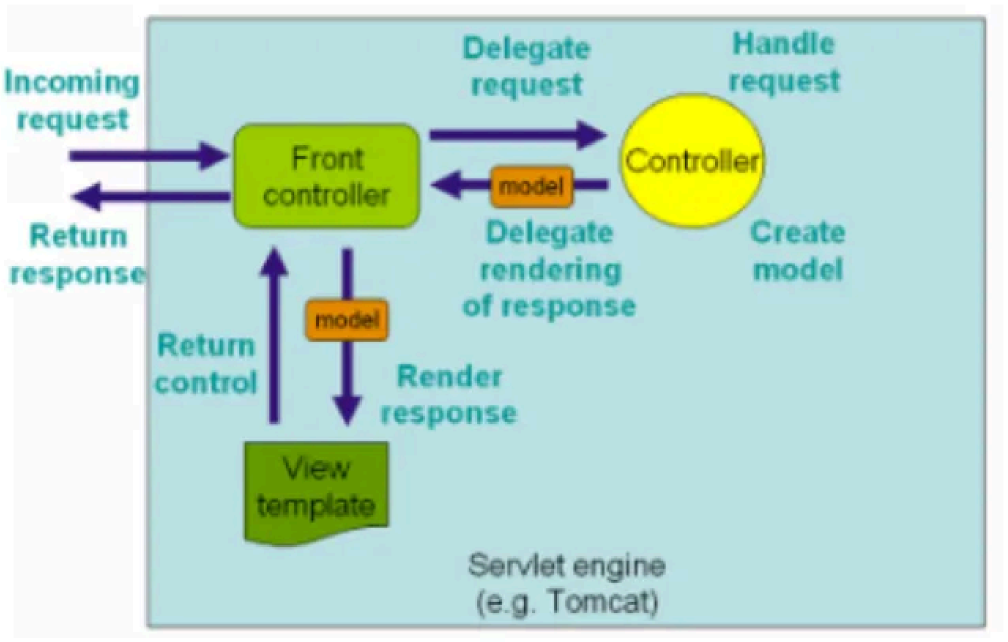
- View → manages the graphical and textual output to display that is allocated to its application.
- Controllers → It interprets inputs from the user, commanding the model and the view to change as appropriate.
- Model → It manages the behavior and teh data of the application domain.
  - responds to requests for information about its state.
  - Responds to instruction to change state.
- How MVC Fits into the Software Development Life Cycle26
  - Requirements Gathering → Identify the entities (models) and user interactions (views and controllers).
  - Design → Architect the separation of concerns. (improving modularity)
  - Implementation → Write separate modules for models, views, and controllers.
  - Testing → Easier to test each layer independently. (improving testability)
  - Maintenance → MVC allows for modifying views or controllers without affectingthe whole system.

The Spring Framework Overview

- Comprehensive Platform for Building Java-based application.
- Spring provides dependency injection and a host of modules (AOP,security, data access, etc.).
- It simplifies the use of design patterns, including MVC, in real-world applications.

Spring MVC (Model/View/Controller)

- It is a specific implementation of the MVC pattern in Java.
    - MODEL→ Java Objects (POJOs) that hold data.
    - VIEW → JSPs Thymeleaf, etc for UI rendering
    - Controller → Annotated with `@controller` and `@RequestMapping` to handle web requests.



- Why Spring Boot?
    - As a framework built on top of spring, designed to make it easier to create stand-alone, production-ready applications.
    - It automates the setup process and allows for rapid prototyping.

        - Setup process → Embedding Servers, Starter Dependencies, Auto-Configuration, Convention over Configuration.
    - Support a fasr development cycle enabling teams to move more quickly from design to implementation and testing.

Gradle: Building and Managaing Dependencies

- Gradle as a powerful build automation tool that manages project dependencies (libraries) and automates taskslike compiling, testing, and packaging.
- Thinking about software development life cycle, Gradle helpsmanage the build and deployment phases by providing a consist entenvironment.

Integrating MVC, Spring Boot and Gradle

- Spring MVC: Focus on the design and implementation stages,ensuring code modularity.
- Spring Boot: Accelerates the development, testing, and deploymentphases by simplifying configuration and setup.
- Gradle: Automates tasks across the build, test, and deploymentstages.

Spring Boot

- Spring Boot is an open-source framework designed to simplify the development of Java applications.
- It is built on top of the Spring Framework and makes it easier to create production-ready applications with minimal configuration.
- Spring Boot automates configuration, dependency management, and embedded servers (like Tomcat).