Design Patterns

Design Patterns

What are design patterns?

Why is it important to learn about them?

How do we describe them?

How do we use them?

Dictionary Entry for *Pattern*

A repeated form (especially used to decorate something).

Regular and repeated way in which something happens or is done.

Something that happens in a regular and repeated way.

Particular way in which something is organized, or happens.

Design Patterns

Codify standard solutions to recurring design requirements/problems.

Descriptions of communicating objects

customized to solve a general design problem in a particular context.

Why Should We Care About Them?

Design patterns make it easier to reuse successful designs

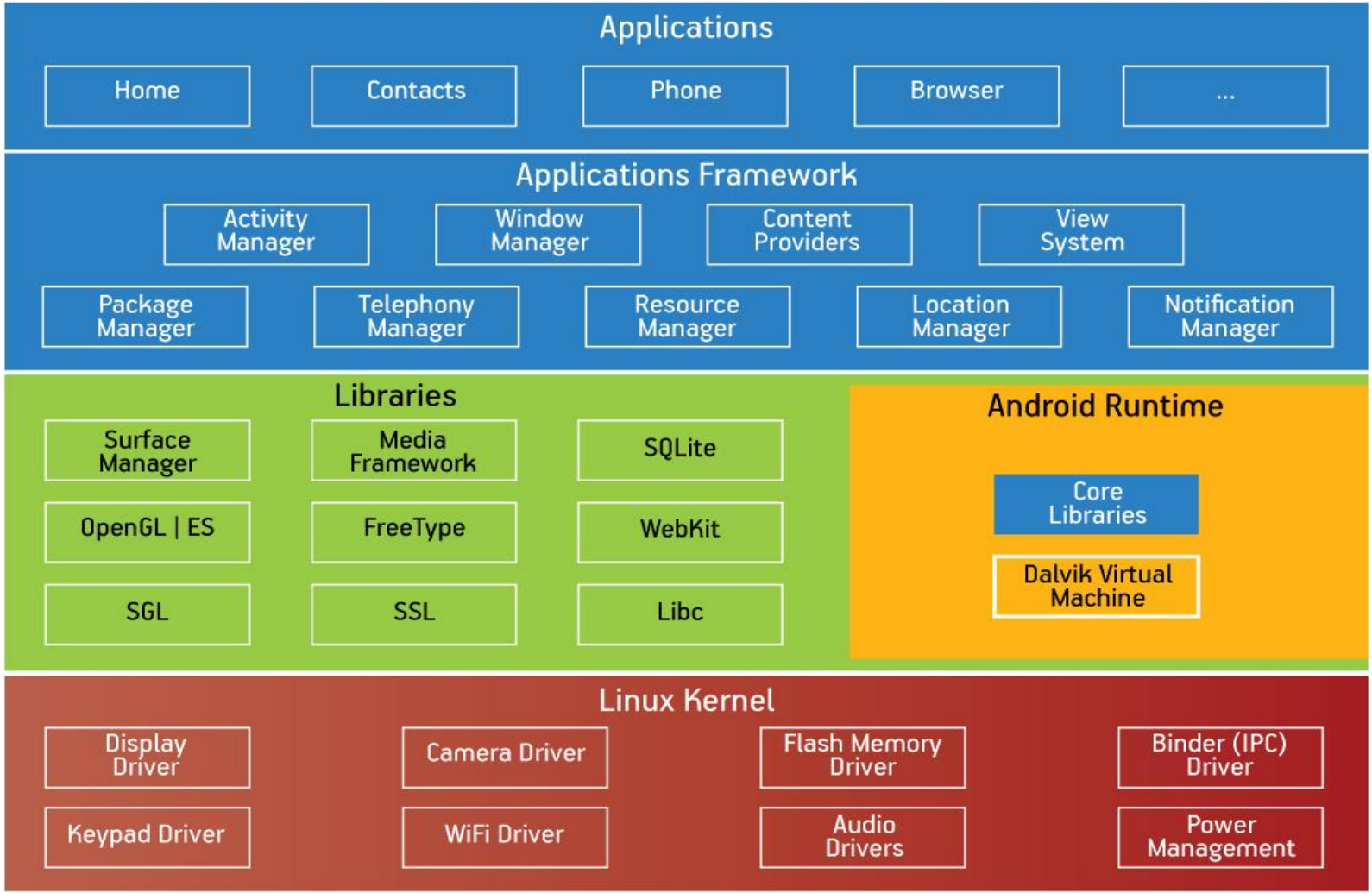
They make communication and documentation efficient.

Examples in the Software World

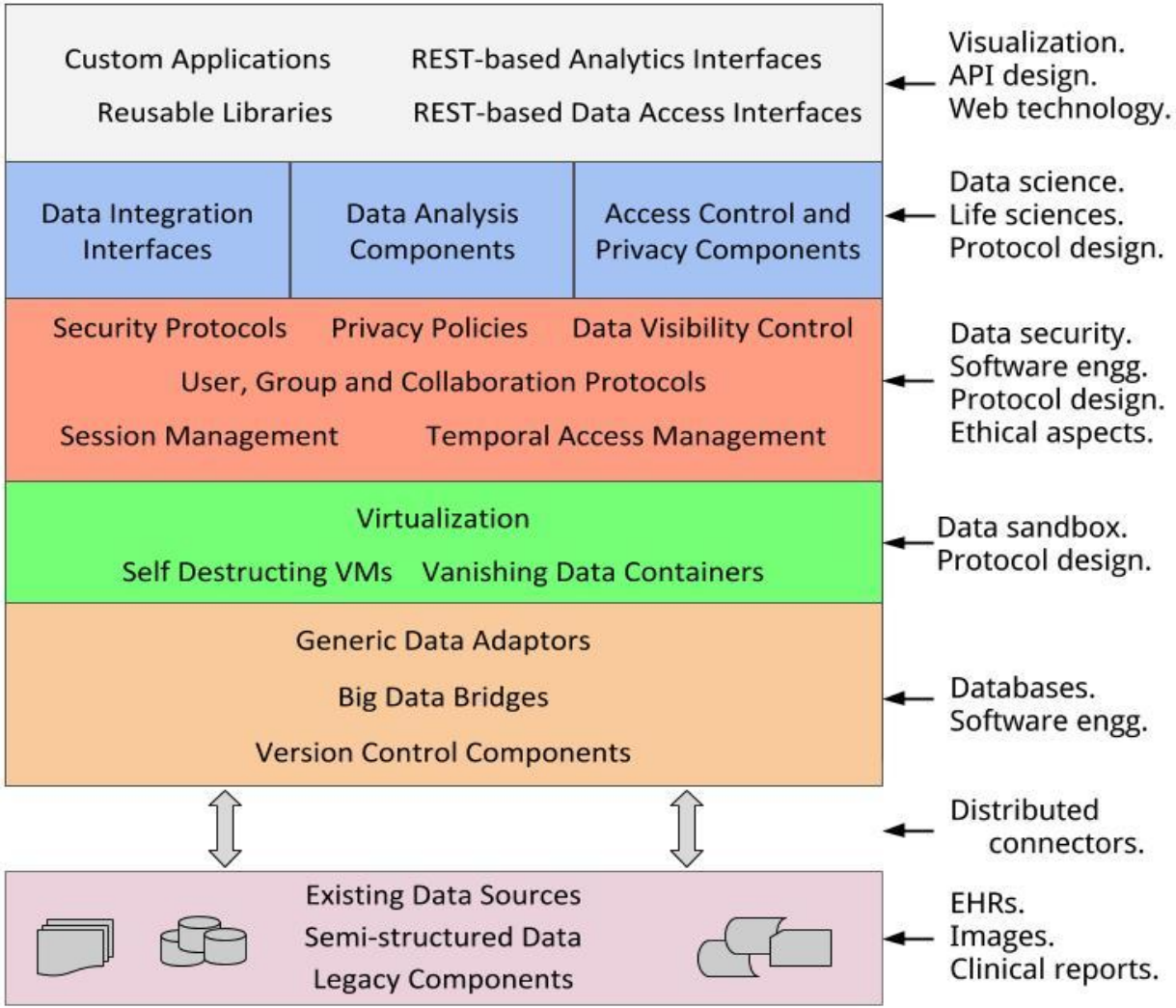
Layered system pattern

State machine pattern

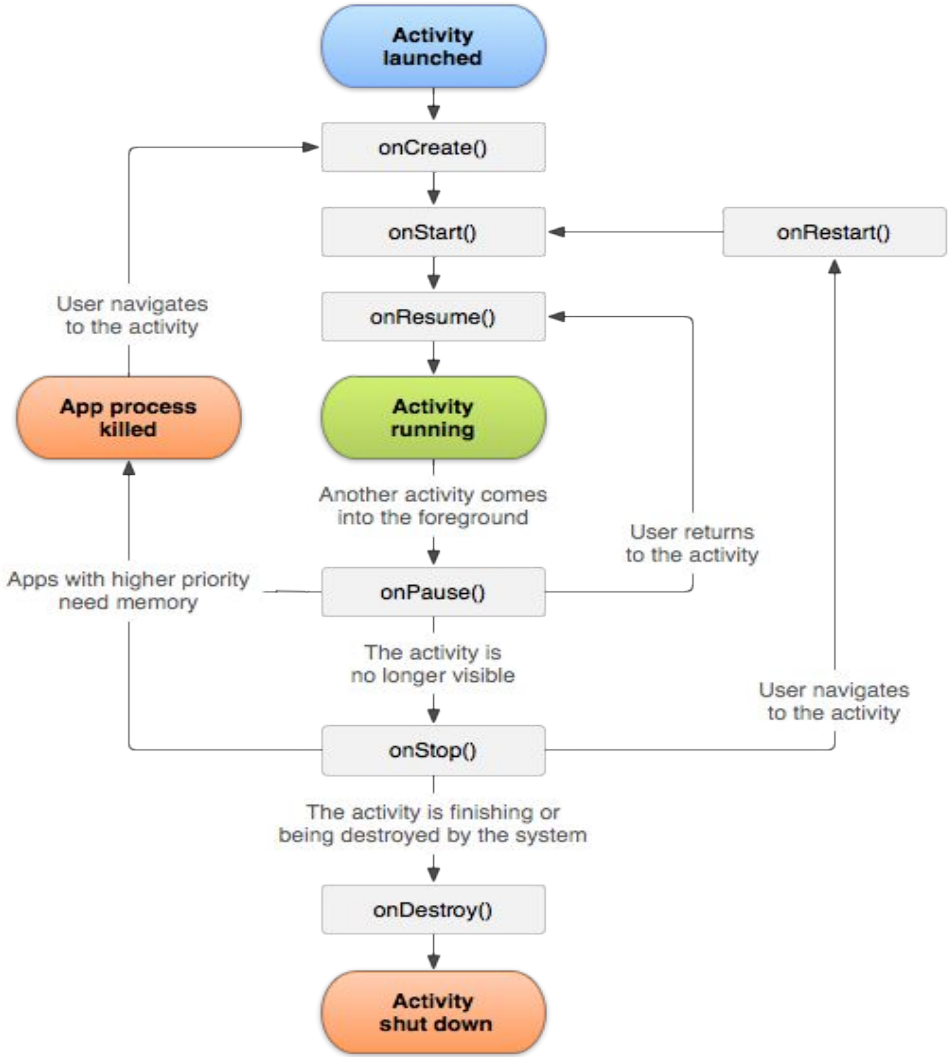
Android


Applications
Home
Contacts
Phone
Browser
...
Applications Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
Libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

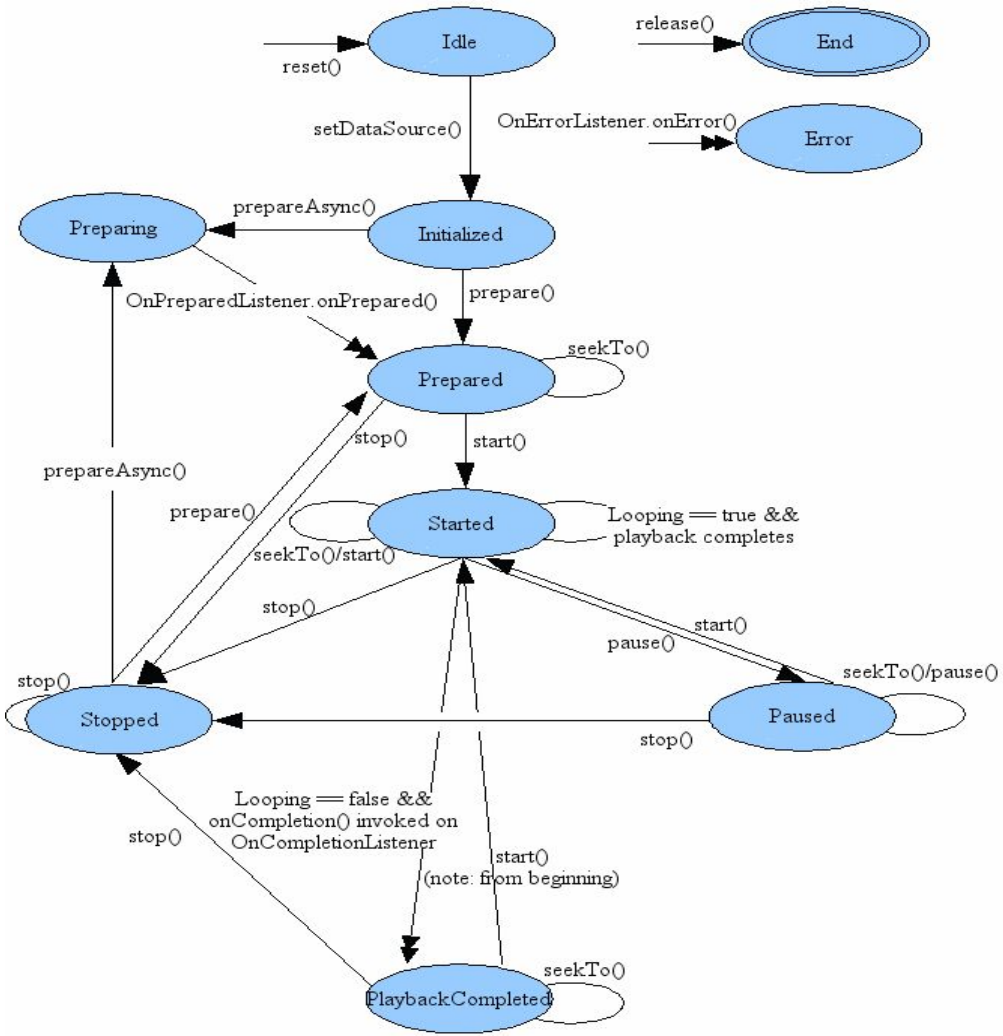
Proposed System (SOIS)



Android Activity Lifecycle



Media Player's Behavior



What are *NOT* design patterns

Not code

Not implementation (hardware or software)

Not UML

Not merely a textual description

Design Patterns

Patterns are concrete descriptions of abstract solutions.
They have to be instantiated in a particular system.

There are four essential elements

- Pattern name
- Problem
- Solution
- Consequences

Describing the Design Patterns

Use a consistent textual format.
Divided into 13 sections.

Pattern Name and Classification
Intent
Also Known As
Motivation
Applicability
Structure
Participants
Collaborations
Consequences
Implementation
Sample Code
Known Uses
Related Patterns

Builder Pattern

Pattern name

Builder

Intent

Separate the construction of a complex object from its representation so that the same construction process can create different representations.

Motivation

Not Specified

Builder

Applicability

Use the Builder pattern when the algorithm for creating a complex object should be independent of the parts that make up the object an how they are assembled.

The construction process must allow different representations for the object that's constructed.

Structure

<Show the UML model>

Builder

Participants

Builder (INameBuilder)

Specifies an abstract interface for creating parts of a Product object

ConcreteBuilder (StringNameBuilder, JSONNameBuilder)

Defines and keeps track of the representation it creates.

Provides an interface for retrieving the product

(e.g., getName, toString).

Builder

Participants

Director (NameCatalog)

constructs an object using the Builder interface.

Product (Name, JSONObject)

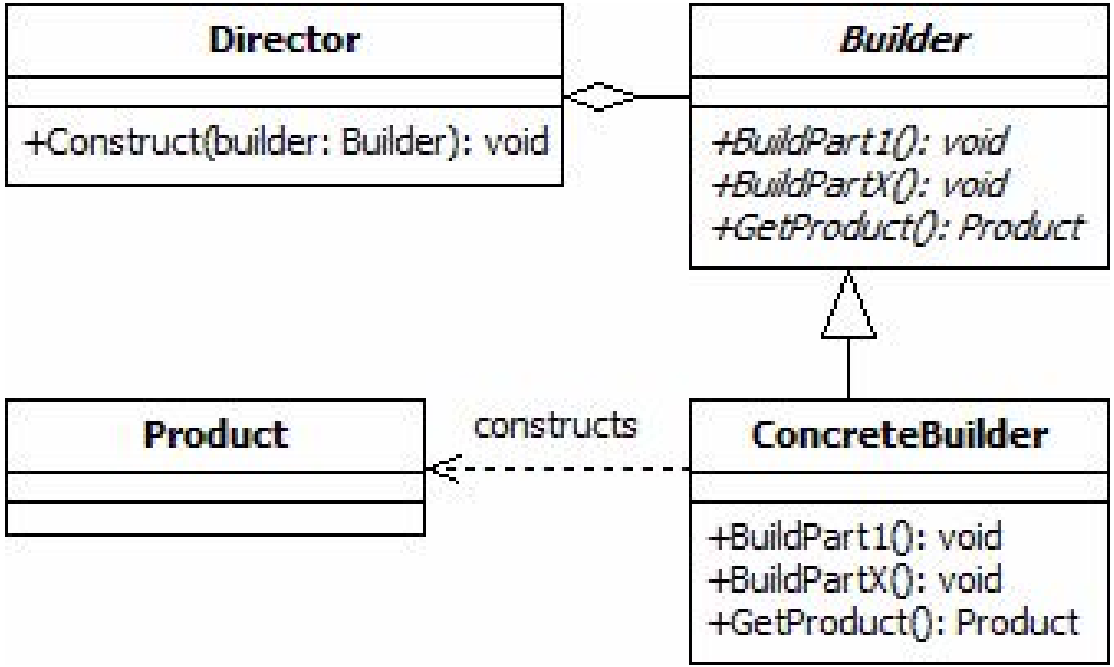
Represents the object under construction.

ConcreteBuilder builds the product's internal representation.

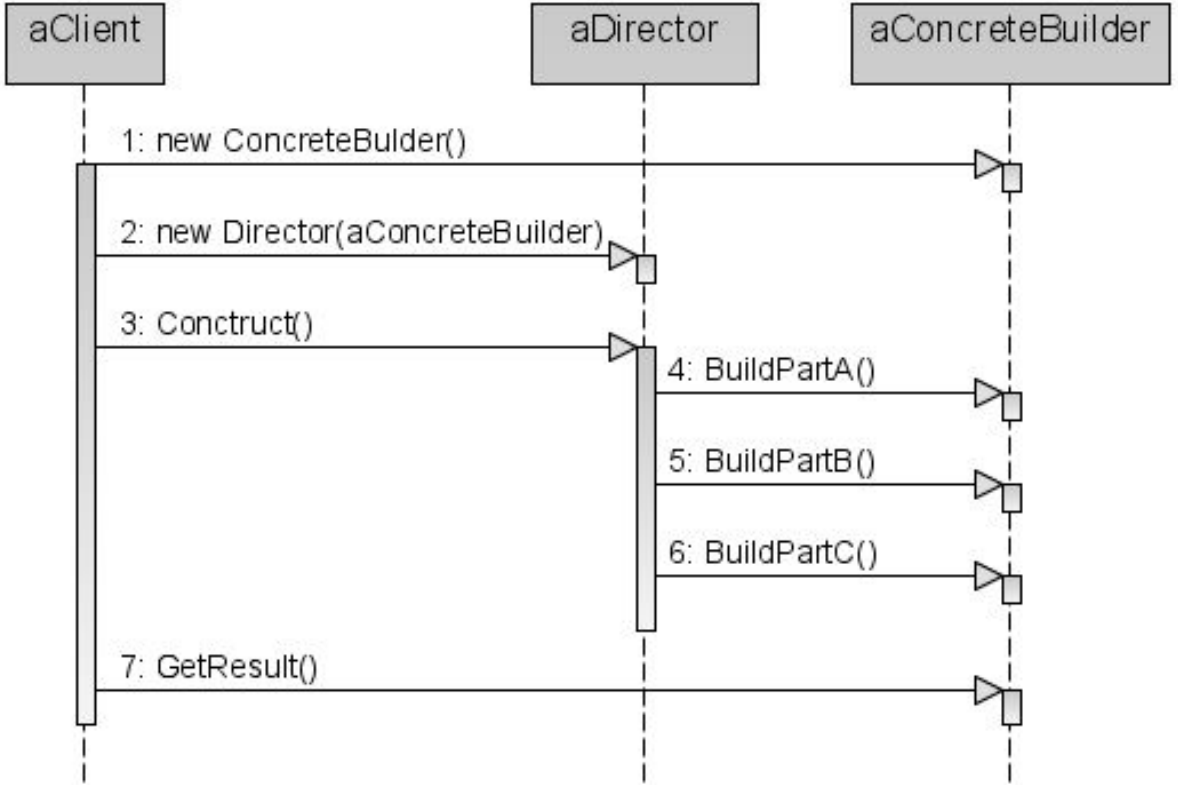
ConcreteBuilder defines the process by which it is assembled.

Builder

Collaborations



Builder


aClient
aDirector
aConcreteBuilder
1: new ConcreteBulder()
2: new Director(aConcreteBuilder)
3: Conctruct()
4: BuildPartA()
5: BuildPartB()
6: BuildPartC()
7: GetResult()

Builder

Consequences

It isolates code for construction and representation.

It improves modularity.

Builder's interface is oblivious to JSON, XML and such.

It gives finer control over the construction process.

Builder

Consequences

It allows varying a product's internal representation.

Director works with an abstract Builder interface.

Concrete builders define the structure of the product.

When a new representation is required, define a concrete Builder.

Builder

Implementation

Sample Code

Known Uses

In Java JDK library, StringBuilder is a builder.
String is an immutable product.

Builder

Related Patterns

- Abstract Factory
  - Builder builds one object with potentially complex structure in different steps.
  - Abstract Builder returns the product as a final step.
  - But as far as the Abstract Factory pattern is concerned, the product returns immediately.

Builder

Related Patterns

Composite

Builder often builds a composite.

Builder

Assignment

Construct the UML diagram for the NameBuilder example from the *src* directory.