Spring Framework

Introduction:

Spring is a Framework, which is used to develop end to end applications which is par with JEE.

In a typical web application there are 3 types of logics we write, which is also referred as 3 tiers/layers.

1. Presentation-tier logic = The code/logic we write in interacting with the end-user of the application.

2. Business-tier logic = The code/logic we write in computing the output by using the data

3. Persistence-tier logic = the logic we write for storing/accessing the data from the persistence storage devices like database or file.

While working with Struts Framework, it helps us in developing Presentation-tier aspects of developing a web application, it doesn't support building business/persistence-tier of a web application.

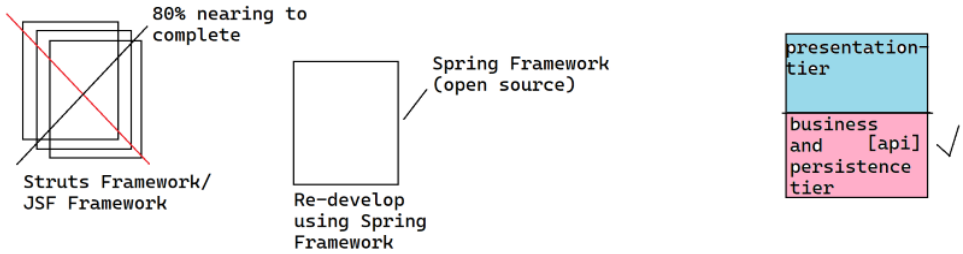
Struts is not a Framework that support building end-to-end application development, it only helps us in building Presentation-tier aspects only, so it is not a complete framework.

Unlike Struts, Spring Framework supports not only developing web application, it supports multiple application development types.

Difference between Spring and Struts?

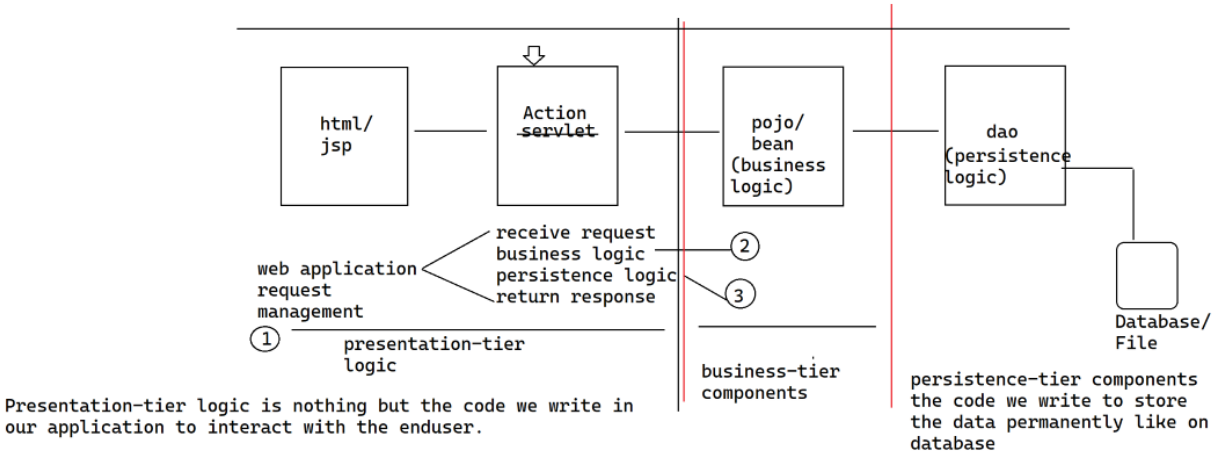
Using Struts, we can build only web applications, and we can build only presentation-tier aspects of build a web application we cannot build end-end application using the Struts Framework.

Unlike Struts, Spring Framework supports multiple types of application development types like Standalone applications, distributed web applications, enterprise applications and Integration-Tier solutions as well. Using Spring Framework, we can build almost all the application development types, and more over it supports building end-end application,



we cannot scrapout our existing investements and re-write using Spring Framework because of Spring Framework being provided freely.

Struts Framework has provided classes using which we need to develop application



How many types of applications we can build using Spring Framework?

Whatever the type of applications we can build with JEE API, all those application development types are supported by the Spring Framework. In such case why do we need to learn Spring Framework, when we can do the same thing using JEE API.

JEE (Java Enterprise Edition) is a Standard API. API stands for application programming interface. Java Standard API's provides interfaces/abstract classes only, there will not be concreate classes. So we cannot directly work with API, we need to use implementation for that API to work with, this seems to be a very complex for a novice or a beginner to understand.

By nature, API will be huge in nature, they provide lot of classes as a part of them, since we have more number of classes, learning the API takes lot of time. These classes provided by the API are inter-linked with each other, to use a class we need to know the information about its depend, thus making the API more complex to learn and use it.

From the above we can understand the API are very complex to understand and takes more time in learning them because of huge in nature and their inter-dependencies. We cannot learn an API partially and we cannot jump start in building an application, because of inter-dependencies between the classes.

by using the Spring Framework, we can develop end-end applications, so we can consider Spring Framework as a complete application development Framework.

How many types of applications we can develop using Spring Framework?

Spring Framework support not only web application development, it supports multiple application development types like

1. Standalone Application. 2. Distributed Web Application.
3. Enterprise Application. 4. Integration-Tier etc.

Using Spring Framework, we can develop multiple application development types in par with JEE.

Then why do we need to use Spring Framework when we can develop the same type of applications using JEE?

Problems in working with API:

#1. JEE Stands for Java Enterprise Edition, it is a java standard API. API's are always partial, they provide only interfaces and abstract classes there will not be any implementation classes within them. Now to work with API we always need an implementation provided by the vendor. So by looking at API/implementations a beginner/novice always find complex to work with API.

#2. API are huge in nature, they provide lot of classes as part of them, so learning API will take more amount of time.

#3. The classes within the API are inter-dependent on each other, for e.g. to use Connection class in JDBC API we need know DriverManager similarly to use Statement class we need to use Connection for creating statement. Since the classes are inter-dependent on each other learning API and understand is very complex which takes more amount of time to learn as well, than usual.

#4. We cannot partial learn an API to develop an application, since the classes are interdependent on each other unless we understand the complete API we cannot work with, so API doesn't support quick start application development

#5. API will not provide boiler-plate logic

Boiler-plate logic: it is a piece of code that has to be written redundantly/repeatedly across various different applications we are working on is called boiler-plate logic. for e.g. while we are working with JDBC API to execute select query we write same lines of code irrespective of the query we are executing as shown below.

JDBC Sample Code:

```
Class.forName("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.getConnection(url, un, pwd);

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("select * from emp");

while(rs.next()) {

  // extract data into object and use it

}
```

Now developer has to repeatedly write the same piece of code in achieving the functionality, due to which we run into lot of problems.

If API are not providing boiler plate logic, we need to write more lines of code in developing an application which will put us in more problems

1. If we are writing more lines of code in building application, we need to spend more amount of time in developing the application.

2. We need more no of developers to develop the application

3. The cost of developing the application goes high

4. The efforts of building the application is very high

5. If more lines of code, chances of increasing the bugs within our application is very high.

6. Since we have lot of code to test, the amount of time it takes in certifying or testing the application will be high.

7. The complexity in understanding the application is very high.

From the above we can understand API's doesn't support rapid application development, so go for frameworks like Spring Framework

Spring Framework supports multiple types of application development types in par with JEE. Then why do we need to learn Spring Framework to build applications when it supports same type of application development types similar to JEE.

What is a Framework?

Frameworks provides a bunch of classes, all the classes provided by the framework are concreate in nature.

What does these Framework classes will contain?

The Framework provided classes will contain pre-identified functionality, which is nothing but boiler plate logic where while developing the application with Framework we can directly use the Framework classes without writing boiler-plate logic

Advantages: -

#1. Since Frameworks directly provides concreate classes to us, the complexity of API/implementation is gone, now developer can directly work with Framework classes.

#2. Unlike API Framework will have less number of classes, so learning Frameworks is quick when compared with API.

#3. Since they provide concreate classes, there is no or minimal inter-dependency between the classes, so it is obvious that those are less complicated and quick to learn and use it.

#4. Framework supports jump start in building an application which means, we can partially learn a Framework and can begin development of the application.

#5. Framework provides boiler-plate logic, thus the number of lines of code we need to write in building the application is very less when compared with API.

5.1 it takes less time and resources in building an application.

5.2 efforts of developing the application is very less when compared with API

5.3 cost of building the application comes down

5.4 chances of bugs in the application will be very less due to less number of lines of code

5.5 The framework developers have written the boiler-plate logic by adopting the best-practices and optimizations in implementation, so while using the Framework all those best practices would come to our project as well, so we are indirectly getting benefited

5.6 Since we are writing less number of lines of code, the time required for testing/certifying the application will be less.

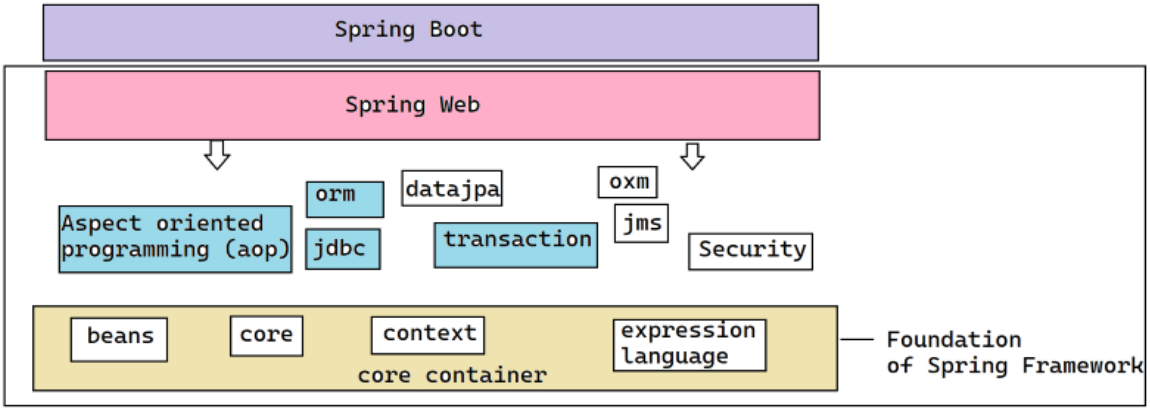
From the above we can understand frameworks support rapid application development.

Spring Framework even though it is very big, still it is being called as Light weight application development Framework, why is it being termed as light weight application development framework?

Even through Spring Framework supports lot of application development types, still it is being considered as light weight application development framework, because Spring Framework developers has not served the Spring Framework as one Single distribution. The Spring developers to make the Spring Framework learn quickly and easy to use, they broken down the Spring Framework into parts called "Module". While breaking the framework into modules, they ensured each module is at most independent of another module, so that the developers never need to worry about how big it is, rather they only need to think about what they want in it and how to use that module. So if we can use what we want out of the Framework without worrying about how big it was we can obviously say it is

“light-weight application framework “.

Let us try to understand the architecture of the Spring Framework to judge it as light weight application development framework.



How do we support Spring Framework is a light weight application development framework?

Spring Framework has been broken down into multiple modules where each module is independent of another.

The foundation or fundamental modules on which other modules of the Spring Framework work is core container which has 4 modules inside it

1. core
2. beans
3. context
4. spring expression language.

To work with Spring Framework, the developer doesn't have to learn the entire Spring Framework and should use it, instead depends on the type/nature of the application he has identify the relevant module, can learn and use along with core container the specific module of the Spring Framework.

Thus making Spring Framework light weight, since he can choose and use the module of his choice only.

What are the challenges are encountered by the Spring Framework initially? How does Spring Framework overcome them?

1. Struts Framework by then has been used by lot of people in the market and it has a huge customer base and many of the people were aware of it, being new Spring Framework, no one knows about it and doesn't have any customer base. Due to this most of the people shows interest in building their application using Struts Framework rather than Spring Framework.

2. There are lot of applications developed on Struts Framework and were successfully deployed onto production, whereas Spring Framework is a newbie, no one has used it and there are no successful deployments made through Spring Framework. So there is no guarantee that people can develop application and can delivery using Spring Framework, so most of the people go for Struts Framework.

3. By the time Spring Framework got release, the Struts Framework has got multiple releases and most of the bugs in the Struts Framework are already fixed and turned to be a Stable Framework, so if we choose Struts Framework in developing our application there is a guarantee we can complete our application development. Spring Framework just got release, it may still have bugs if we use Spring Framework there is no guarantee we can build our application, because there might be some bugs in Spring Framework which might block our application development

4. Struts Framework can be considered as next door boy, where while working with Struts Framework if got stuck somewhere, we can always find someone around us who can help us in resolving it, whereas Spring Framework being newbie, we cannot find someone who can jump in unblocking us, which is biggest problem.

What are the challenges in front of the Spring Framework in their initial days, to overcome in order to with stand in the market?

1. Struts Framework exists a way before Spring Framework, it has huge customer base and lot of users are already using the Struts Framework, when it comes to Spring Framework it has little or no presence in the market. Looks like most of the users are going to use Struts Framework only for their application development due to its presence and popularity

2. There are lot of application that are already developed and deployed in production environment using Struts Framework, looks like Spring Framework is a newbie where there are very less or no application are built on it and deployed in production. Since there are no use cases running in production, there is no guarantee that by using Spring Framework we can deliver the application into production

3. By then Spring Framework has been released there are multiple releases of Struts Framework taken place, which made it more stable. Whereas Spring Framework being newbie there are no release of the Spring Framework happened and there is no guarantee all the bugs in the Spring Framework are eliminated, so there is a risk involved in building an application using Spring Framework, during the development if we encounter any critical bug in Spring Framework, our development will be halted.

4. We can always find someone who can help us in resolving the problem, when we stuck while working with Struts Framework, because it is being used my many people in the market. whereas Spring Framework being newbie looks like most of the people don't know, if we stuck somewhere while developing application using Spring Framework we don't get any references barely that further delays the application development.

5. There is a huge community support and rich amount of documentation is available with Struts Framework being established very well, so we can always find it easy to develop the application using Struts Framework. Spring Framework is a newbie, there is no community support or documentation available in the market due to which always it is quite difficult to develop application

6. Struts Framework and Spring Framework are open source, being open source always there is a risk involved. The open source technologies might exist and may sunset by themselves at any time without providing the support or future releases due to which all the applications being developed using the open source stack will be effected. Struts Framework exists from long time and lot of people are using the Struts Framework there is guarantee struts would continue to exists for long time. whereas Spring Framework being newbie there is no guarantee it would exist for long time.

Looks like Spring has quite a number of challenges in front of them, to be popular like Struts Framework Spring should address all the above challenges as well.

- ✓ Spring Framework has added spring community support
- ✓ provided better documentation to help
- ✓ quick release of the Spring Framework made it more active and stable
- ✓ Spring Framework developers has conducted lot of technical conferences in the market
- ✓ Spring Framework built use cases and deployed on production to give market more confidence about them and partnered with many assignments helping them to make production

---

#1 Struts Framework support developing Web Applications only and we can develop only the presentation-tier aspect of a typical web application, we can consider Struts as not a complete application development framework.

Unlike Struts Framework, Spring Framework supports developing multiple application development types like Standalone application, Distributed Web Application, Integration Application and Enterprise Applications.

Spring Framework supports building all the aspects/tiers of an application, so we can consider Spring Framework as an end-to-end application development framework.

#2 Spring Framework supports developing all the application development types in par with JEE.

#3 Spring Framework is a very huge/big as it supports lot of application types, but even then also it is considered as light weight application development framework.

The Spring Framework is considered as light weight application development framework due to its modularity in nature. The Spring has been broken down into several modules where each of these modules are completely independent of each other. The only module on which all the modules Spring Framework are dependent is core container (core, beans, context, expression language).

So to work with Spring Framework the developer doesn't have to learn the entire spring and no need to use it as a whole, based on the requirement he can choose an appropriate module of the Spring Framework learn it and use for developing the application, thus making it completely light weight because I don't need to bother anything apart from what I need.

What are the challenges Spring Framework having when it was introduced in market?

1. No market footprint for Spring Framework, but there is a huge customer base already exists for Struts Framework

2. There are lot of use cases successfully deployed into production by developing through Struts Framework, whereas Spring is newbie and there are no use cases being deployed in production

3. Struts Framework is more stable without bugs, whereas Spring Framework has just then released

4. More community and documentation is available for Struts Framework, whereas Spring doesn't have

5. Struts is next door boy, where we can find many people around us who knows Struts Framework, but Spring Framework being new, we don't find anyone around us to get help

6. Struts and Spring Framework are open source, but Struts has a guarantee of existence, whereas Spring being newbie there is a risk involved that it might be removed

---

There are 2 key features of Spring Framework that makes spring unique in the market.

1. versatile application development framework
2. non-invasive application development framework

There are 2 key features of the Spring Framework that made Spring unique and more successful

1. Versatile application development framework

Spring Framework is very flexible enough in integrating with any of the existing technologies with which we build the application, so that we don't need to rewrite the entire system to use Spring Framework rather with little changes in our application we can integrate Spring and enrich the entire system.

2. Non-Invasive application development framework

usually while working with an API/framework, we need to use API/framework provided class within our application classes to build the functionality/application.

For e.g. to write an Action class in struts framework

```
class EmployeeAction extends Action {
  public ActionForward execute(HttpServletRequest request, HttpServletResponse, ActionForm, ActionMappings) {
    // logic
  }
}
```

Here, Action

ActionForm

ActionForward

ActionMappings

are the classes provided the Struts Framework, in our code we are using Framework/API classes to use that framework or API.

Here our code is tightly coupled with Framework/API since we are using Framework/API classes directly within our code. let us say we want to remove Struts Framework in our application and want to use a different framework, now how to separate the application from Struts Framework?

go to each class of our application and identify where we are using Struts Framework classes and remove the references of them, looks like we need to rewrite the entire code of our application since we are using Struts Framework everywhere.

This is called invasive, which means the framework or API code will be creeped into our code.

While working with Spring Framework, we don't need to write our code either by extending/implementing or referring Spring Framework classes. So our code is loosely coupled from Spring Framework, but still we can use the functionality of the Spring Framework. The real benefit is to separate our application from Spring Framework we need to modify zero lines code of our application, which is called non-invasive application development framework.

What does Spring Core module offer?

we are going to build an application, by writing several classes, these classes can be broadly classified into 3 types based on the nature of the code we are writing with them, those are as below.

1. POJO = plain old java object

if we can compile/run the code without using any of the external third-party libraries under classpath, then the class is called POJO class.

2. java bean

A class written with attributes, it should have 0-arg constructor and setters and getters for all of the attributes declared in that class, then the class is called java bean.

```
class Bike {
 int bikeNo;
 String manufacturer;
 double price;
 public int getBikeNo(){}
 public void setBikeNo(int bikeNo){ }
 public String getManufacturer() {}
 public void setManufacturer(String manufacturer) {}
 public double getPrice() {}
 public void setPrice(double price){}
}
```

3. bean (or) component

A bean or component class is a general purpose class which may contain attributes and methods, the methods contain arbitrary logic in performing operation, such type of classes are called "bean" or "component".

An application is build out of several classes, each of them performing various different types of functionality. Based on the nature of code written inside the classes, we can broadly classify the classes into 3 types.

In an application the bean or component classes plays a major role in building the application.

we can write a class with any amount of lines of code, but to avoid maintainability problems it is often recommended to break the code into multiple classes.

what type of maintainability problems we run into?

1. complexity in understanding the code
2. debugging is very difficult
3. we cannot reuse the code in other parts of the program

we can achieve reusability through modularity.

always break the code into multiple components so that we can reuse a part of the code in another place where ever we need.

From the above we can understand we need to modularize and distribute the code into multiple bean or component classes within an application.

Every class cannot be complete by itself, it may have to use or refer or talk to some other class to complete its functionality. So a class may be dependent on another class to complete its functionality.

How to manage dependency between the classes?

In one class we need to write the logic to talk to another class, so that it can complete its functionality.

Instead of we managing the dependencies between the classes, spring core takes care of managing the dependencies for us.

Spring core is a module, that help us in managing the dependency between the classes.

```
class A {
  void m1() {
    B b = new B();
    int j = b.m2();
  }
}
class B {
  int m2() {  }
}
```

The majority of application logic will be written as part of component classes of our application those places a crucial role in building an application.

We should not write the entire application logic into one or few component classes, we need to break down and distribute the code into adequate number of component classes for readability, maintainability and reusability.

From the above we understood we broke the code into multiple component classes, so looks a component class may need to talk to another component class to complete its functionality.

A class cannot be complete by itself, it may have to talk to some other class within our application to complete its functionality, which are called dependent classes.

How to manage dependency between the classes?

The Spring Core is all about managing the dependencies of the classes.

What is Spring Core?

Spring core is a module that help us in managing the dependency between the classes. Spring core can manage dependencies between any 2 arbitrary set of classes, but it provides recommendations in designing the classes, so that those can be better managed through spring core and we take lot of advantage of using Spring Framework.

Spring Core recommends us to design our application classes based on Strategy Design Pattern, so that we can get more benefit of using Spring Framework in developing our application.

What is Spring core is about and, what does it provide?

Spring core is all about managing the dependencies between the classes.

Even though Spring core can manage dependencies between any 2 arbitrary set of classes, it recommends us to design the classes based on the Strategy Design Pattern and give to him. So, that it can better manage the dependencies and we can get benefited out of using Spring Framework.

What is a Design Pattern?

For a recurring problem, there is pre-computed best solution that can be applied under a specific context to solve the problem together documented is called design pattern.

The roots or notion of design patterns in the software engineering world has been started by 4 people Richard Helm, Ralph Johnson, Enrich Gamma, John Vissel who are popular know as Gang of Four. The document the recurring problems and their best applicable solutions under a context and published a book called "Elements of Reusable object oriented Software".

These people have document the problems and solutions that we generally face while building the application on object oriented programming principles.

Strategy Design Pattern is one of the design pattern out of the patterns of GOF, that Spring people recommends us to use in design the application classes, to get more benefited.

Strategy Design Pattern

The strategy design pattern has provided 3 principles based on which we need to design our application classes.

1. Favor composition over inheritance
2. always design to interfaces, never design to concreate classes
3. your code should be open for extension and should be closed for modification

---

#1. Favor composition over inheritance

If a class wants to reuse the functionality of another class, there are 2 ways are there

1. Inheritance

2. Composition

Inheritance: -

We can establish Inheritance relationship between the classes by extending one class from another.

eg.

```
class A extends B {

}
```

When we inherit one class from another class, all the traits of the parent will be part of the child class. The methods of the parent can be called by child class method directly as if those methods are also part of the child without using the object of parent.

Inheritance always establishes IS-A relationship between the classes. IS-A relationship means always the child can be expressed in terms of Parent.

Spring core is all about managing the dependency between the classes. Spring core can manage the dependencies between any 2 arbitrary set of classes, but it recommends us to design the classes based on a design pattern called "Strategy Design Pattern", to get most benefited out of using Spring Framework.

For a recurring problem, there is a best applicable solution that can be used/applied to solve the problem, under a context documented together is called a "Design Pattern".

Strategy design pattern is one of the design patterns defined as part of GOF Patterns.

We need to design our application classes based on Strategy Design Pattern, and pass them to Spring Framework to better manage the dependencies.

Strategy design pattern has 3 principles following which we need to design our classes.

1. Favor composition over inheritance

2. Always design to interfaces never design to concreate classes

3. Code should be open for extension and closed for modification

#1. Favor composition over inheritance

There are 2 ways a class can reuse the functionality of another class

1. Inheritance

2. Composition

#1. What is Inheritance?

A class can reuse the functionality of another class through inheritance. We can establish an inheritance relationship between the classes by extending one class from another class.

```
class B {
  int m2(int i) {
    // operation
    return 34;
  }
}
```

```
class A extends B {
  void m1(int i) {
    int j = m2(i);
    // operation
  }
}
```

when we inherit a class from another class all the traits of the parent class will be part of child. So that child can directly use the methods or attributes of the parent as if those are part of child only.

Here A class can call the method m2(int i) of B class without using the object of B class, because A is having the traits of B inside it.

Inheritance establishes IS-A relationship between the classes, which means always a child can be expressed interms of parent.

#2. What is composition?

The another way we can reuse the functionality of another class inside a class is through "Composition". We declare another class as an attribute inside our class, instantiate the object of other class and use the methods and attributes of other class through the reference of the other.

```
class A {
  B b;
```

```
  void m1(int i) {
    b = new B();
    int j = b.m2(i);
    // operation
  }
}

class B {
  int m2(int i) {
    // operation
    return 93;
  }
}
```

Composition always refer to a thing/or a substance has been made up of what other things. For eg.. a Bicycle is made up on lot of other parts like break, chain, frame, handle etc.

Composition establishes HAS-A relationship between the classes, which means we have the reference of other to use the functionality of other class.

when to use inheritance, and when we should go for composition?

Inheritance: -

1. If we want to reuse all the traits of another class, within our class then use Inheritance.
2. if we can replace a parent with the reference of child, which means always a child can act as an substitute of the parent. (nothing but child is-a parent)

```
class Printer {
 - init() {}
 - print() {}
 - diagnose() {}
}
```

```
class LaserJetPrinter extends Printer { // here child can be expressed interms of parent


}


class Engine {
  void ignite() {}
  void accelerate(int kmph) {}
  void break() {}
}


// this is wrong, because we cannot replace a Engine with Car. It is not a IS-A relationship.
class Car extends Engine {


}
```

When to go for composition?

If we want to partially use the functionality of another class inside our class, then use Composition, do go for Inheritance.

What is a design pattern?

For a recurring problem under a context, there is best applicable solution that can be applied to solve the problem together documented called "design pattern".

GOF design patterns document problems and the solution we encounter while building the applications on object oriented programming languages. one of the design pattern out of GOF patterns is "Strategy Design Pattern".

Strategy Design Pattern defined 3 principles, which we need to apply in writing our application classes.

1. Favor composition over inheritance

2. always design to interfaces, never design to concreate classes

3. code should be open for extension and closed for modification

#1. Favor composition over inheritance

There are 2 ways a class can reuse the functionality of another class. Either through Inheritance or using Composition.

Inheritance:

When we go for inheritance in reusing the functionality of another class, all the traits of parent will be inherited (part of) to child class. we use extends keyword in deriving the inheritance relationship. Inheritance establishes "IS-A" relationship between the classes.

IS-A:

The child contains all the features of the parent and we can see a child as a substitution of the parent as he poses everything of his parent.

(or)

A child can be expressed in terms of the parent.

Composition:

Within a class we declare other class as an attribute, instantiate the object of another class. and through reference class we are going to call the methods in reusing the functionality which is called "Composition".

The word "Composition", made of multiple parts inside it. A class has completed its functionality by keeping multiple class references within it thus being called as "Composition" here.

Composition establishes "HAS-A" relationship between the classes.

When do we need to use Inheritance and when we should use composition?

Inheritance:

There are 2 thumb rules on which we need to choose Inheritance

1. In our class we want to reuse all the methods of another class

2. our class can be seen as a substitute of another class

then we need to go for inheritance

Composition:

If we want to use few of the methods of another class, then go for composition only.

(or)

if we want to partial use the functionality of another class within our class then use Composition

Strategy Design pattern is recommending us to use Composition, rather than Inheritance most of the time because there are few challenges we will encounter if we use Inheritance, to avoid them it is recommending us to use Composition.

What are the challenges or difficulties of using Inheritance?

Problems or challenges of using Inheritance:

1. In most of the real-time use cases, a class wants to use few of the functionalities of another class, but not all of them. In such case it is recommended to use Composition only than inheritance. By which we can understand most of the real-time use cases are solvable through composition rather than inheritance.

2. Most of the programming languages including java doesn't support multiple inheritance. In a case where if your class wants to reuse the functionality of multiple other classes the only way we can accomplish is through Composition, we cannot use Inheritance because of the above limitation.

There are 3 principles of Stragey Design Pattern

1. Favor composition over inheritance

2. always design to interfaces, never design to concreate classes

3. code should be open for extension and closed for modification

#1 Favor composition over inheritance

If we want to reuse the functionality of another class in our class, use composition rather than inheritance.

When we should use Inheritance and when we need to use Composition?

Inheritance:

If we want to reuse all the traits of another class, in our class

If child can expressed interms of parent

Composition:

If we want to reuse few of the behaviours of another class in our class

There are certain problems we run into if we use Inheritance, to overcome them Strategy Design Pattern is suggesting us to use Composition only.

Problems in using Inheritance:

1. Most of the realtime usecases are solvable through composition rather than inheritance because many of the times a class want to reuse partially the functionality of another class only.

2. There are cases where even we want to use Inheritance also we cannot and should gofor composition only. If a class wants to reuse the functionality of multiple other classes we cannot, because most of the programming languages including java doesnt support multiple inheritance. The only way to reuse the functionality of multiple other clases is through Composition.

3. The classes will be come fragile if we go for inheritance, if we use composition those are less fragile.

```
class A {
  int m2(int i) {
    // operation
    return 23;
  }
}

class B extends A {
  int m2(int i) {
```

```
    int j = super.m2(i);
    // perform some additional operation
    return 24;
  }
}
```

A a = new B();

always to the parent class reference variables we can assign any of the child class objects, through which we can achieve runtime polymorphism.

int k = a.m2(10);

In general java determines the method to be called on a class based on the reference type we are using in calling the method.

But whenever we assign a derived class object to a base class reference variable, java will not call the method based on the reference type, rather it checks to see the reference variable is pointing to which class object and calls the method on that corresponding class, this is called "Runtime Polymorphism". Here the method to be called on which class whether it is "A" or "B" class is determined only at runtime because objects are instantiated and assigned to variables at runtime only.

Without modifying the code inside m2() method of class A, we can replace the logic or method m2() of class A through overriding. If we see even though we are calling m2() method using reference variable A still the method of B class m2(){} method is called, which looks like we are invoking new m2() using variable A (indirectly means replacements);

```
class A {
  double m2(int i) {
    // operation
    return 23.23;
  }
}
```

```
class B extends A {
  int m2(int i) {
    int j = super.m2(i);
    // perform some additional operation
    return 24;
  }
}

class C {
  void m3(int j) {
    B b = new B();
    int k = b.m2(j);
  }
}
```

Strategy Design Pattern

1. Favor composition over inheritance

2. always design to interfaces, never design to concreate classes

3. code should be open for extension and should be closed for modification

#1 Favor composition over inheritance

When we are reusing the functionality of another class, preferrably use composition rather than inheritance.

Why Strategy Design Pattern is suggesting us to use Composition than Inheritance?

There are problems with using Inheritance are there due to which it is recommending us to use Composition

1. Most of the realtime usecases are solvable through composition rather than inheritance, because many of times a class wants to use partially the functionality of another class.

2. Many programming languages including java doesnt support multiple inheritance, so in case if our class wants to reuse the functionality of multiple classes, we need to use composition only

3. code will be fragile when we use inheritance, and would be less fragile when we go for composition

Fragile: quickly/easily breakable (or) delicate to handle

If we use inheritance to reuse the functionality of another class, then our classes becomes highly fragile.

How to modify the functionality of a method, without changing the code inside the class?

Through method overriding, so that we can replace or enhance the logic of an existing method by overriding in a subclass. by using runtime polymorphism we can invoke the new method with parent class reference variable which is nothing but replacing.

```
class A {
  double m2(int i) {
    // logic
    return 39.1;
  }
}

class B extends A {
  double m2(int i) {
    int j = super.m2(i);
    // add some additional logic
    return 40;
  }
}

class C {
  public void m3(int i) {
    B b = new B();
```

```
    double j = b.m2(i);
    // do some operation
    sop(j);
  }
}

class Test {
  public static void main(String[] args) {
    C c = new C();
    c.m3(10);
  }
}
```

What is overriding?

overriding happens between the classes which are under inheritance relationship. A method to participate in overriding, the name of the super class method and subclass method should be same including parameters and returnType.

What is overloading?

overloading may happen within the classes or across the classes of inheritance hierarchy. A method is said to be overloaded if the method name should be same, but parameters should be different between those 2 methods. but it doesnt consider returnType in overloading.

```
class A {
  int m2(int i) {
    return 10;
  }
  int m2(int i, int j) {
    return 20;
  }
```

```
  double m2(int i) {


  }
}

A a = new A();
a.m2(10);
```

The above problem can be solved through composition, let us see.

```
class A {
  double m2(int i) {
    // logic
    return 24;
  }
}

class B {
  A a;
  int m2(int i) {
    int j = 0;

    a = new A();
    j = (int) a.m2(i);
    // logic
    return j;
  }
}
```

```
class C {
  void m3(int i) {
    B b = new B();
    int k = b.m2(i);
    sop(k);
  }
}
```