**OOPP concepts where you used in your project**

Yeah, we had used all Oops concepts in our project.

**The main oops concepts in java is**

* **Abstraction,**
* **Encapsulation,**
* **Polymorphism and**
* **Inheritance.**

**coming to abstraction,** hiding the background details of a particular object is known as abstraction.

Abstraction can be achieved in two ways by using

* **Abstract class and**
* **Interface.**

**Coming to Abstract class,**

Abstract class contains both concreate and abstract methods . concreate methods are implemented in with in the same abstract class , where as abstract methods are implemented in the child class using extends keyword .

**Coming to Interface,**

We can use implements keyword to implement the methods of an Interface. So, like that all abstract methods will be implemented in child class.

**In both the cases of abstract class or interface, we can not create an object**, just we can create reference object only and those abstract methods implementation will be referred and executed from the child class by using reference object.

**We can use abstract class ,** When we have requirement like where few set of functionalities common across all the subclasses and few functionalities are vary from subclasses, then the common functionality we can define in the abstract class. . In such cases we can go with the Abstract class and supports partial abstraction like that

**In our project , we have different types of customers and common functionality like contact or personal information implemented in concreate methods and unique functionality is implemented through abstract methods in child class**

**We can use Interface ,**

If we have requirement like , where all set of functionalities to be implemented every product or entity specific , then all abstract methods of interface is implemented according to the features of that product. In such cases we can use Interface and supports full abstraction like that

**In our project policy product interface methods are implemented for different types of policy products & also while customizing testng emailable reports , we had implemented Itest listener interfacre methods (OnStart,On Test Failure, On Test Succes ) abstract methods implemented in Listener class**

* **Coming to Encapsulation,**

Encapsulation is nothing but , wrapping of data and member functions within the same class

We can create a java class entity with setter and getter methods and we can initialize the object values securely, Like that we can achieve encapsulation in java

In our project, while creating contact, address entities object initialization, we did by using setters and getter methods, like that we did test data creation by using encapsulation technique

* **Coming to Polymorphism**

Polymorphism refers to writing a same method name implementation in multiple ways

* Coming to compile time polymorphism, writing a same method. With the different input arguments and within the same class implementation is known as compile time polymorphism. We can also call it as a Method Over loading

In our project to implement search centre functionality. We had used compile time polymorphism. The search centre method had been implemented with various input arguments. So that list view record details are filtered and displayed based on the search centre input parameters. Like that we had used compile time polymorphism in our project.

* Coming to Run time polymorphism, The same method is implemented in parent and child classes . The child class object overrides the parent class method during runtime. that is known as runtime polymorphism. We can also call it as Method Over riding

To customize the testing emailable reports. I test listener interface abstract methods are implemented. in the child classes. So in this case we had used method overriding concept. Like that we had used runtime polymorphism in our project.

* Coming to Inheritance

Sharing the properties of a class from one class to another. That is known as inheritance in Java.

* We can achieve inheritance by using extends and implements keywords.
* Extends key, supports single inheritance. Multi level inheritance ,Hierarchical inheritance
* Interface supports multiple inheritance and hybrid inheritance. A class can able to implement multiple interfaces by using implements keyword in Java. So like that we can able to achieve multiple inheritance in Java.

While extending base class driver , we had used extends key word

While sharing address or contact entity methods to another classes , we had used inheritance in java

So like that we used abstraction , encapsulation, polymorphism and inheritance in java