Problem 2:-

```
interface vehicle{
Int set_num_of_wheels();
Int set_num_of_passengers();
Boolean has_gas();
}
```

Answer a:- I can create Car and plane class that implement vehicle interface and have to implement all of the interface methods. So our car class will be like

```
Public class Car implements Vehicle{
Int wheels = 4;
Int passengers = 4;
Boolean gas = true;
@override
Public int set_num_of_wheels(){
Return this.wheels;
}
@override
Public int set_num_of_passengers (){
Return this.passengers;
}
@override
Public boolean has_gas (){
Return this.gas;
}
```

Our plane class will be as well

```
Public class Plane implements Vehicle{
Int wheels = 10;
Int passengers = 50;
Boolean gas = false;
@override
Public int set_num_of_wheels(){
Return this.wheels;
}
@override
Public int set_num_of_passengers (){
Return this.passengers;
}
@override
Public boolean has_gas (){
Return this.gas;
}
}
```

Now if we define main class

```
Public static void main (String [] args){
Vehicle car = new Car();
Vehicle plane = new Plane();
Car. set_num_of_wheels();
Car. set_num_of_passengers ();
Car. has_gas ();
plane. set_num_of_wheels();
```

plane. set_num_of_passengers ();

plane. has_gas ();

}

Answer b:- We can use abstract class instead of interface but for abstract class we can add others method and can implement there but for abstract method child class must implement abstract method.

Public abstract class vehicle{

Sub class should implement below methods

Public abstract Int set_num_of_wheels();

Public abstract Int set_num_of_passengers();

Public abstract has_gas();

//no need to implement this method

Public void Boolean has_engine(){

}

The car and plane class should implement abstract methods by extending Vehicle class.

}

Problem 3:-

I will use MVP model view presenter pattern for this video application.

Our model class will hold the list of videos. These videos can be got from local db or api hit. We will use a Video Manager class that will be connected with Video Model class and also with local db implementation or api calling class. We will call an interface here for video list parsing if it is complete or not. These classes will be in separate package.

Now we will use android activity class as presenter where we will implement an interface to be notified about video parsing. We will implement business logic here. When video file/url will be gotten we will send this video to view/UI to load. We will add click listener into activity to capture play,forward,rewind button. And for play forward and rewind we will use android videoview class which implements mediaplayer control to play video or forward video to specific position or rewind video.

And lastly the UI/view will hold only the videoview class xml tag where we will send the video url to load the video.

So the presenter -> the activity class will make bridge between model and view class here in our application.