Some fundamental facts and principles of programming in ArchStudio 4 are listed below. You need to understand them before I talk about some details.

1. Each architecture component can have *provided interfaces* (i.e. inward interfaces) and *required interfaces* (i.e. outward interfaces). Every interface has an interface type, which is linked to a Java interface file. In other words, every component interface can be seen as a Java interface consisting of a set of semantically-related operations.

2. Every component must implement all the operations included in its *provided* interfaces. Meanwhile, it can use or call the operations included in its required interfaces.

3. In the version of ArchStudio that we are using, the implementation of every component has at least two classes: architecture-prescribed code and user-defined code. The architecture-prescribed code is automatically generated and should never be manually edited. You can only change user-defined code. Of course, you can also create new classes if necessary. More detailed introduction of architecture-prescribed code is included below.

4. Every architecture component in ArchStudio is implemented based on the myx.fw framework. The only thing you need to know about the myx.fw framework is that it provides four lifecycle methods for each component to override: init(), begin(), end(), and destroy(). These methods can include the code that you want to be executed when a specific component is initialized, attached, detached, or destroyed respectively.

Next, I am going to briefly introduce some important elements in the architecture-prescribed code of a component. I will use the chatting application that we built in our lab as an example.
If you look at the architecture-prescribed code (i.e. ChatClientArch.java) of Chat Client, the first thing you will notice is the reference to the required interface. In the following code, the variable `OUT_IChat` is the reference to the required interface `IChat`. It is initialized in the begin() method.

Note that the definition and the initialization of references to the required interfaces are automatically generated. You do not need to manually change anything here.

```java
//The names of the associated interfaces
public static final IMyxName msg_IChat = MyxUtils.createName("edu.sce.chatsys.IChat");
public static final IMyxName msg_IChatListener = MyxUtils.createName("edu.sce.chatsys.IChatListener");

//Reference to the required interface
public IChat OUT_IChat;
```

As mentioned earlier, you will need to call the operations included in the required interface at some point in your application. This is done via the reference to a required interface. For example, look at Line 90 of ChatClientImp.java (i.e. the user-defined code).

```java
_arch.OUT_IChat.sendMessage(getTitle(), text);
```

In the above code, `_arch` is a reference maintained in the user-defined code that points to the corresponding architecture-prescribed code.

Similarly, the architecture-prescribed code of each component also maintains a reference to its user-defined code, `_imp`. The following code is from ChatClientArch.java. Again, it is automatically generated.

```java
//Reference to the user-defined code
private IChatClientImp _imp;
```
The architecture-prescribed code also contains the implementation of all the operations included in its provided interfaces. In ChatClientArch.java, there is only one method included in its provided interface.

```java
public void messageSent (String sender,String message) {
    _imp.messageSent(sender,message);
}
```

Note that the code above actually implements the method simply by redirecting the request to the user-defined code. This is done via the reference, `_imp` mentioned above. In other words, all the real work or specific implementations (including the implementation of the myx lifecycle methods) are actually done in user-defined code.

Finally, there is an interface file between architecture-prescribed code and user-defined code of every component. It is also automatically generated and includes all the operations that the architecture-prescribed code expects the user-defined code to implement. In ChatClientImp.java, you will see the following code at the beginning.

```java
public class ChatClientImp extends JFrame implements ActionListener, IChatClientImp

IChatClientImp above is just the name of the interface between ChatClientArch and ChatClientImp.

I hope this document helps. I suggest that you use it as a reference and examine the chatting application carefully to understand programming in ArchStudio.