The integration of the other program parts by implementation of interfaces was not possible, because no interfaces were available.

To avoid the problem, our adapter classes inherit direct from the domain objects from the other team.

Also every adapter implements a generic adapter interface, to set and read its type.

This enables dynamic programming, there every of our adapters support this interface and so they can be read and created by other classes.

\*click\*

Here we see the class diagram of our adaption +blah+

\*click\*

The SWT-GUI from Team F was not easy to integrate into our Swing-GUI, because they are 2 different toolkits. Fortunately the user interfaces from Team F was not extensive, so we replaced the SWT Message Box with a Swing counterpart.

The payment on account booking was implemented by the Accommodation -Panel from Team F, due to a conflict with the SWT-Panel integration into a Swing application.

Java don’t allow multiple inheritance. This prevents us to inherit from other adapters. So we override the methods from the superclass of the adapted class.

\*click\*

To realize the resevaration homepage we uses JSF.

It has several advantages compared to jsp

The first big advantage is that event handling is supported. This makes it easy to react on user input.

We have used this for the add room and delete room buttons

\*click\*

Another big benefit of JSF is that it automatically binds bean properties to user interface components.

This can be seen at most of our interface components

\*click\*

Since version 2 Ajax is integrated into JSF.

This means explicit programming of it is not necessary. You don’t even had to care about it on the server this is handled automatically

We have use this feature for rendering parts of the page when we add or delete a room from the reservation

\*click\*

Also a validation a conversion of input is integrated into JSF.

We use this with every of our input fields

\*click\*

JSF provides special tags that are converted to HTML code. So there is no need to write HTML by oneself and the programmer can concentrate on the main part of his job.

There are also a lot of third party libraries which extends this functionality.

A good example where we’ve used this is the change user data Table

\*click\*

JSF provides full-fledged page templating system that let one build pages that share layout and contet.

Because of this we could concentrate to the content and were not held up with the layout

\*click\*

Last but not least, JSF encourages the use of the model view controller principle

\*click\*

This picture shows how this works.