

Information Systems Analysis & Design (M8748)

Tutorial 17 Answer

1. Why should the user interface classes be kept separate from the business classes and application logic?

Objects may be used in different use cases with different display requirements. It may not be appropriate to try to build a standard layout into the class definition, so interface classes are used to provide views of the object.

2. Explain the difference between vertical and horizontal prototyping.

A horizontal prototype deals with only one layer of the system architecture, usually the user interface. A vertical prototype takes one sub-system and develops it through each layer.

3. What is meant by a throwaway prototype?

A throwaway prototype is one that is developed to test out ideas about the interface design, but is not then developed to become the working system. Often it is developed in a language that allows it to be developed quickly rather than the language in which the full application will be developed.

4. What does the «import» stereotype mean?

It means that classes in one package have a dependency on classes in another package and will need to 'import' the interface to those classes in order to use them. The exact mechanism depends on the language used, e.g. 'import' in Java or 'using' in C#.

5. What role does each element of the Model-View-Controller architecture play?

Model classes provide the main functionality of the system.

View classes provide a display of the attributes of objects to the user.

Controller classes respond to interaction from the user.

6. What else do we use statechart diagrams for, apart from modeling the state of interface objects?

We use statechart diagrams to model the lifetime of instances of business classes.

7. What are the five steps in preparing a statechart to model a user interface?

Describe the high-level requirements and main user tasks.

Describe the user interface behavior.

Define user interface rules.

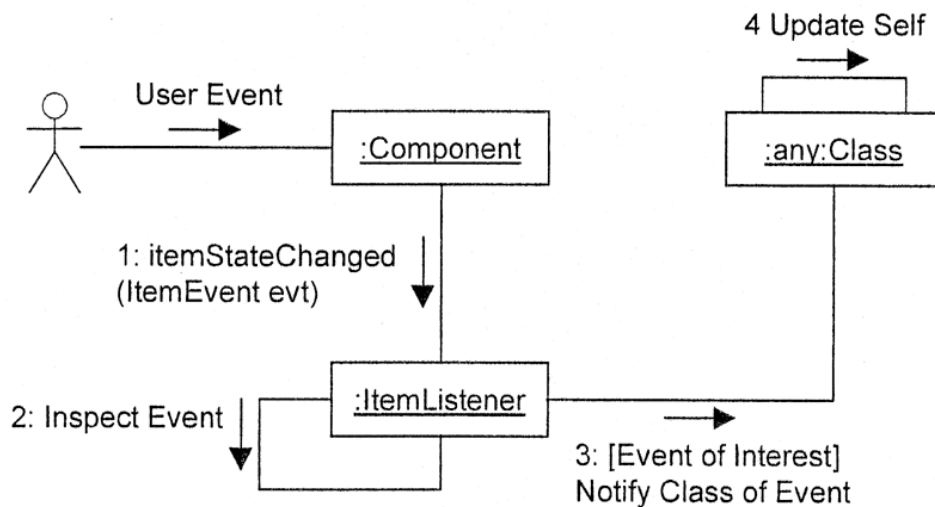
Draw the statechart (and successively refine it).

Prepare an event-action table.

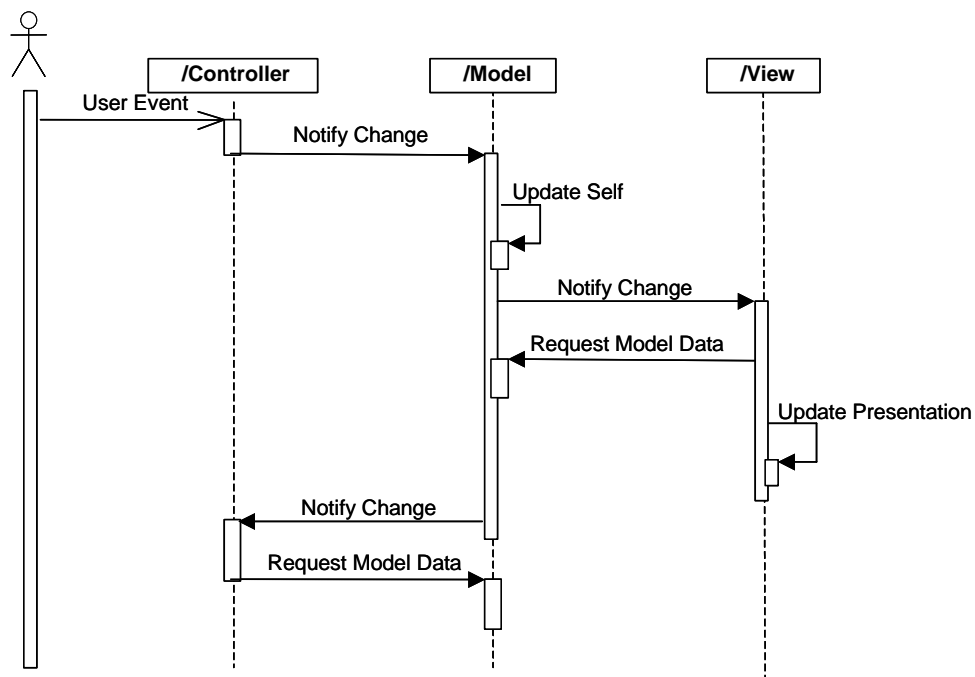
8. What information is held in an Event-Action table?

A list of states, for each state the valid events that can cause a transition from that state, the state that each transition leads to, and any operations associated with the transition into the new state.

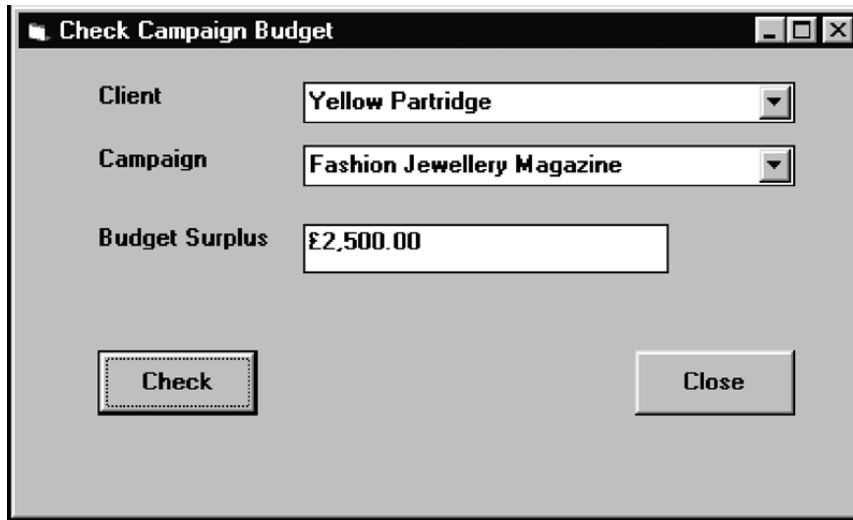
9. Convert the collaboration diagram of Figure 17.28 into a sequence diagram.



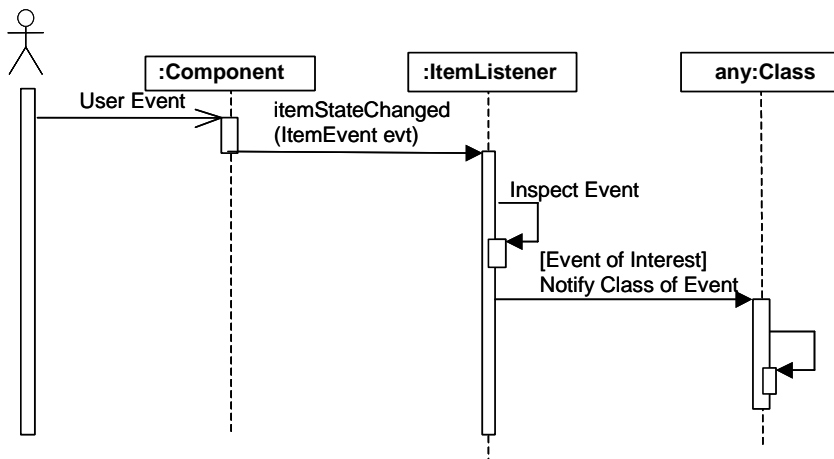
There is an error in Figure 17.28. The message '6. Update Presentation' should be number 5, with the other messages numbered 5 and 6 becoming 6 and 7.



10. Convert the collaboration diagram of Figure 17.29 into a sequence diagram.



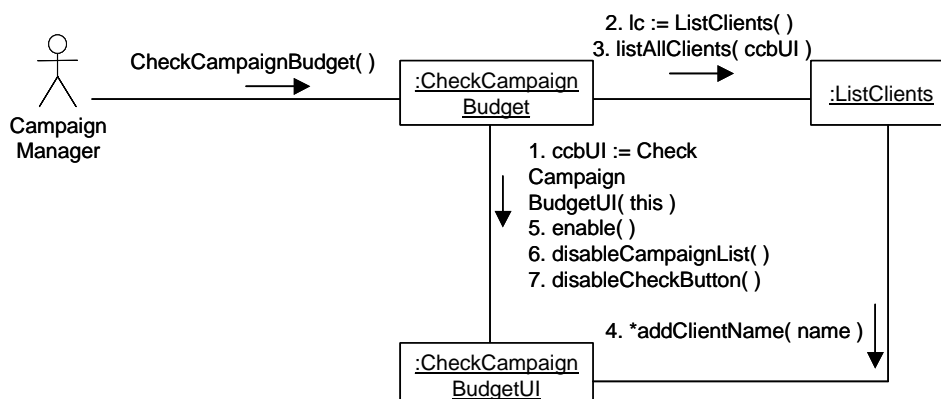
There should not be an initial colon on 'any:Class'.



11. What are the differences between the MVC and Java EventListener approaches?

Java EventListener only handles changes to interface objects. MVC deals with changes to Model objects. The Java Observer and Observable interfaces provide MVC mechanisms.

12. Convert the sequence diagram of Figure 17.38 into a collaboration diagram.



13. Convert the sequence diagram of Figure 17.39 into a collaboration diagram.

