ERD/UML GENERATOR

HELLO USER,

Welcome to HELP!!

Our application "**ERD/UML GENERATOR**" provides you the ease to draw the entity relationship diagrams/UML diagrams/sequence diagrams. Here we will provide you the basic knowledge of what and how the diagrams are!!

To know how to draw the diagrams neatly and correctly please refer to the help provided in persistent diagram section set!! We have told you the tool which are used in drawing the particular diagram.

You can refer to the examples given below of each diagram after learning about the diagrams and the tools provided by us !!

Use-case diagram

A use case illustrates a unit of functionality provided by the system. The main purpose of the use-case diagram is to help development teams visualize the functional requirements of a system, including the relationship of "actors" (human beings who will interact with the system) to essential processes, as well as the relationships among different use cases. Use-case diagrams generally show groups of use cases -- either all use cases for the complete system, or a breakout of a particular group of use cases with related functionality (e.g., all security administration related use cases). To show a use case on a use-case diagram, you draw an oval in the middle of the diagram and put the name of the use case in the centre of, or below, the oval. To draw an actor (indicating a system user) on a use-case diagram, you draw a stick person to the left or right of your diagram (and just in case you're wondering, some people draw prettier stick people than others). Use simple lines to depict relationships between actors and use cases, as shown in Figure 1.

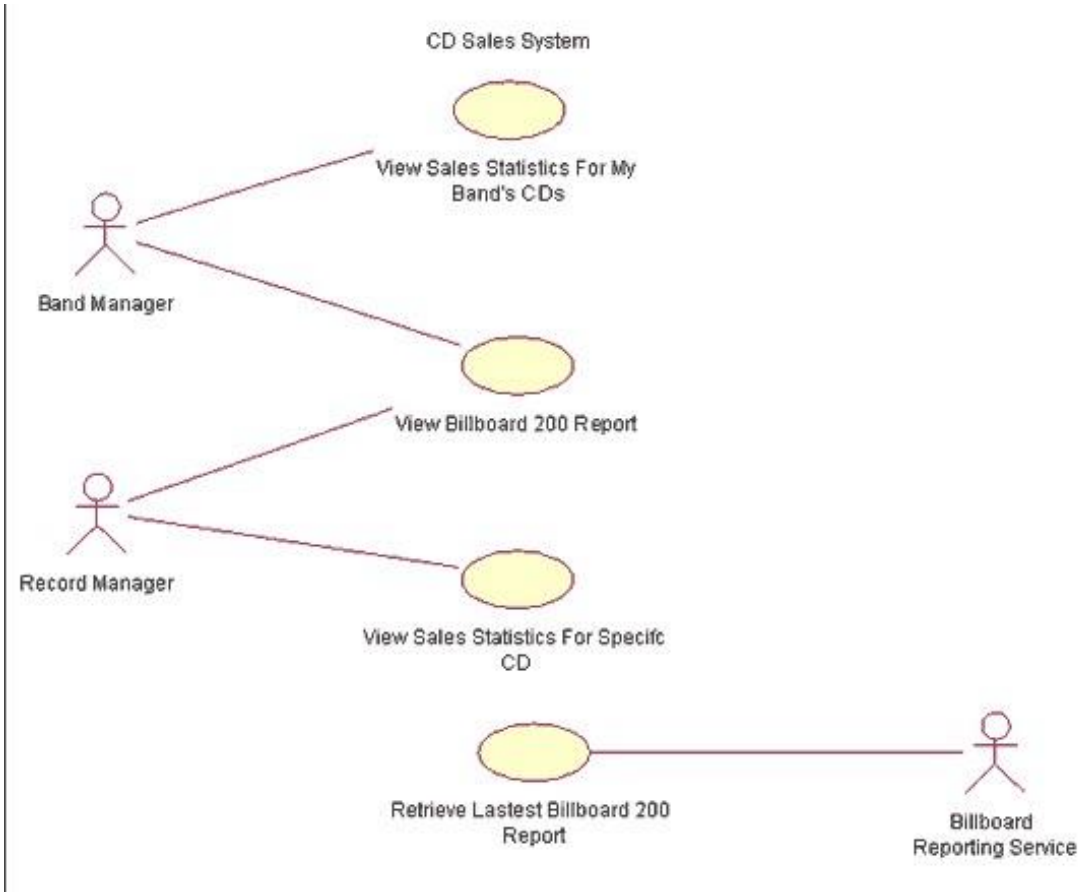


Figure 1

Sequence diagram

Sequence diagrams show a detailed flow for a specific use case or even just part of a specific use case. They are almost self explanatory; they show the calls between the different objects in their sequence and can show, at a detailed level, different calls to different objects.

A sequence diagram has two dimensions: The vertical dimension shows the sequence of messages/calls in the time order that they occur; the horizontal dimension shows the object instances to which the messages are sent.

A sequence diagram is very simple to draw. Across the top of your diagram, identify the class instances (objects) by putting each class instance inside a box (see Figure 4). In the box, put the class instance name and class name separated by a space/colon/space " : " (e.g., myReportGenerator : ReportGenerator). If a class instance sends a message to another class instance, draw a line with an open arrowhead pointing to the receiving class instance; place the name of the message/method above the line. Optionally, for important messages, you can draw a dotted line with an arrowhead pointing back to the originating class instance; label the

return value above the dotted line. Personally, I always like to include the return value lines because I find the extra details make it easier to read.

Reading a sequence diagram is very simple. Start at the top left corner with the "driver" class instance that starts the sequence. Then follow each message down the diagram. Remember: Even though the example sequence diagram in Figure 4 shows a return message for each sent message, this is optional.

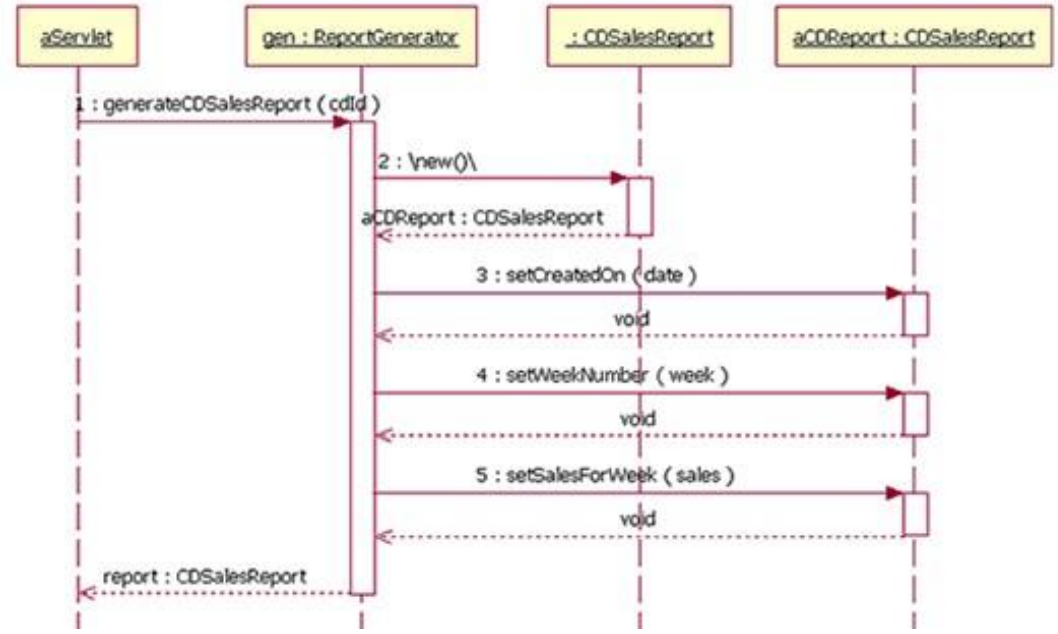
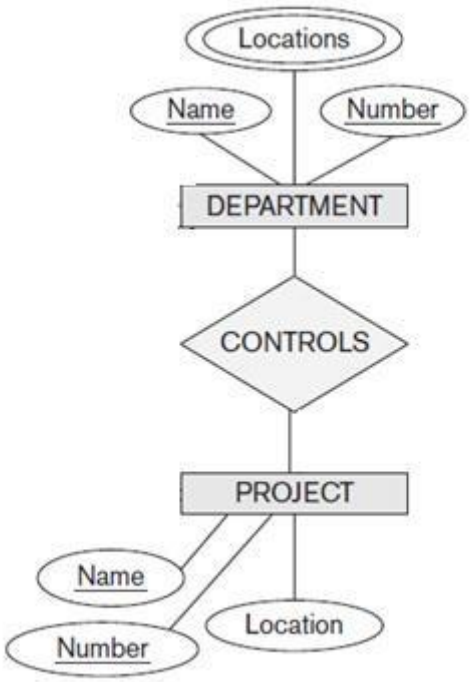


Figure 2

ER DIAGRAMS

ER model describes data in terms of:

- Entities and entity sets
- Objects
- Relationships and relationship sets
- Connections between objects
- Attributes
- Properties that characterize or describe entities or relationships.



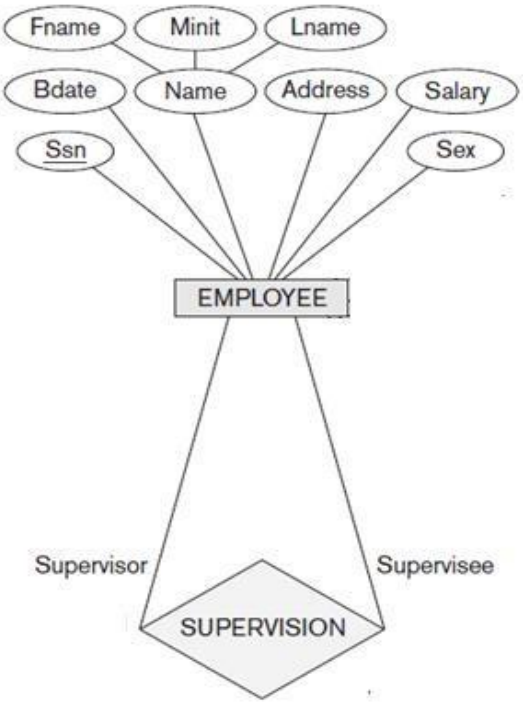


FIGURE 3 AND 4