

Question1:

a)b)

Classes:

The purpose of a class is to specify a classification and the Features of objects and to define their structure and behavior. The structure and behavior of objects is characterized by classes, and classes can act as namespaces for other classes defined within their scope. A class is shown using the Classifier symbol : the default notation for a classifier is a solid outlined rectangle with the classifier's name, with compartments separated by horizontal lines below the name, the name should be centered and in boldface, and should begin with an uppercase letter. If the class is abstract it is written in italics. Compartments can be suppressed and then separator lines aren't drawn and no inference may be drawn from the presence or absence of the elements in the suppressed compartments. A Class has four mandatory compartments: attributes operations receptions and internal structure (and these must appear in this order). If the class represents a meta class may be extended with <<Metaclass>> before its name. And a class can have optional compartments that all classifiers can have. Any compartment that contains features may show these grouped under the strings: private public protected representing their visibility. A compartment's name may be shown or hidden it should be centered and start with lowercase letters, it can contain spaces but no punctuation. A class with the property isActive = true can be shown by a class box with an extra vertical line on each side. Any keywords including stereotype names can also be written centered in plain face in a pair of "<<" above the name of the class name if there are multiple keywords/stereotypes then each enclosed in a separate pair of << and listed one after the other or listed all in the same pair of << separated by commas.

(Source: <https://www.omg.org/spec/UML/2.5.1/PDF/> Page 237 + 101 + 102)

Objects: (Instancespecifications?)

InstanceSpecifications represent the possible or actual existence of an instance in a modeled system and completely or partially describe these. InstanceSpecifications represent instances of Classifiers and Classes are Classifieres in a modeled system. InstanceSpecifications are depicted like classifiers with the exception that in the place where the Classifier name is written, the name of the instance followed by a colon followed by the classifier names if any (and if there are multiple then these are separated by commas) is written and underlined. Names are Optional for Classifiers and InstanceSpecifications, the standard name for an unnamed Classifier or anonymous Instance Specification is an underlined colon, If the Instance Specification has Value specifications these are represented with: value name followed by "=" followed by a ValueSpecification in the enclosing shape. Structural features if any are shown with structural feature name followed by = followed by a value specification. There aren't any mandatory

parts except maybe that the box with the name need to be drawn.

(Source: <https://www.omg.org/spec/UML/2.5.1/PDF/> Page 127 + 128)

Attributes(Properties?):

Properties Represent attributes of classifiers, a memberEnd of an Association or both, they also represent attributes of Classes since they are classifiers. Properties are represented as a string consisting of the visibility followed by / followed by the name : followed by the property type followed by the multiplicity type in "[" brackets, followed by "=" followed by the default value followed by any modifiers in "{" brackets. In Classifiers the type visibility , default , multiplicity and property string may be suppressed from being displayed even if the values are in the model. Properties can be shown in columns rather than continuous strings. And an attribute may be shown using the association notation where the hollow or filled diamond is shown at the end of the tail. There aren't really any mandatory

(Source: <https://www.omg.org/spec/UML/2.5.1/PDF/> Page 113 + 114)

Operations:

An Operation is a Behavioral Feature of a Class, DataType or Interface. Operation Elements specify the name type, Parameters and Constraints for When These Operations are Invoked (Executed) on instance of their Featuring Classifiers. An Operation is a Feature of a Class and determines what its objects can do and how. Operations are shown with text strings in the form of: [<visibility>] <name> '[' [<parameter-list> ']' ['<return-type>'] ['<multiplicity-range> ']' ['<oper-property> ['<oper-property>']* ']]'. Operations probably must be shown in this form it seems everything is optional except the name + "()".

(Source: <https://www.omg.org/spec/UML/2.5.1/PDF/> Page)

Associations:

Associations specify semantic relationships between typed instances. An Association declares that there can be links between instances whose types conform to or implement the associated types. Associations specifies semantic relationships between typed instances so in other words it describes relationships between Objects and Objects of Classes. Associations Can be drawn as a diamond with a solid line for each association connection of the diamond to the classifier that is that ends type. A binary Association is drawn as a solid line connecting two Classifiers or a solid line connecting a single classifier to itself. At least a line required. The Association's name can be shown as a name string near the Association symbol a slash in front of the Association's name means it has is being derived. A property string may be placed near the Association symbol. On Binary Associations a solid triangular arrowhead can be drawn next to or in place of the name pointing in the direction of the line indicates that the object the arrow is pointing

away from should be read first. Generalizations between Associations can be shown with a generalization arrow between the association symbols. An Association end is the connection between the line of the association and the icon of the Classifier a name string can be placed here to show the name of the Association end. Other notations can also be placed near the end of the line like a multiplicity a visibility symbol or a prop modifier. open arrow head on the end indicates it is navigable ownership of association can be shown with a filled dot at the end of the line. Qualifiers if there are any are mandatory parts so must be shown, qualifiers are small boxes at the end of Associations with attributes drawn within.

(Source: <https://www.omg.org/spec/UML/2.5.1/PDF/> Page 126 - 128)

Multiplicity:

A Multiplicity Element is an Element that can be instantiated to represent a collection of values depending on the type of element. Associations can have multiplicities. The notation is specified for each type of Multiplicity element but in general a notation will be shown as a text string containing the bounds of the multiplicity and a notation for showing the optional ordering and uniqueness specifications. It can also be displayed as a symbol. A multiplicity can be shown in the format <lower-bound>..<upper-bound> .Just at least Some notation about a bound is required if there is one. Further Optional Components depend on the specific Multiplicity element.

(Source: <https://www.omg.org/spec/UML/2.5.1/PDF/> Page 33 - 35)

Generalization (Generalization Sets?):

Generalization Elements depict a relationship in which a (child) element is based on another (parent) element. This is used to indicate that for example: in class diagrams child classes receive all attributes operations and relationships defined in their parent class. A generalization Relationship is represented as a solid line with a hollow arrowhead that points from the child element to the parent element placed at the end of the line at the parent element. At least this is required it seems. The lines can be named to designate Generalization Sets with names written near the line, two lines to the same parent can be joined so that they are part of the same generalisation set.

(Sources:

<https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-generalization-relationships>)

Komposition(Components):

A Composition Association Relationship represents a relationship where An element is

comprised of another, and that the life time of the part classifier is dependent on that of the whole classifier. Classifiers can be composed of other classifiers so Classes can be composed of other classes so they would have such a relationship depicted between them. A composition association relationship depicted with a solid line with and a filled diamond at the association/end of the line which is counted to the whole/composite classifier.

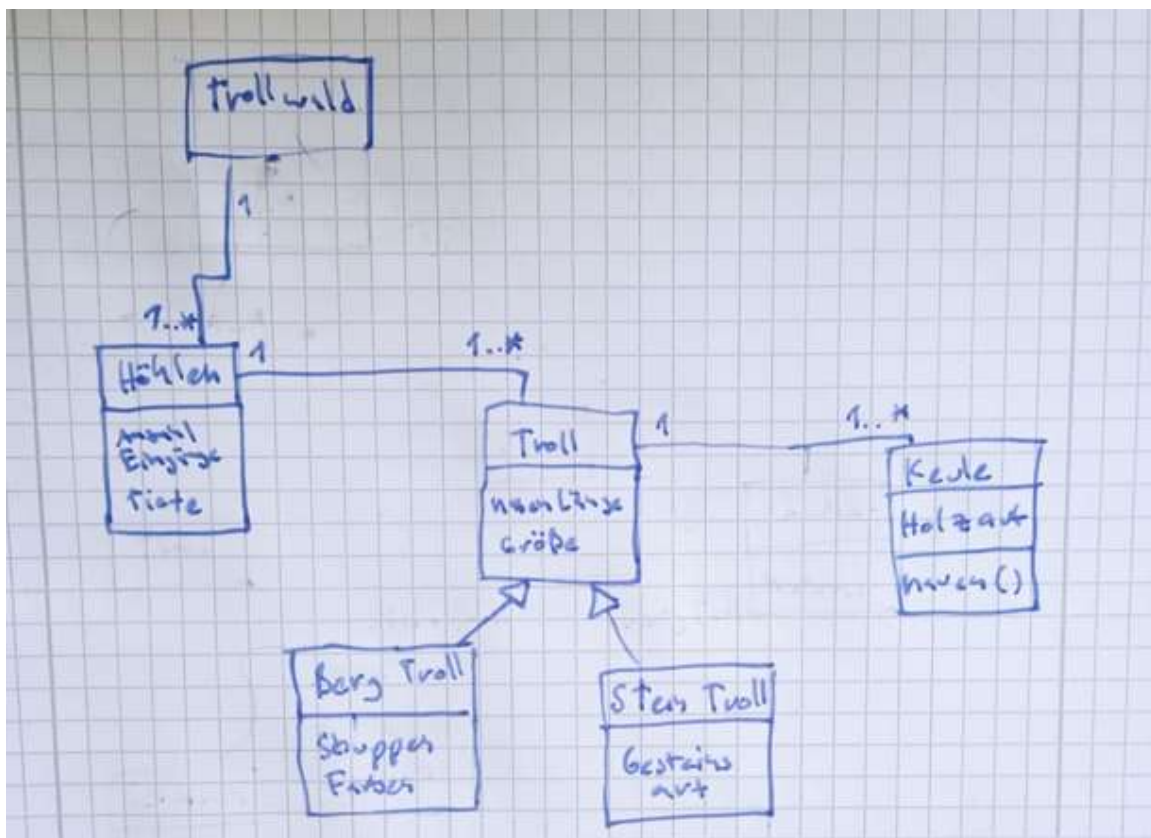
(Source: <https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-composition-association-relationships>, <https://www.omg.org/spec/UML/2.5.1/PDF/> Page 209)

Aggregation:

Used to show a relationship between classifiers where one is part or subordinate to the other. Can be used with classes to show this type of relationship. The representation of an aggregation is: a solid line with an unfilled diamond at the association end which is connected to the classifier that represents the aggregate.

(Source: <https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-aggregation-relationships>)

Aufgabe 3-2:



Aufgabe 3-3:

a) Objekt: (15.4 Object node???) InstanceSpecification 9.8, 9.9.9

Generalisierung: Generalization 9.7 , 9.9.7

Attribut: Property 9.9.17

Operation: Operation 9.6, 9.9.11

Komposition & Aggregation:?

Schnittstelle: Interface 10.4 , 10.5.5

Implementierung:Realization 7.8.14

Assoziation: Association 11.5

b)

Methodenaufruf : CallEvent 13.4.3

Interner Übergang: Transition but with kind set to internal 14.5.11.5 , 14.5.12 (TransitionKind), 14.5.12.3 (values)...

Externer Übergang: Transition 14.5.11

Aufgabe 3-4:

a) A sequenz diagram is a diagram that describes the interaction of a user with software over time and is useful for refining the model in the requirements analysis and to explain the behavior of the software to stakeholders.

b) keine?

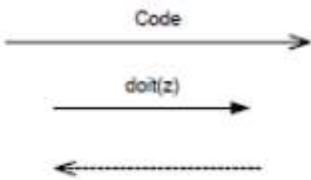

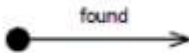
c) The lifelines of objects are displayed as vertical lines and time: the start of the interaction is at the top the end at the bottom?

d) The Function call of a Method is represented as a horizontal arrow with the method signature on it?

e) Yes, The Messages are the Method calls these point with arrows to what the results of their execution are.

- f) The relation consist of the fact that the sender object has created the reciever object.
- g) You can deduce this from the fact that there is only one line and that only one specific order of events is depicted.
- h)

<https://www.omg.org/spec/UML/2.5.1/PDF/>

Message	
LostMessage	
FoundMessage	
GeneralOrdering	