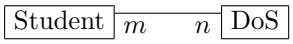
1

In a use case diagram for a system, an *actor* may be:

1. an object in that system
2. a user of that system
3. either of the above
4. none of the above

2

Each student has a lecturer as their Director of Studies (DoS). A DoS may direct any number of students. These facts are to be reflected in a software system. Suppose



is part of a class diagram of the system, where m and n are the *multiplicities*. Which of the following gives the best model?

1. m is 1 and n is 1
2. m is 1 and n is 0..*
3. m is 0..1 and n is 0..*
4. m is 0..* and n is 1
5. m is 0..* and n is 1..*

3

Consider again the diagram in the question above. Imagine that, in a run-time instance of a system which implements the diagram, `student` is an instance of class `Student` which is linked by a link of the association in the diagram to an object `dos` which is an instance of class `DoS`. Then:

1. Object `student` must at some time send a message to object `dos`
2. Object `dos` must contain a reference to object `student`.
3. At least one of 1 and 2 must be true.
4. Both 1 and 2 must be true.
5. Neither 1 nor 2 need be true.

4

A use case diagram shows an actor connected to a use case. Which of the following is true?

1. every scenario in the use case involves an instance of the actor
2. every scenario in the use case involves every instance of the actor

3. every scenario in the use case results in an outcome which has value for an instance of the actor
4. there is at least one scenario in the use case which involves an instance of the actor
5. there is at least one scenario in the use case which results in an outcome which has value for an instance of the actor

5

Consider the following code, extracted from the definition of a class GridBagLayout.

```
/**
   * Gets the constraints for the specified component.  A copy of
   * the actual <code>GridBagConstraints</code> object is returned.
   * @param       comp the component to be queried
   * @param       defaultConstraints the default constraints
   * @return      the constraint for the specified component in this
   *                  grid bag layout; a copy of the actual constraint
   *                  object is returned
   */
  public GridBagConstraints getConstraints(Component comp) {
      GridBagConstraints constraints = t.get(comp);
      if (constraints == null) { // if no object was returned
          setConstraints(comp, defaultConstraints); // use defaults
          constraints = t.get(comp); }
  return (GridBagConstraints)constraints.clone();
  }
```

(Explain the special notation used in the comment section. What tool will make use of it, and how will it be interpreted? Comment on any problems that will result if this fragment is used as it stands.)

(Criticise the quality of the code, and, where possible, suggest specific improvements. Your answer should include a very brief justification that it is worth bothering to make the improvements.)

Draw a sequence diagram to illustrate a scenario in which the method above is called by an actor `c:Client`, and the first call to `t.get` returns `null`.

6

UML contains many kinds of arrows. Write down as many as you can remember (without referring to the slides until you must!) with a note on what they mean and what kind of diagram they occur in.