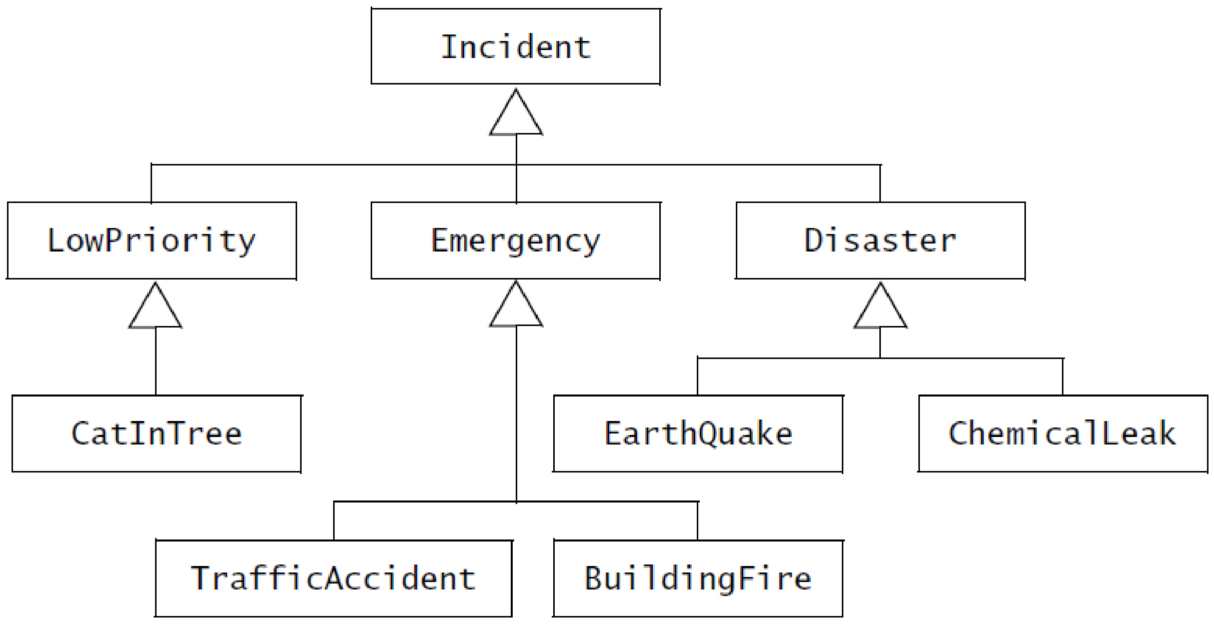
1. Please describe shortly what do you mean by stereotype in UML and give some examples of stereotypes. 1.5 pt

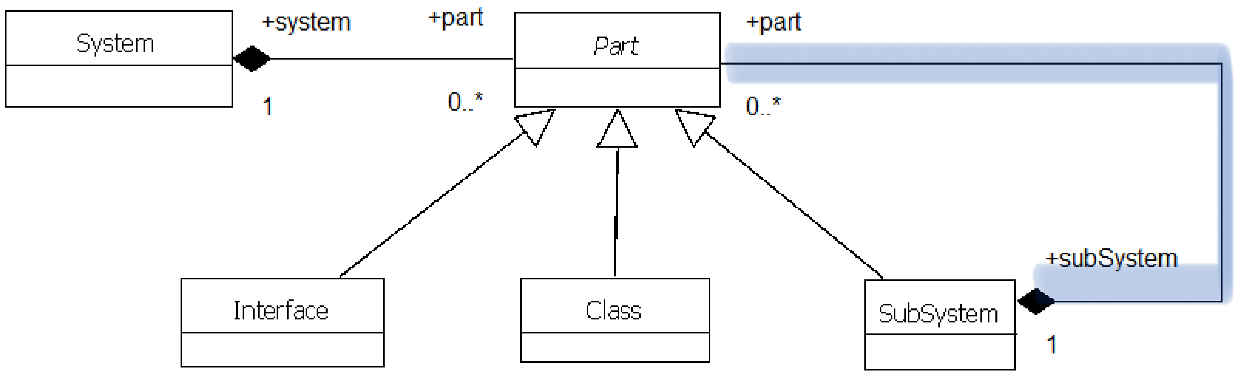
A **stereotype** is an extension mechanism that allows developers to classify model elements in UML. A stereotype is represented by string enclosed by guillemets (e.g., «boundary») and attached to the model element to which it applies, such as a class, association, operation a.s.o. Formally, attaching a stereotype to a model element is semantically equivalent to creating a new class in the UML meta-model (i.e., the model that represents the constructs of UML). This enables modelers to create new kinds of building blocks that are needed in their domain. For example, during analysis, we classify objects into three types: **entity**, **boundary**, and **control**. Entity, boundary, and control objects have the same structure (i.e., they have attributes, operations, and associations), but serve different purposes. The base UML language only includes one type of object. To represent these three types, we use the stereotypes «entity», «boundary», and «control» (Figure 2-49). The «entity», «boundary», and «control» stereotypes are described in Chapter 5, *Analysis*. Another example is the relationships among use cases. As we saw in Section 2.4.1, include relationships in use case diagrams are denoted with a dashed open arrow and the «include» stereotype. Similarly for exclude relationships.

2. Please analyze the architecture of the model represented below by means of a class diagram, mention and argue your point of view about the correctness of this diagram. What kind of inheritance relationship is represented in the diagram. 1 pt



The classes: **Incident**, **LowPriority**, **Emergency** and **Disaster** are **abstract classes.** In the diagram these are represented as concrete classes. **Abstract classes names have to be written in italics.**

3. Using the UML language, please represent the structure o a model that illustrate the structure of a system containing subsystems, classes and interfaces. At its round, each subsystem may contain other subsystems, classes and interfaces. Comment the architecture. 2.5pt



The architecture represented in figure above is the composite pattern.

4. Please describes the functionality of the Turnstile with Diagnostic Mode represented in the diagram below mentioning the type of the UML components used in this diagram and the kind of this diagram. What's the role of the components named NormalMode and DiagnosticMode? 2pt

It is about a State Transition Diagram (STD).

The UML concepts represented in this diagram are:
1. **composed states**: NormalMode and DiagnosticMode
2. **concrete states**: Violation, Locked, Unlocked, TestCoin and TestPassReady
3. **pseudo states**:
a. **2 input states** (one in each composed state) and,
b. **a history state** in NormalMode
4. **transitions: between two different** states or in the same state like those triggered by stimuli: Pass, Reset, Coin
5. **events (stimuli)**: Pass, Reset, Coin, Diagnose, Return, TestLock, TestUnlock, TestAlarm and TestResetAlarm
6. **actions**: Alarm, Lock, Unlock, Thankyou a.s.o.

After powering on the system, the action Lock will be executed and the system will be in the state Locked. In this state, the events that will trigger a transition are: Coin and Pass. Coin trigger the execution of Unlock action and the transition in the state Unlock. Pass will trigger the execution of the Alarm action and the transition in the state Violation. The behavior corresponding to the remained concrete states: Violation, Unlocked, TestCoin and TestPassReady is similar.
If the system is in the composed state NormalMode (that is in one of its inner concrete states) and a Diagnose event will appear, than, a transition will be executed: from the inner concrete state of NormalMode in the inner concrete state TestCoin of the composed state DiagnosticMode. The semantics of the auto-transitions specified for the composed state DiagnosticMode is that the system will remain in the concrete state in which it is and, the corresponding action of each event will be executed. If the system is in the composed state DiagnosticMode and the Return event will appear, than, the ReturnDeviceStates will be executed and the system will transit in the last concrete state of NormalMode in which the system was before to transit toward TestCoin. This happen due to the pseudostate History.

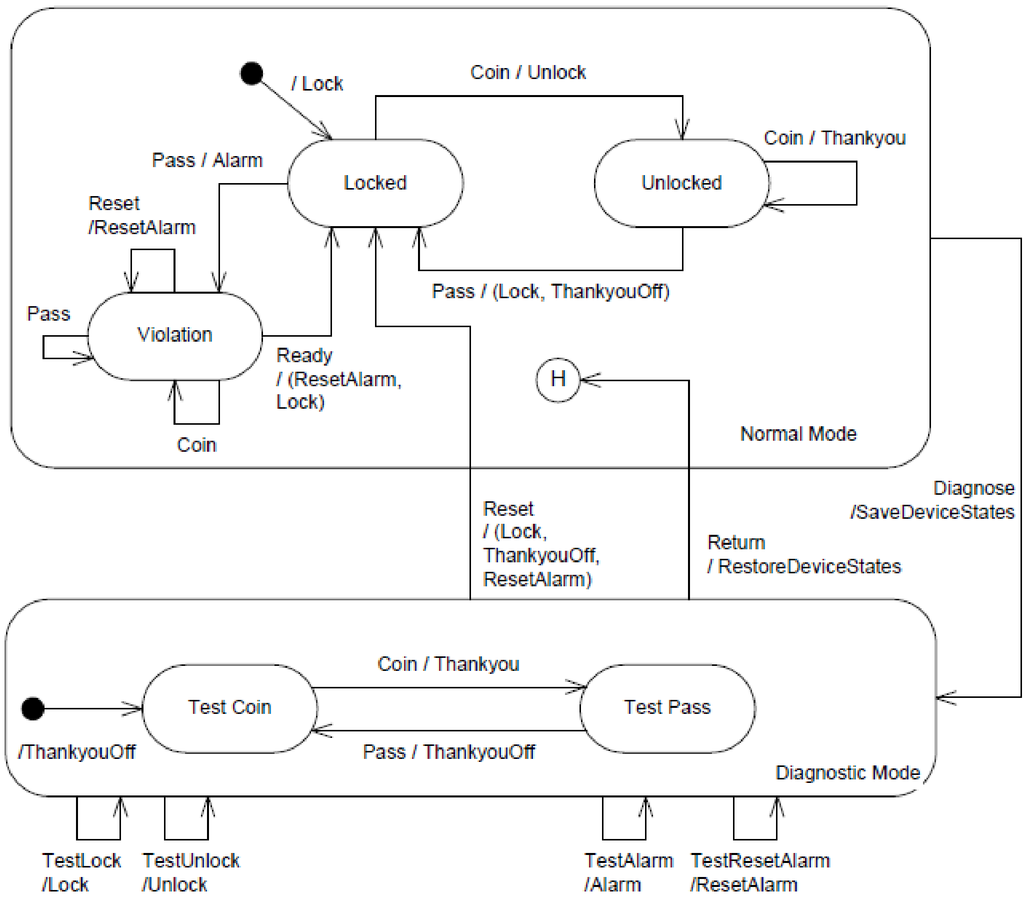
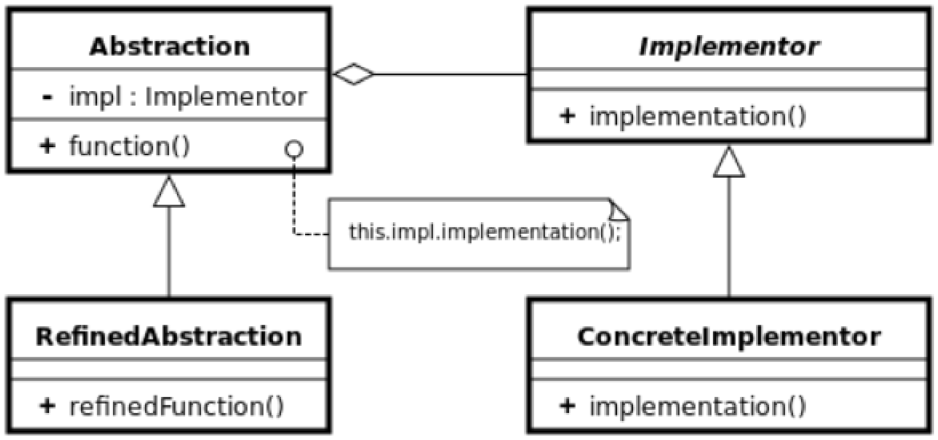


Figure 4: Turnstile with Diagnostic Mode.

5. Please mention if in the class diagram represented below there are some informational redundancies from the point of view of the UML specification. If the diagram represents a design pattern, please name it and describe how the pattern works. 2pt



As we may notice, in the Abstraction interface the impl attribute (containing a reference towards the *Implementor* interface) is redundant because this is represented graphically by means of an aggregation (composition in pattern description). The diagram represents the Bridge design pattern.

The Bridge pattern decouples abstraction from implementation so that both can vary independently. It has been achieved with delegation (composition) rather than by using inheritance.

The pattern contains four components.

Abstraction: defines an interface
RefinedAbstraction: implements abstraction:
Implementor: defines an abstract class (interface) for implementation
ConcreteImplementor: implements Implementor interface.
Two orthogonal class hierarchies using delegation (and no inheritance). The Abstraction and Implementation hierarchies can vary independently. Implementation never refers Abstraction. Abstraction contains Implementation interface as a member (through composition). This composition reduces one more level of inheritance hierarchy.