**UML**

**Participants Notes**

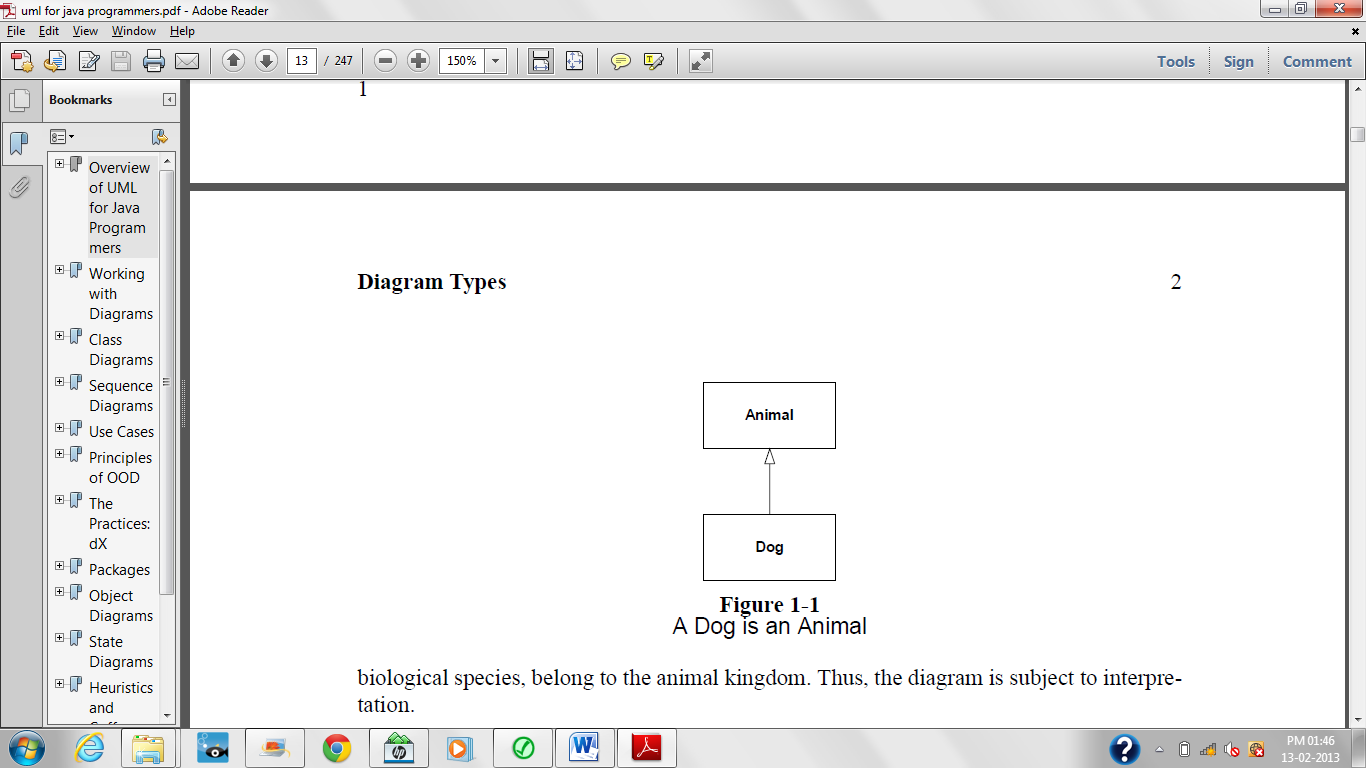
**What is UML?**

The Unified Modeling Language (UML) is a graphical notation for drawing diagrams of software concepts. One can use it for drawing diagrams of a problem domain, a proposed software design, or an already completed software implementation. Martin Fowler describes these three different levels as Conceptual, Specification, and Implementation. This deals with the last two.

* Industry standard mechanisms for visualizing, specifying, constructing and documenting software systems.
* Can capture various processes and structures related to business and software.
* Able to express static and dynamic structure.
* A project can rely on UML as the standard language to express requirements, system design, deployment instructions and code structure.

Specification and Implementation level diagrams have a strong connection to source code. Indeed it is the intent for a Specification level diagram to be turned into source code. Likewise it is the intent for an Implementation level diagram to describe existing source code. As such there are rules and semantics that diagrams at these levels must follow. Such diagrams have very little ambiguity, and a great deal of formality. On the other hand, diagrams at the Conceptual level are not strongly related to source code. Rather they are related to human language. They are a shorthand used describe concepts and abstractions that exist in the human problem domain. They don’t follow strong semantic rules and therefore their meaning can be ambiguous and subject to interpretation.

Consider, for example, the following sentence: A dog is an animal. We can create a Conceptual UML diagram that represents this sentence.



This diagram depicts two entities named Animal and Dog connected by generalization relationship. An Animal is a generalization of a Dog. A Dog is a special case of an Animal. That’s all the diagram means. Nothing more can be inferred from it. We might be asserting that our pet dog, Sparky, is an animal; or we might be asserting that dogs, as abiological species, belong to the animal kingdom. Thus, the diagram is subject to interpretation.

**Goals of UML:**

* Provide users with a ready-to-user, expressive visual modelling language so they can develop and exchange meaningful models,
* Be independent of particular programming languages and development processes.
* Provide a formal basis for understanding the modeling language.

**Who invented UML?**

* Grady Booch
* Rumbaugh
* Jacobson
* In 1996
* UML is a part of OMG (Object Management Group)

**Benefits of UML:**

* Modeling makes complex and huge system to break up in to a simple and descrete pieces that can be individually understood.
* Example: Simple flowchart drawing is a modeling.
* Other benefits:
  + Readability
  + Reusability
  + Improves documentation
  + Improves developers communication

**UML Diagrams:**

* Nine types of diagrams
  1. Use case diagram
  2. Class diagram
  3. Object diagram
  4. State diagram
  5. Sequence diagram
  6. Collaboration diagram
  7. Activity diagram
  8. Deployment diagram
  9. Component diagram

**Use Case Diagrams:**

* They describe “WHAT” of a system
* Used to identify the primary elements and processes that form the system.
* The primary elements are termed as “actors” and the processes are called as “use cases”
* Use case diagram shows “actors” and “roles”
* It contains actors, use cases and their relationships (includes, extends, generalization)

**Class Diagrams:**

* From use case diagram, we can go to detail design of system.
* The best way to identify classes is to consider all “NOUNS” in a use cases as classes, “VERBS” as method of classes.
* Relation between actors can be used to define relation between classes.
* Relationship or association between the classes can be either “is-a” or “has-a” relationship.

**Object Diagrams:**

* An object is an instance of a class.
* Object diagram captures the state of classes in the system and their relationships or association at a specific point of time

**State Diagrams:**

* Represents the different states that objects in the system undergo during their life cycle
* Object change in response to certain simulation so this simulation effect is captured in state diagram.
* Basically, it has a initial state and final state and events that happen in between them.

**Sequence Diagrams:**

* Used to explore the logic of a complex operations, functions or procedures.
* They are called sequence diagrams because sequential nature is shown via ordering messages.
* First msg. starts at the top and the last message ends at bottom.

**Collaboration Diagram:**

* It groups together the interactions between different objects to fulfil a common purpose.

**Activity Diagram:**

* It is typically used for business process modeling.
* For modeling the logic captured by a single use case or for visualizing the detailed logic of a business rule.
* Complicated process flows in the system are captured in this diagram.
* It has activities, actions, transitions, initial and final state, and guard conditions.

**Deployment Diagram:**

* It shows
  + the hardware for the system
  + The software that is installed on that hardware
  + The middleware used
* Basically, it shows how the hardware and software work together to run a system.
* In a single line, it shows the deployment view of the system

**Component Diagram:**

* Represents the high-level parts that make up the system.
* From Java point of view, they form “PACKAGES”.
* From .NET point of view, they form “NAMESPACES”.
* At a high level, it shows what components form part of the system and how they are interrelated.
* It shows the logical grouping of classes or group of other components.

**System Development Phases:**

* Requirements (Use Case diagrams, activity diagrams)
* Analysis
* Design (class diagrams, object diagrams, Component diagrams, collaboration diagrams, deployment diagrams, sequence diagrams)
* Implementation (Class diagrams, Reverse engineering)
* Testing