**UML 2.5 NOTATION RULES FOR DRAWING CLASS DIAGRAMS**

**SYSTEMS ANALYSIS DIAGRAMS**

The UML notation is rich and full bodied. It is comprised of two major subdivisions.

* **There is a notation for modelling the static elements of a design such as classes, attributes, and relationships.**
* There is also a notation for modelling the dynamic elements of a design such as objects, messages and finite state machines.

Static models are presented in diagrams called: Class Diagrams

**CLASS DIAGRAMS**

|  |  |  |
| --- | --- | --- |
| **COMPONENT** | **RULES** | **EXAMPLE** |
| **CLASSES** | * A class is a description of a set of objects that share the same attributes, operations, relationships, and semantics * Classes are composed of three things: a name, attributes, and operations.      * Graphically, a class is rendered as a   rectangle, usually including its name,  attributes, and operations in separate,  designated compartments |  |
| **Elements of a Class Diagram** | * **Class** * **Attributes** * **Operations** * **Relationships** * **Associations** * **Generalisation** * Dependency [ not studied] * Realisation [not studied] * Constraint rules and notes [not studied] |  |
| **NAMING CLASSES** | **Classes are singular, no spaces, underscore or hyphens.**  **First letter is a capital letter.**  **New word starts with a capital letter.** |  |
| **CLASS - ATTRIBUTES** | **Attributes comprise a name and data type**  **Attributes – first letter is lower case, any new word starts with an upper case letter. No spaces, hyphens or underscores.**  **A colon follows the attribute name.**  **The attribute data type is then listed – first letter for the data type is uppercase.** | **name: String**  **age: Integer**  **weight: Double**  **isMarried: Boolean** |
| **CLASS - BEHAVIOURS** | **Behaviours – first letter is lower case, any new word starts with an upper case letter. No spaces, hyphens or underscores. A pair of brackets (empty for now) follow the behaviour name.** | **getTotal()**  **calculateArea()**  **checkCreditRating()**  **addStudent()**  **updateCourse()**  **displayResults()**  **dealHand()** |
| **CLASS ASSOCIATIONS** |  |  |
| **Plain Association** | **A directional arrow points from the source class (Bill) to the destination class (Patient). The association is named e.g. presented to**  **Multiplicity must be included at the source and destination end of the association.** | Illness  Doctor  1..n  1  1..n  1  *diagnoses*  MedicalHistory  Appointment  0..n  1  0..n  1  *has scheduled*  Symptom  0..n  0..n  0..n  0..n  *indicates*  Patient  0..1  1  0..1  1  *provides*  0..n  1  0..n  1  *schedules*  1..n  0..n  1..n  0..n  *describes*  Bill  1  1  1  1  *leads to*  1  1..n  1  1..n  *presented to* |
| **Inheritance** | **UML permits a class to inherit from multiple superclasses, although some programming languages (e.g., Java) do not permit multiple inheritance** |  |
| **Aggregation** | **The part class [Module] is partially dependent on the whole class [Honours Course]**  **Aggregation is depicted by a white diamond arrow pointing from the part class to the whole class. The aggregation arrow can present horizontally or vertically. Multiplicity must be included.** |  |
| **Composition** | **A composition is a stronger form of aggregation, in which the parts are tightly bound up with the whole. The whole does not exist without the parts. Typically the parts live and die with the whole.**  **Composition is depicted by a black diamond arrow pointing from the part class to the whole class. The aggregation arrow can present horizontally and vertically. Multiplicity must be included. For composition – the multiplicity at the whole class end cannot be more than 1.**  **The part class [Square] is totally dependent on the existence of the whole class [Board]** |  |
| **MULTIPLICITY** | **The number of objects involved in the relationship** |  |

**UML 2.0 NOTATION RULES FOR OBJECT DIAGRAM AND CLASS DIAGRAM**

|  |
| --- |
| **Mary\_Jones: Student** |
| **name = Mary Jones**  **dateOfBirth = 4/15/1978**  **year = junior** |
| **displayDetails()** |

**An object diagram represents an instance of a class and its attributes are populated with data.**

|  |  |  |
| --- | --- | --- |
| **OBJECT DIAGRAM** | **CLASS DIAGRAM** | **BOTH** |
| Object Name – capital letters, with underscore between first name and surname. Followed by Colon and class name. All underlined. | Class name rules – noun or n oun phrase, centred, bold, singular, no spaces, no possessives (eg My), no articles (e.g. a, the),First word letter in capital, new words start with capital. | Attribute name – starts with small letter, no spaces or underscores, new word starts with a capital letter. A colon separates attribute name and data type. Data type starts with a capital letter. |
| Attributes – values are populated for the attribute. Attribute name has a small letter – followed by equal sign and the value.  Behaviours are as for the class diagram e.g. displayDetails() | Operation name followed by empty brackets. | Operation starts with a small letter, no spaces, hyphens or underscores. New word starts with a capital letter. |