**Nom**: Graph (Domini)

**Descripció**: Classe encarregada de gestionar tots els elements que conformen un graf, és a dir, tots els nodes i les seves relacions.

**Cardinalitat**: El controlador del domini sempre tindrà una única instància de graph (el graf que tingui l'usuari obert en aquell moment).

**Atributs**: name:String, nodeContainers:HashMap<NodeType, Container<Node>>, relations:Container<Relation>

**Metodes:**

* **public Graph**(String name): Constructor de la classe Graph. Instància un container per cada tipus de node i afegeix les relacions bàsiques.
* **public Graph**(): Construeix el Graph passant com a nom "-1".
* **public String getName**(): Retorna el nom del graf.
* **public void setName**(String name): Edita el nom del graf
* **public int getSize**(NodeType type): Retorna el nombre de nodes del tipus indicat que hi ha al graf.
* **public void addNode**(Node node): Afegeix un node al graf. Mira de quin tipus és el node per saber a quin container introduir-lo. Es fa servir per nodes nous introduits manualment per l'usuari (l’Id serà el que toqui per defecte).
* **public void addNode**(Node node, int id): Afegeix un node al graf. Mira de quin tipus és el node per saber a quin container introduirlo. Es fa servir per nodes llegits a la capa de persistencia (ja sabem el seu id).
* **public Node createNode**(NodeType type, String value): Classe encarregada de crear un nou node.
* **public void removeNode**(NodeType type, int id): Classe encarregada d’eliminar un node.
* **public Node getNode**(NodeType type, int id): Donat un tipus de node y el seu Id, retorna el respectiu node.
* **public ArrayList<Node> getEdges**(int relationID, Node node): Retorna una llista dels nodes als cuals està connectat el node paràmetre.
* **public ArrayList<Node> getEdges**(int relationID, NodeType type, int id): Igual que el metode anterior pero passant el node en forma de Id+TipusNode en comptes d’una instància de Node.
* **public void addEdge**(int relationID, Node a, Node b): Donats dos nodes diferents afegeix una aresta que els connecti.
* **public void addEdge**(int relationID, NodeType typeA, int idA, NodeType typeB, int idB): Fa el mateix que el mètode anterior pero en comptes de rebre els nodes paràmetre com una instància de node, els reb com a parelles Id+TipusNode.
* **public void removeEdge**(int relationID, Node a, Node b): Donats dos nodes esborra la aresta que els connecta.
* **public void removeEdge**(int relationID, NodeType typeA, int idA, NodeType typeB, int idB): Fa el mateix que el mètode anterior pero en comptes de rebre els nodes paràmetre com una instància de node, els reb com a parelles Id+TipusNode.
* **public Container<Node>.ContainerIterator getNodeIterator**(NodeType type): Retorna un iterador que permet recorrer tots els nodes del tipus passat com a paràmetre.
* **private void i\_addRelation**(Relation relation, NodeType type)
* **public void addRelation**(Relation relation): Afegeix una relacio nova al sistema. Part del codi d’aquest mètode està al mètode privat i\_addRelation.
* **private void i\_removeRelation**(Relation relation, NodeType type)
* **public void removeRelation**(int relationID): Elimina una relació existent. Part del codi d’aquest mètode està al mètode privat i\_addRelation.
* **public Relation getRelation**(int relationID): Retorna la relació referenciada per l’Id passat com a paràmetre.
* **private NodeType getNodeType**(Node node): Classe encarregada de retornar el tipus del node passat com a paràmetre.
* **private Container<Node> getNodeContainer**(NodeType type): Retorna el contenidor associat als nodes del tipus indicat.