**Especificació de les classes compartides:**

**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 relationID.
* **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.

**Nom**: NodeType (Domini)

**Descripció:** Classe de tipus Enumeration que defineix un dels 5 tipus de Nodes que hi ha al programa (Paper, Author, Conference, Label, Term).

**Cardinalitat:** -

**Atributs**: -

**Mètodes: -**

**Nom**: Node (Domini)

**Descripció**: Classe que s’encarrega de crear les instàncies de nodes, i tenir totes les relacions que té.

**Cardinalitat**: Molts. Un graf tindrà moltes instàncies de la classe Node

**Atributs**: private HashMap<Integer, HashSet<Integer>> relations;

**Metodes:**

* **protected Node(String value):** Crea un nou node donat el seu nom.
* **public Node(int nodeID, String value):** Crea un nou node donant-ne el seu identificador i el seu nom.
* **public Node(Node n) :** Crea una copia del node.
* **protected void addEdge(int relationID, int nodeID):** Afegeix una relació del tipus donat per el primer parametre amb el node determinat per el segon.
* **protected void removeEdge(int relationID, int nodeID):**  Esborra una relació del tipus donat per el primer parametre amb el node determinat per el segon.
* **public boolean isRelated(int relationID, int nodeID):** Determina si un node està relacionat amb un altre donant el id de la relació i el id del segon node.
* **public boolean isRelated(int relationID, Node node):** Determina si un node està rlacionat amb un altre donant el id de la relació i el segon node.
* **protected ArrayList<Integer> getEdges(int relationID):** Retorna totes les relacions del tipus relationID que té el node.
* **public int getSizeEdges(int relationID):** Retorna el nombre de relacions del tipus relationID que té el node.
* **protected void addRelation(Relation relation):**  Afegeix un nou tipus de relació per al node.
* **protected void removeRelation(Relation relation):**  Esborra un tipus de relació per al node.

**Nom**: Relation (Domini)

**Descripció**: Classe que representa un tipus de relació entre dos tipus de nodes. És subclasse d’Element.

**Cardinalitat**: Moltes. S’instanciarà un cop per cada tipus de relació present al graf.

**Atributs**: NodeType A, Nodetype B, String name, Int Id.

**Metodes:**

* **public Relation**(NodeType A, NodeType B, String name): constructora que crea una instància de Relation, que va d’un node de sortida A a un node d’entrada B i que té nom name.
* **public Relation**(NodeType A, NodeType B, String name, int id): constructora que crea una instància de Relation, que va d’un node de sortida A a un node d’entrada B, que té nom name i identificador igual a id.
* **public Relation**(Relation r): constructora que crea una instancia de Relation, que serà copia de la relació “r” passada per paràmetre.
* **public String getName**() : retorna el nom de la relació.
* **protected boolean isDefault**(): retorna true si la relació és una de les relacions predefinides. Retorna false altrament.
* **protected NodeType getNodeTypeA**(): retorna el tipus de node del node de sortida de la relació.
* **protected void setNodeTypeA**(Nodetype type): canvia el tipus de node del node de sortida de la relació per l’indicat al paràmetre type.
* **protected NodeType getNodeTypeB**(): retorna el tipus de node del node d’entrada de la relació.
* **protected void setNodeTypeB**(Nodetype type): canvia el tipus de node del node d’entrada de la relació per l’indicat al paràmetre type.

**Nom**: Element (Domini)

**Descripció**: Classe abstracta dedicada arepresentar qualsevol element dins del domini.

**Cardinalitat**: 0. Ésabstracta.

**Atributs**: String: value, int: id

**Metodes:**

* **Element(String Value):** Constructora.
* **Element(int Id, String Value):** Constructora.
* **Element (Element e):** Constructora.
* **getId():** retorna la id de l’element
* **setId(int id):** assigna un id a l’ element.
* **setValue(String Value):** guarda la informació a l’element.
* **getValue():** retorna la informació d’un element