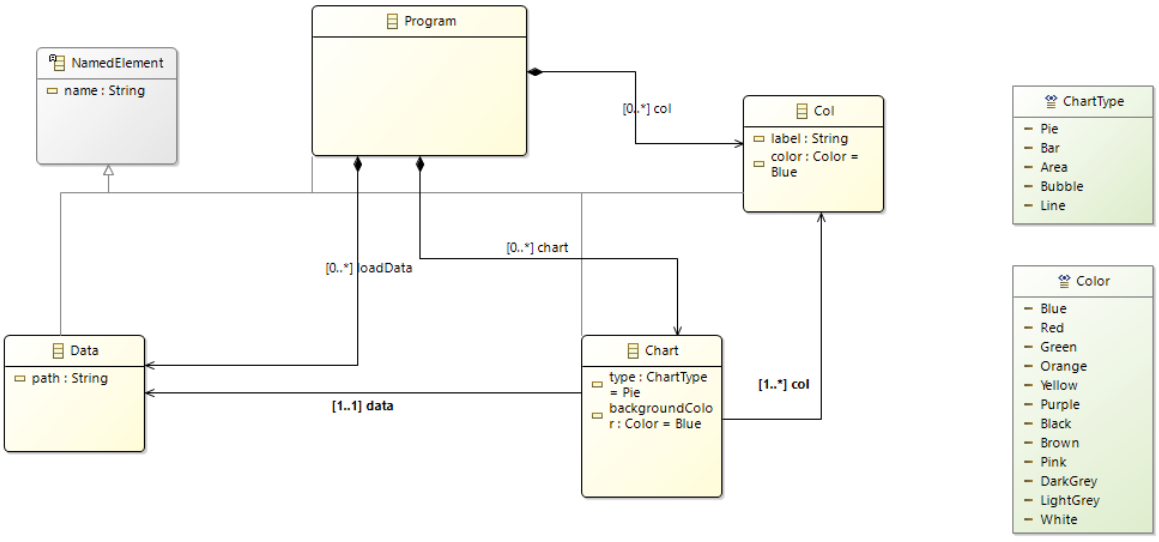
Langage et Compilation

ChartIT

I/ Le Chart

Le Chart est un langage qui permet de générer un graphique sur une page html à partir d'un csv. Le nom donné au programme définit le titre de la page html. Avant de créer un Chart, il faut créer une variable Data qui prend un chemin absolu vers le csv où se trouve les données voulant être afficher, il faut aussi créer une variable Col dont le nom sera le nom de la colonne voulu dans le csv et le label le nom affiché sur le graphique. La variable Col contient aussi une variable color permettant de différencier avec une des couleurs proposées les données de la colonne. Ensuite, pour créer un Chart, on initialise une variable Chart dont le nom sera le nom du graphique. On choisit ensuite les données voulues ainsi que le type de graphique souhaité, puis on définit les colonnes à utiliser ainsi que la couleur de l'arrière-plan voulu.

II/ Syntaxe concrète (metamodèle)



J'ai tout d'abord ajouté la classe abstraite NamedElement pour que toute les classes possèdent un nom et j'ai pris comme racine la classe Program. J'ai ensuite ajouté la class Data contenant un path afin de récupérer le chemin absolu menant au csv voulu ainsi que la classe Col contenant un label et une color afin de renommer les données de la colonne et de les colorées sur le graphique. Ces 2 classes sont contenues dans la classe Chart qui contient aussi un type et un backgroundColor pour choisir le type du graphique ainsi que sa couleur de fond. Pour finir j'ai créé les énumération ChartType et Color pour avoir des types ainsi que des couleurs prédéfinis.

III/ Syntaxe concrète en BNF

{ <Program> } 'P' ':' name=STRING '{'

'loadData' '{' loadData+=<Data> + ("," loadData+=<Data>)'}'

'col' '{' col+=<Col> + ("," col+=<Col>) '}'

'chart' '{' chart+=<Chart> +("," chart+=<Chart>)*'}'

'}';

{ <Data> } name=STRING '(' path=STRING ')';

{ <Chart> } name=STRING

'('

'data' '=' data=[<Data>|STRING]',' 'type' '=' type=<ChartType> ',' 'col' '=' col+=[<Col>|STRING] (',' col+=[<Col>|STRING])* ',' 'backgroundColor' '=' backgroundColor=<Color>

')' ;

{ <Col> } name=STRING '(' 'label' '=' label=STRING ',' 'color' '=' color=<Color> ')';

<Color> ::= Blue='Blue' | Red='Red' | Green='Green' | Orange='Orange' | Yellow='Yellow' | Purple='Purple' | Black='Black' | Brown='Brown' | Pink='Pink' | DarkGrey="Darkgrey" | LightGrey='LightGrey' | White='White';

<ChartType> ::= Pie = 'Pie' | Bar = 'Bar' | Area = 'Area' | Bubble = 'Bubble' | Line = 'Line';

IV/ Description du xtend

Pour cette partie, j'ai comme base la fonction « generateHTMLCode » dans laquelle on parcourt tous les « Chart » créés dans le programme. Le code génère un fichier html type en prenant le nom du programme comme nom de page, ensuite, la variable « chartH » contient tous les « canvas » avec leurs noms associés pour créer un graphique lesquels sont chacun contenus dans une div de la couleur d'arrière-plan choisit, tout ceci généré dans la fonction « generateHTMLChart ». Pour ce qui est du javascript, la variable « dataload » contient toutes les données nécessaires aux graphiques générée dans la fonction « generateData » et la variable « js » contient la génération de « Chart » de la librairie Chartjs en se référant par id à leur « canvas » respectif et en utilisant les données générées précédemment ainsi que la couleur choisie pour chaque donnée.