Rapport Projet Java

### Annuler demandes

Pour annuler une demande on ne va pas supprimer la tâche physiquement du tableau de tâche. La méthode d’annuler de la classe tâche va être appelée. Cette méthode va changer l’état de da tâche. Pour tous les sous type de ressources dans son emploi du temps on a le pointeur vers la tâche. Donc une fois le état de objet tâche a changé. Pour toutes les ressources qu’il a besoin les créneaux va être libéré.

Pour annuler la tâche logiquement on va changer un variable « etat : int» de la classe « Tache » qui signifie l’état de la tâche.

Les valeurs de etat et sa sens sont marqué dans le tableau au de sous :

|  |  |
| --- | --- |
| Valeur | Sens |
| 0 | Tâche déjà initialisé, en attendre de traiter |
| -1 | Tâche déjà annulé |
| 1 | Tâche déjà traiter |

Donc pour annuler les tâches on modifier simplement l’état de la tâche a -1. La méthode annuler() utilise le algorithme au de sous :

|  |
| --- |
| Classe :Tache |
| annuler() :booléen |
| Local :resultat :booléen |
| boolean resultat = (etat==0);  //Pour vérifier si la tâche est annulable  //si le tâche tâche a déjà annulé le méthode va retour faux  etat = -1;  return resultat; |

### Traiter les demandes

On va d’abord créer une nouvelle tâche. Pour cette tâche par apport son type donc soit une tâche d’entretien, soit une tâche informatique, soit une tâche générique, on ajoute dans son tableau de ressources les ressources (les salles, les logiciels, les matériels etc.) qu’il a besoin.

On ajoute la nouvelle tâche dans le tableau de tâche.

On parcourir le tableau de employés pour chercher une employé disponible.

Pour faire ça pour les objets de type on appelle la méthode ajouterTache(Tache) de son emploi du temps qui retour une valeur de type booléenne pour dit que si la tâche a déjà ajouter avec succès.

La méthode ajouterTache(Tache) utilise le algorithme au de sous :

|  |
| --- |
| Classe :EmploiTemps |
| ajouterTache(Tache) :booléen |
| Donne de la classe :  nbCreneauJour :entier //nombre de créneaux dans un jour  nbJour :entier //nombre de jours dans une semaine  Donner : tache : Tache  Local :  nbLibre ptJour ptCre: entier initialisé à 0;  flagTrouve : bool initialisé à false |
| tq flagTrouve==false et ptJour<nbJour faire  nbLibre = 0;  ptCre = 0;  tq flagTrouve==false et ptCre<nbCreneauJour faire  si est libre ou la tache a ete annuele faire  nbLibre = nbLibre+1  si nbLibre >= nombre de créneau de tache  et  pour tous les ressources  dans la tableau de ressources de tache  sont libre  faire  tableTache[ptJour][ptCre-tache.nbCreneaux+1] = tache;  tache.etat = 1;  flagTrouve = true;  fin si  ptCre = ptCre++  sinon  nbLibre = 0;  ptCre = ptCre + tableTache[ptJour][ptCre].nbCreneaux;  fin si  fait  ptJour = ptJour+1;  fait  return flagTrouve; |

Si après le parcourir de tableau des employés la tâche va être annulée (rejeté).

La classe ressources est une classe abstraite qui va être hérité par la classe employé, la classe salle et les classe logiciel et matériel. Il est interdit de être instancier. Il contient les variables et les méthodes de la classe d’employé, la classe salle, logiciel, matériel etc. Donc on a nom et numéro des ressources et plus important les emplois du temps.

La méthode « verifier » va appeler la méthode « verifier » de la classe « EmploiTemps » donc on peut est-ce que la ressources est disponible ou pas.

La méthode « ajouterTache(int jour,int creneau,Tache tache) » va appeler la méthode « ajouterTache » de la classe « EmploiTemps ». Donc on va pouvoir ajouter la tâche à l’emploi du temps.