DOCUMENT 3 : exemple d'événements au format *JSON*

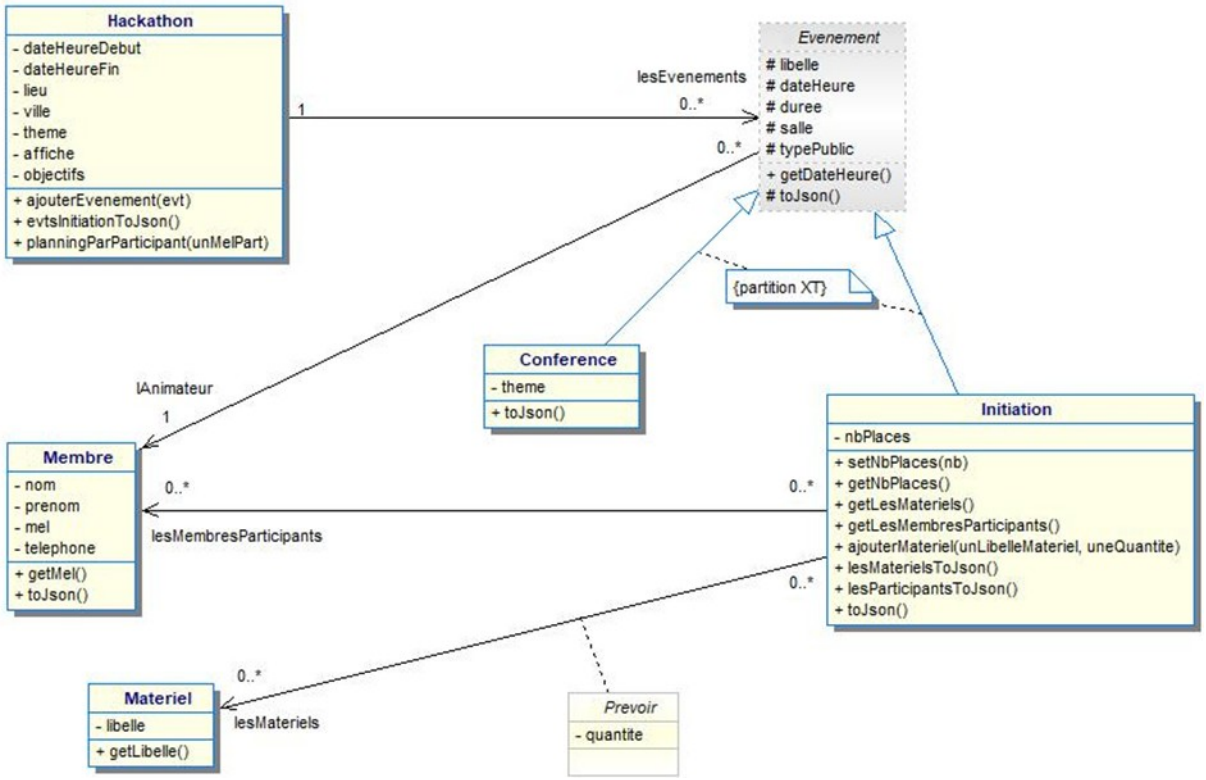
Exemple d'évènements de type initiation :

```
[{
 "libelle" : "Introduction au PHP",
  "dateHeure" : "19/06/2021 13:00",
  "duree" : "2h",
  "salle" :  "Alan Turing",
  "typePublic" : "étudiants ou jeunes développeurs débutants",
  "animateur" : {
                        "prenom" : "Morgan",
                        "nom" : "Friche",
                        "mel" : "mfriche@mail.com",
                        "telephone" : "06 39 98 65 14"
            },
  "nbPlaces" : 40,
  "materiels" : [
                        {
                        "libelle" : "ordinateur portable ",
                        "quantite" : 1
                         },
                        {
                        "libelle" : "IDE PHPStorm en mode évaluation gratuit",
                        "quantite" : 1
                         }
            ]
},
{ "libelle" : "Introduction à la cybersécurité",
  "dateHeure" : "19/06/2021 16:00",
  "duree" : "2h",
  "salle" :  "Alan Turing",
  "typePublic" : "étudiants ou jeunes développeurs débutants",
  "animateur" : {
                        "prenom" : "Morgan",
                        "nom" : "Friche",
                        "mel" : "mfriche@mail.com",
                        "telephone" : "06 39 98 65 14"
            },
  "nbPlaces" : 40,
  "materiels" : [
                        {
                        "libelle" : "bloc-notes papier",
                        "quantite" : 1
                         }
            ]
} ]
```

Exemple d'un événement de type conférence :

```
{ "libelle" : "Les méthodes agiles ?",
  "dateHeure" : "19/06/2021 09:00",
  "duree" : "2h",
  "salle" :  "Hedy Lamarr",
  "typePublic" : "chefs de projets novices",
  "animateur" : {
            "prenom" : "Louison",
            "nom" : "Gelin",
            "mel" : "lgelin@mail.com",
            "telephone" : "06 39 98 23 01"
            },
  "theme" : "gestion de projet"
}
```

DOCUMENT 4 : diagramme de classes du module *Hackat'Event*



La classe Evenement est une classe abstraite donc elle ne pourra pas être instanciée. Seules ses deux classes filles, Conference et Initiation pourront être instanciées.

DOCUMENT 5 : code *PHP* partiel des classes du module *Hackat'Event*

```
class Materiel{
   private $libelle;

   public function __construct($libM){
      $this->libelle = $libM;
   }
   public function getLibelle() {return $this-> libelle ;}
}
```

```
class Hackathon {
   private $dateHeureDebut;
   private $dateHeureFin;
   private $lieu;
   private $ville;
   private $theme;
   private $affiche;
   private $objectifs;
   private $lesEvenements; // type : collection d'objets Evenement

   public function __construct($dhd, $dhf, $lieu, $ville, $theme, $aff, $obj){
      $this->dateHeureDebut = $dhd;
      $this->dateHeureFin = $dhf;
      $this->lieu = $lieu;
      $this->ville = $ville;
      $this->theme = $theme;
      $this->affiche = $aff;
      $this->objectifs = $obj;
      $this->lesEvenements = array();
   }
   public function ajouterEvenement($evt){
      array_push($this->lesEvenements, $evt);
   }
   public function evtsInitiationToJson() {/// retourne une chaine de caractères
     $chaineJson = '"EvenementsInitiations" : [';
     foreach ($this->lesEvenements as $evt){
        if ($evt instanceof Initiation){
           $chaineJson = $chaineJson . $evt->toJson(); // concaténation
        }
     }
     return $chaineJson . " ]\n" ;
   }
   public function planningParParticipant($unMelPart) {
      $dicoEvtPart = array();
      foreach ($this->lesEvenements as $evt){
         if ($evt instanceof Initiation){
            foreach ($evt->getLesMembresParticipants() as $unMembre){
               if ($unMembre->getMel() == $unMelPart){
                  $dicoEvtPart[$evt->getdateHeure()] = $evt;
               }
            }
         }
      }
      return $dicoEvtPart;
   }
}
```

```
class Membre{
   private $nom;
   private $prenom;
   private $mel;
   private $telephone;

   public function __construct($nom, $prenom, $mel, $tel){...}
   public function getMel() { return $this->mel ; }
   public function toJson() {
       // méthode retournant tous les attributs de la classe au format JSON
       return json_encode(["nom"=> $this->nom,"prenom"=> $this->prenom,
                           "mel"=>$this->mel,"telephone"=>$this->telephone]);
   }
}
```

```
abstract class Evenement{
    protected $libelle;
    protected $dateHeure;
    protected $duree;
    protected $salle;
    protected $lAnimateur; // type: Membre
    protected $typePublic;

    public function __construct($libelle, $dateHeure, $duree, $salle, $lAnimateur, $leTypePublic) {
                    $this->libelle = $libelle;
                    $this->dateHeure = $dateHeure;
                    $this->duree = $duree;
                    $this->sale = $salle;
                    $this->lAnimateur = $lAnimateur;
                    $this->typePublic = $leTypePublic;
            }
    public function getDateHeure(){ return $this->dateHeure;}
    protected function toJson(){
            // méthode retournant tous les attributs de la classe au format JSON
        return " { \"libelle\" : \""
                                .$this->libelle."\", \n \"dateHeure\" : \""
                                .$this->dateHeure."\", \n \"duree\" : \""
                                .$this->duree."\", \n \"salle\" :  \""
                                .$this->salle."\" , \n \"typePublic\" :  \""
                                .$this->typePublic."\", \n \"animateur\" : "
                                .$this->lAnimateur->toJson();
    }
}
```

```
class Conference extends Evenement {
   private $theme;

    public function __construct($libelle, $dateHeure, $duree, $salle, $lAnimateur, $leTypePublic, $theme) {
        parent::__construct($libelle, $dateHeure, $duree, $salle, $lAnimateur, $leTypePublic);
        $this->theme = $theme;
    }
    public function toJson(){
        // méthode retournant tous les attributs de la classe au format JSON
        return parent::toJson().",\n \"theme\" : \"".$this->theme."\"\n }\n";
    }
}
```

```
class Initiation extends Evenement {
   private $nbPlaces;
   private $lesMateriels;  /* Ce dictionnaire de (string, int) implémente la classe-association Prévoir.
                              La clé correspond au libelle du matériel et la valeur à la quantité.*/
   private $lesMembresParticipants; // type : collection de Membre

   public function __construct($libelle,$dateHeure, $duree, $salle, $lAnimateur, $leTypePublic, $nb) {
                // Question B.1 b) :  à compléter et à reporter sur votre copie
   }
   public function setNbPlaces($nb){
      if ($nb >= 0) {
         $this->nbPlaces = $nb;
      }
   }
   public function getNbPlaces(){return $this->nbPlaces;}
   public function getLesMateriels(){return $this->lesMateriels;}
   public function getLesMembresParticipants(){
       return $this->lesMembresParticipants;
   }
   public function ajouterMateriel($unLibelleMateriel, $uneQuantite) {
                // Question B.1 a) :  à corriger et à reporter sur votre copie
      if ($unLibelleMateriel != null ) {
        if (array_key_exists($unLibelleMateriel, $this->lesMateriels) == false)
        {  // on vérifie que la clé n'existe pas déjà dans le dictionnaire
             $this->lesMateriels[$unLibelleMateriel] = $uneQuantite;
             /* ajoute dans le dictionnaire la clé de type String correspondant au libelleMateriel et la
                  valeur de type entier correspondant à la quantité demandée*/
        }
      }
   }
   private function lesMaterielsToJson(){
         // retourne une chaine de caractères
      $chaineJson = "\"materiels\" : [  \n";
      $debutChaine = true;
      foreach ($this->lesMateriels as $lib => $qte) {
             if ($debutChaine == false) {
                 // on va ajouter un élément supplémentaire, on le sépare par une virgule
                 $chaineJson = $chaineJson . ",\n"; //concaténation
             }
             else {
                  $debutChaine = false ;
             }
                 // on ajoute l'élément (concaténation)
             $chaineJson = $chaineJson . "{ \n \"libelle\" : \"".$lib . "\",\n \"quantite\" : \"" . $qte . "\" }";
      }
      $chaineJson = $chaineJson . "]  \n"; //concaténation
      return $chaineJson;
   }
   private function lesParticipantsToJson() {      // retourne une chaine de caractères
                // Question B.1 d) :  à compléter et à reporter sur votre copie
   }
   public function toJson()  {         // retourne une chaine de caractères
                // Question B.1 e) :  à compléter et à reporter sur votre copie
   }
}
```

DOCUMENT 6 : extrait de la classe de test TestInitiation

```
class TestInitiation{

 private $evtInitPHP ;

 public function testCreationInitiation() {
     $lAnim = new Membre("Friche", "Morgan", "mfriche@mail.com", "06 39 98 23 01");
     $this->evtInitPHP = new Initiation("Introduction au PHP","19/06/2021 13:00", "2h",
                                        "Alan Turing", $lAnim, "étudiants et jeunes développeurs", 40);
 }
 public function testAjouterMaterielQuantiteZero(){
    $this->testCreationInitiation();  // appel de la méthode qui instancie l'évènement initiation
    $leMateriel = new Materiel("ordinateur portable");
    $this->evtInitPHP->ajouterMateriel($leMateriel->getLibelle(), 0);
    $this->assertEquals(0, count($this->evtInitPHP->getLesMateriels()),
                                                    "Matériel ajouté alors que la quantité est égale à 0");
 }
 public function testAjouterParticipantLimiteNombrePlaces() {
     $this->testCreationInitiation();
      // on force le nombre de places de l'initiation à 2 places
     $this->evtInitPHP->setNbPlaces(2);
     $leParticipant1 = new Membre("Mallien", "Yannick", "myannick@mail.com", "06 39 98 15 12");
     $this->evtInitPHP->ajouterParticipant($leParticipant1) ;
     $leParticipant2 = new Membre("Dus", "Dominique", "ddus@mail.com", "06 39 98 00 56 ");
     $this->evtInitPHP->ajouterParticipant($leParticipant2) ;
     $leParticipant3 = new Membre("Smith", "Jean", "jsmith@mail.com", "06 39 98 85 17");
     $this->assertFalse($this->evtInitPHP->ajouterParticipant($leParticipant3),
                                                          "mauvaise gestion des places disponibles");
     $this->assertEquals(2, count($this->evtInitPHP->getLesMembresParticipants()),
                                                            "erreur dans l'ajout du 3ème participant");
 }
}
```