**Hackat'Innov**

Le sujet compte 11 pages dont 6 pages de documentation.

Le sujet est constitué de quatre dossiers qui peuvent être traités de façon indépendante.

Présentation du sujet 2-5

Dossier documentaire 6-12

**Dossier documentaire**

DOCUMENT 1 : exemple d’événements au format *JSON* 5

DOCUMENT 2 : diagramme de classes du module *Hackat'Event* 6

DOCUMENT 3 : code *PHP* partiel des classes du module *Hackat'Event* 7

DOCUMENT 4 : extrait de la classe de test FormationTest 10

DOCUMENT 5 : extrait de documentation *PHP* 10

**Barème**

|  |  |  |
| --- | --- | --- |
| **Dossier** | Gestion d’évènements organisés autour des hackathons | 100 points |
|  | TOTAL | 100 points |

**Présentation du contexte**

Contraction de « hack » et de « marathon », l'hackathon est un processus créatif utilisé dans le domaine de l'innovation numérique. Durant généralement un week-end, des développeurs volontaires se réunissent pour faire de la programmation informatique en mode collaboratif sur un thème défini. L’engouement pour ce type de manifestation témoigne d’une volonté de développer toujours davantage le numérique au service de la société.

Nous distinguerons les phases du déroulement d’un hackathon, sur le week-end, des étapes de son organisation, qui commence bien en amont.

**Les phases du déroulement d’un hackathon**

Même s’ils peuvent porter sur des thèmes variés, les hackathons se déroulent toujours selon un même schéma :

* l’accueil des participants et la présentation (thème, projets retenus, constitution des équipes, planning, jury, etc.) constituent la première phase du hackathon ;
* les équipes travaillent ensuite sur le développement de leur projet : remue-méninges, maquettage, prototypage, etc. Lors de ces phases de travail, les équipes sont accompagnées par des mentors (*coachs*) de différents domaines (développement, graphisme, communication, etc.) ;
* à l’issue de ce travail intensif, chaque équipe présente son prototype au jury ;
* enfin, la délibération du jury puis la proclamation des résultats et la remise des récompenses constituent la dernière phase du hackathon.

Un hackathon est également souvent l’occasion d’organiser des évènements satellites tels que des conférences, des ateliers de formation ou encore des tables rondes. Ces évènements satellites ont un double objectif :

* permettre aux membres des équipes de développer ou d’acquérir des connaissances et des compétences utiles pour leurs projets ;
* attirer du public extérieur pour le sensibiliser aux thématiques abordées et faire connaître le hackathon.

**Les étapes de l’organisation d’un hackathon**

Le processus d’organisation, essentiel pour assurer le bon déroulement du hackathon, se décompose en plusieurs étapes :

* initialisation : choix du lieu, de la date et du thème, composition du jury, contact avec les organisateurs ;
* publication du hackathon : édition et mise en ligne du planning ;
* inscription des participants : ouverture des inscriptions en ligne ;
* choix des projets : détermination des projets retenus en fonction du positionnement des participants et constitution des équipes ;
* lancement : lancement du hackathon avec choix de la ou du chef de projet de chaque équipe, démarrage ;
* clôture : livraison des prototypes par les équipes pour la présentation au jury de chacun par sa ou son chef de projet ;
* résultats : gestion des votes des membres du jury puis édition des résultats.

**L'organisation cliente et le prestataire de service informatique**

Pour répondre au développement des hackathons en France, Mme Majdouline Mabille a créé, il y a deux ans, une entreprise jeune-pousse (*start-up*) Hackat’Innov qui a pour but de simplifier la gestion de l'organisation d'hackathons. Son entreprise a reçu un prix de l'innovation numérique et a donc pu lever des fonds nécessaires à l'embauche de deux personnes supplémentaires.

Le service informatique de l’entreprise, composé de trois personnes, est chargé de réaliser les projets d’évolution du système d’information d’Hackat’Innov.

**Les projets d’évolution du système d’information**

Un des principaux projets de Hackat’Innov est le développement d’une application *Web* *Hackat'Orga* qui permet de gérer les étapes du processus d’organisation d’un hackathon : initialisation, publication, inscription, choix des projets, lancement, clôture et résultats. La première étape (l’initialisation d’un hackathon) a été développée dans l'application lors d’une première itération. Aujourd'hui, Mme Mabille souhaite :

* finaliser l'application *Hackat'Orga* pour gérer l'ensemble des étapes (à l'exception du choix des projets) ;
* développer un module *Hackat’Event* permettant la gestion des évènements satellites organisés lors des hackathons.

Par ailleurs, à la demande des organisateurs d’hackathons et des participants, Hackat’Innov a lancé son service informatique dans le développement d’applications mobiles :

* une première application permettant de consulter les informations sur les hackathons gérés par Hackat’Innov est en cours de développement ;
* le développement d’une seconde application mobile, dédiée à la gestion des votes, est à l’étude.

En tant que stagiaire chez Hackat’Innov, vous participez à ces quatre évolutions sous la responsabilité de Mme Majdouline Mabille.

**Vous vous appuierez sur le dossier documentaire mis à votre disposition.**

**Gestion d’évènements organisés autour des hackathons**

***Documents à utiliser : 1,2,3,4 et 5***

***IMPORTANT : la candidate ou le candidat peut choisir de présenter les éléments de code à l’aide du langage PHP***

Afin d'apporter de l'attractivité auprès d'un plus large public, Hackat'Innov a décidé d'organiser des évènements satellites lors des hackathons. Pour gérer ces évènements qui peuvent être des conférences comme des formations aux outils numériques animées par une ou un intervenant, un nouveau module *Hackat'Event* doit être développé.

Mme Mabille souhaite que vous implémentiez les classes métier de ce futur module.

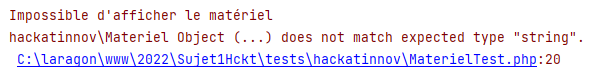
Le diagramme de classes de l'application est fourni dans le dossier documentaire. Les évènements de type « formation » seront soumis à inscription car le nombre de participants est limité pour ce type d’évènement.

On s’intéresse ici à la classe Formation. Les formations nécessitent que les participants apportent une liste de matériels et de logiciels prédéfinis. Cela peut-être des fournitures ainsi que certains logiciels à installer. Certaines méthodes de cette classe retournent une chaîne au format *JSON (JavaScript Object Notation)* afin de faciliter les échanges entre les différents applicatifs, en particulier les applicatifs mobiles.

La méthode de développement de ce projet est centrée sur les tests. Le développement piloté par les tests (*TDD - Test Driven Development*) conduit à écrire d'abord les tests et ensuite les méthodes associées. Pour s'assurer de la validité de chaque méthode, celle-ci devra passer tous les tests.

L’écriture des tests est achevée ; il faut maintenant écrire les méthodes qui réussissent ces tests. Vos collègues ont commencé, il faut compléter leur travail. Voici la liste des erreurs relevées lors de la dernière exécution des tests sur la classe Formation :

1. Le test *testAjouterMaterielQuantiteZero* signale une erreur dans la méthode *ajouterMateriel* de la classe Formation, le message retourné est « Matériel ajouté alors que la quantité est égale à 0».
2. Le constructeur de la classe Formation est incomplet et celui de la classe Conference est manquant.
3. Le test testAffichageMateriel signale une erreur :



1. Le test *testAjouterParticipantLimiteNombrePlaces* ne renvoie rien de concluant car la méthode *ajouterParticipant* de la classe Formation n’est pas écrite. La signature de cette méthode n’est d’ailleurs pas définie.
2. Cette méthode une fois écrite devra finir d’être testée
3. Il manque également la méthode *lesParticipantsToJson dans la classe* Formation*.*
4. La méthode *toJson* de la classe Formation n’est pas écrite non plus. Les spécifications demandent que toutes les caractéristiques d’une séance de formation apparaissent, ainsi que la liste des matériels et des participants sauf s’il n’y a pas de participants.
5. Cette méthode devra également être testée

**Question 1 – Corrections des erreurs signalées ci-dessus**

* 1. Corriger la méthode *ajouterMateriel*.
  2. Compléter et Écrire le constructeur des classes Formation et Conference
  3. Effectuer les corrections nécessaires pour que le test testAffichageMateriel fonctionne
  4. Écrire la méthode *ajouterParticipant*
  5. Compléter la méthode de test unitaire de la méthode ajouterParticipant (ajout possible d’un 2ème participant mais pas d’un 3ème)
  6. Écrire la méthode *lesParticipantsToJson*.
  7. Écrire la méthode *toJson*
  8. Ecrire une méthode de test unitaire de la méthode toJson

**Question 2 – Gestion du planning des évènements des membres au sein de l'hackathon**

Décrire la structure du dictionnaire retourné par la méthode *planningParParticipant* de la classeHackathon.

Expliquez ce qu’il faudrait faire comme tests unitaires pour valider pleinement cette méthode.

**Question 3 – Demande d’évolution**

Mme Mabille souhaite pouvoir planifier à l’avance l’intervention des conférenciers. Elle souhaite pouvoir disposer d’une liste des animateurs devant intervenir pour une conférence ultérieurement.

Une nouvelle classe métier, nommée Planification, est proposée, avec comme attribut une dateDePlanification. Cette classe est associée aux Hackatons.

1. Vous développerez une méthode permettant d’ajouter un Hackaton à la liste des Hackatons de la Planification
2. Vous développerez un test unitaire de cette méthode d’ajout
3. Vous développerez une méthode nommée prepareListeConferencier qui renvoie la liste des conférenciers prévus après la date de planification (on ne souhaite voir apparaître le conférencier, identifié par son mail, qu’une seule fois dans la liste, s’il intervient plusieurs fois …)
4. Vous vérifierez que l’embryon de test unitaire prévu passe bien.
5. Vous compléterez la classe de test unitaire pour tenir compte de la demande exprimée

**Note** : pour la question 3.b vous pouvez être amené à compléter à la marge les autres classes existantes.

DOCUMENT 1 : exemple d’événements au format *JSON*

**Exemple d’évènements de type formation :**

[{

"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",

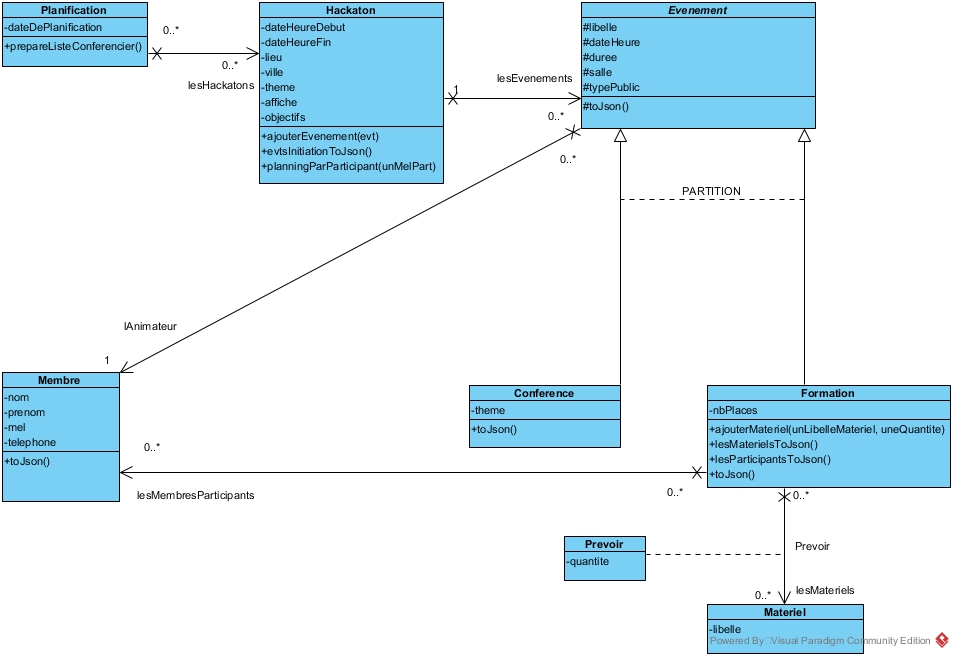
"telephone" : "06 39 98 23 01"

},

"theme" : "gestion de projet"

}

DOCUMENT 2 : 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 Formation pourront être instanciées.

DOCUMENT 3 : code *PHP* partiel de certaines 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 evtsFormationToJson() *{// retourne une chaine de caractères*  
 $chaineJson = '"Evenements Formations" : [';  
 foreach ($this->lesEvenements as $evt){  
 if ($evt instanceof Formation){  
 $chaineJson = $chaineJson . $evt->toJson(); *// concaténation*  
 }  
 }  
 return $chaineJson . " ]\n" ;  
 }  
 **public function planningParParticipant($unMelPart)** {

$dicoEvtPart = array();  
 foreach ($this->lesEvenements as $evt){  
 if ($evt instanceof Formation){  
 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."\", \r\n \"dateHeure\" : \""

.$this->dateHeure."\", \r\n \"duree\" : \""

.$this->duree."\", \r\n \"salle\" : \""

.$this->salle."\" , \r\n \"typePublic\" : \""

.$this->typePublic."\", \r\n \"animateur\" : "

.$this->lAnimateur->toJson();  
 }  
}

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

***// Question : Constructeur complet à ajouter***

public function toJson(){  
 *// méthode retournant tous les attributs de la classe au format JSON* return parent::toJson().",\r\n \"theme\" : \"".$this->theme."\"\r\n }\r\n";  
 }  
}

**class Formation 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, $nb)** {  
 ***// Question : à compléter et à reporter sur votre copie***   
 }  
 public function setNbPlaces($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, $uneQuantite)** {  
 ***// Question : à corriger et à reporter sur votre copie*** if ($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\" : [ \r\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 . ",\r\n"; *//concaténation*

}

else {

$debutChaine = false ;

}

*// on ajoute l’élément (concaténation)*

$chaineJson = $chaineJson . "{ \r\n \"libelle\" : \"".$lib . "\",\r\n \"quantite\" : \"" . $qte . "\" }";

}

$chaineJson = $chaineJson . "] \r\n"; *//concaténation*

return $chaineJson;

}

**private function lesParticipantsToJson()** { *// retourne une chaine de caractères*   
 ***// Question: à compléter et à reporter sur votre copie***

}  
 **public function toJson()** {  *// retourne une chaine de caractères*

***// Question: à compléter et à reporter sur votre copie***

}  
}

**Le contenu partiel des classes Planification, MaterielTest et PlanificationTest est disponible directement en source.**

DOCUMENT 4 : extrait de la classe de test FormationTest

**class FormationTest** {

private $evtFormationPHP ;

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

DOCUMENT 5 : extrait de documentation *PHP*

**Exemple d'utilisation de la collection (tableau indicé en *PHP*)**

private $lesParticipants; *// déclaration d'une variable d'instance qui pourra être une collection*

$this->lesParticipants = array(); *// création d’un tableau indicé de 0 à la taille du tableau (collection)*

$unPart = new Participant(); *// création d'une variable instance de la classe Participant*

$nb = count($this->lesParticipants); *// la variable nb récupère la taille du tableau $lesParticipants*

array\_push($this->lesParticipants, $unPart); *// ajoute un participant à la fin du tableau*

*// $lesParticipants*

foreach ($this->lesParticipants as $unParticipant) { *// parcours de la collection*  
 echo $unParticipant->getMel(); *// affiche le mél du participant courant*  
}

**Exemple d'utilisation du dictionnaire (tableau associatif en *PHP*)**

private $lesMateriels *; // déclaration d'une variable d'instance qui pourra être un dictionnaire*

*// (tableau associatif)*

$this->lesMateriels = array() ; *// création d’un tableau associatif*

$this->lesMateriels[$unLibelleMateriel] = $uneQuantite ; */\* ajoute dans le dictionnaire la clé de type chaîne de caractères correspondant au libelleMatériel et la valeur de type entier correspondant à la quantité demandée\*/*

if ( array\_key\_exists($uneCle, $this->lesMateriels) == true) {

*// on vérifie que la clé existe déjà dans le dictionnaire*

}

foreach ($this->lesMateriels as $cle => $valeur) {

echo $cle; *// affiche ici le libelleMateriel*

echo $valeur; *// affiche ici la quantité nécessaire pour ce libellé de matériel*

}

**parent::**méthode() **permet d'appeler la méthode présente dans la classe mère**.

**instanceof permet de connaître le type d'instance d'un objet.**

$unPart = new Participant(…) ;

if ($unPart **instanceof** Participant) {

*// traitement effectué si unPart est une instance de la classe Participant*

}