Nom :
Prénom :

Test de positionnement

Tous documents sur support papier autorisés.
L'ensemble des réponses sera à donner sur les feuilles d'énoncé.

On s'intéresse à un système simple de mise en place de questionnaires pour des apprentissages simples du cycle élémentaire comme les tables de multiplication, les conversions d'unités, ou les conjugaisons. On prévoit le modèle partiel donné à la figure 1.

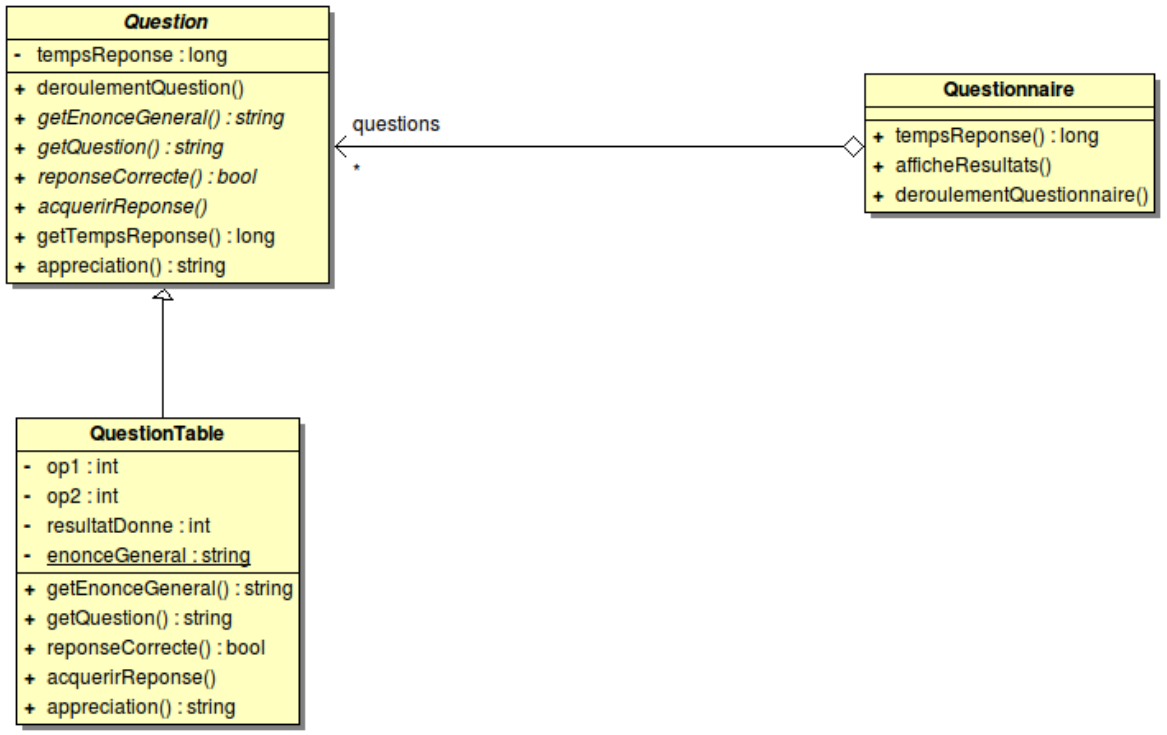


FIGURE 1 – Diagramme de classes partiel

Une question a les caractéristiques suivantes :

— la méthode `deroulementQuestion` permet le déroulement de l'interaction de l'élève face à une question : l'énoncé général est affiché (par exemple : "donne le résultat du calcul"), puis la question proprement dite est affichée (par exemple "2x6="), puis la réponse de l'élève saisie au clavier est récupérée. Le temps de réponse est calculé et stocké dans l'attribut dédié, il correspond au temps entre l'affichage de la question et la fin de la saisie de la réponse. Il sera donné en millisecondes. Tous les temps sont stockés sous forme de `long`.
— la méthode `getEnonceGeneral` retourne l'énoncé général de la question.
— la méthode `getQuestion` fournit la question sous forme de chaîne de caractères (par exemple "2x6=").
— la méthode `réponseCorrecte` indique si l'élève a répondu correctement ou pas.

- la méthode `acquerirReponse` acquiert la réponse saisie par l'élève et la stocke. Cette méthode est abstraite : chaque sous-classe concrète doit la définir de manière à stocker la réponse dans un attribut du bon type (par exemple un entier pour les questions sur les tables, une chaîne de caractères pour les questions sur les énumérations).
- la méthode `getTempsReponse` retourne le temps de réponse de l'élève. Il est à noter que si la question n'a pas encore été proposée à l'élève, le temps de réponse sera -1.
- la méthode `appreciation` retourne une chaîne de caractères contenant l'appréciation, c'est-à-dire au minimum : "bravo !" si la réponse apportée par l'élève est juste, et "Mauvaise réponse ..." si elle est fausse. Cette méthode peut être redéfinie dans les sous-classes pour préciser l'appréciation.

`QuestionTable` est une sous-classe de la classe `Question` mettant en place des questions sur les tables de multiplication. A ce titre, elle possède deux attributs `op1` et `op2` permettant de stocker les opérandes d'une multiplication, opérandes qui seront comprises entre 0 et 9. La réponse fournie par l'élève est stockée sous forme d'entier. Les opérandes seront générées aléatoirement à la construction de la question. Le constructeur correspondant n'est pas spécifié ici. Pour les questions sur les tables, on souhaite que l'appréciation inclue "ta réponse était toute proche" si la réponse de l'élève ne s'éloigne pas plus de 10% de la réponse attendue. On ne représente ici que la sous-classe `QuestionTable`, mais le système comprend aussi d'autres classes pour les autres types de question.

Un questionnaire se compose d'un ensemble de questions. On peut dérouler un questionnaire c'est-à-dire lancer le déroulement des questions l'une après l'autre, calculer (en secondes) le temps total de réponse de l'élève (si le questionnaire a été proposé à un élève), et afficher le résultat c'est-à-dire le temps de réponse total en secondes, puis l'ensemble des questions avec pour chacune l'appréciation.

Pour permettre aux élèves de vérifier leurs acquis, on souhaite également que la réalisation des questionnaires entraîne la création d'évaluation de compétences. Les évaluations sont collectées au sein d'une fiche de compétences. Une évaluation de compétences (dénotée par la classe `EvaluationCompetence`) a un nom de compétence, un état d'acquisition qui peut être acquis, en voie d'acquisition ou non acquis, et une date de réalisation de l'évaluation. Une fiche de compétences (dénotée par la classe `SyntheseCompétences`)possède un tableau des 20 dernières évaluations de compétences. On ne s'intéresse pas à la manière dont la réalisation des questionnaires va entraîner la création des évaluations de compétences.

1 Evaluation compétences et fiches compétences

On donne des portions du code Java des classes `EvaluationCompetence` et `SyntheseCompetences`, ainsi que de l'énumération `Acquisition` aux listings suivants.

Listing 1 – EvaluationCompetence.java

```
package questionnaires;

import java.util.Date;

public class EvaluationCompetence {
    private String nomCompetence;
    private Acquisition etatAcquisition;
    private Date dateEvaluation;

    public String getNomCompetence() {
        return nomCompetence;
    }

    public Acquisition getEtatAcquisition() {
        return etatAcquisition;
    }

    public Date getDateEvaluation() {
        return dateEvaluation;
    }

}
```

Listing 2 – Acquisition.java

```
package questionnaires;

public enum Acquisition {
    nonAcquis, enVoieAcquisition, acquis;
}
```

Listing 3 – SyntheseCompetences.java

```
package questionnaires;

import java.util.ArrayList;
import java.util.Collections;

public class SyntheseCompetences {
    private EvaluationCompetence competences[]=new EvaluationCompetence[nbCompetencesMax];
    private static final int nbCompetencesMax=20;

    public EvaluationCompetence[] getCompetencesParNom(String nom){
        // code que l'on suppose existant mais que l'on masque ici pour ne pas vous perturber
            !
    }

}
```

Question 1. Donnez en Java le code du constructeur paramétré de la classe `EvaluationCompetence` qui se charge d'initialiser le nom de la compétence et son état d'acquisition, ainsi que la date qui est placée à la date du jour (on se souviendra que le constructeur par défaut de la classe `Date` crée une instance de `Date` représentant la date du jour).

```
public EvaluationCompetence(String nom, Acquisition etatAcquisition) {
    this.nomCompetence = nom;
    this.etatAcquisition = etatAcquisition;
    dateEvaluation=new Date();
}
```

Question 2. Ecrivez le code Java permettant de déclarer une variable de type `EvaluationCompetence` et d'y affecter une nouvelle évaluation de compétences de nom "tables de 5" et d'état d'acquisition acquis.

```
EvaluationCompetence e=new EvaluationCompetence("tables de 5", Acquisition.acquis);
```

Question 3. Donnez en Java le code d'une méthode `ligneCompétence` de la classe `EvaluationCompetence` qui retourne une chaîne de caractères de la forme suivante :
Compétence nom-compétence **:** état-acquisition.

```
public String ligneCompetence(){
    return "Competence "+nomCompetence+" : "+etatAcquisition;
}
```

Question 4. Donnez en Java le code d'une méthode `synthese` dans la classe `SyntheseCompetences` qui retourne une chaîne de caractères composée de lignes comme suit :
Compétence nom-compétence **:** état-acquisition **Compétence** nom-compétence **:** état-acquisition **Compétence** nom-compétence **:** état-acquisition

Il y aura une ligne par évaluation de compétence contenue dans le tableau `competences`. Pour insérer un saut de ligne (retour à la ligne) dans une chaîne, on utilise le code `\n`. Par exemple pour obtenir la chaîne :
`un`
`deux`
on écrit : "un\ndeux"

```
public String synthese(){
    String result="";
    for (EvaluationCompetence c:competences){
        result+=c.ligneCompetence()+"\n";
    }
    return result;
}
```

Question 5. Ecrire en Java le code d'une méthode `enProgres` qui retourne un booléen stipulant que l'élève est en progrès pour une compétence donnée (en fait un nom de compétence, par exemple "tables de 5"). Pour simplifier, pour savoir si un élève est en progrès, on regardera juste ses deux dernières évaluations pour cette compétence et on regardera si l'élève a progressé de l'une à l'autre. Pour implémenter cette méthode, on pourra utiliser la méthode `getCompetencesParNom` qui prend en paramètre le nom d'une compétence et retourne un tableau contenant toutes les évaluations de compétences pour cette compétence. Le tableau est plein (pas de case vide ou nulle), et trié par date croissante d'évaluation (évaluation la plus ancienne en case d'index 0 et la plus récente en dernière

case). S'il n'y a qu'une seule évaluation disponible, alors on retournera arbitrairement vrai. Pour comparer des littéraux d'énumération (ici des états d'acquisition), on pourra se baser sur le fait que l'on accède depuis une énumération à l'ordre dans lequel sont déclarés les littéraux, grâce à la méthode `ordinal()`. Dans notre cas, si on déclare :

```
Acquisition a1=Acquisition.acquis;
Acquisition a2=Acquisition.nonAcquis;
Acquisition a3=Acquisition.enVoieAcquisition;
```

alors `a1.ordinal()` retourne 0, `a2.ordinal()` retourne 2 et `a3.ordinal()` retourne 1.

```
public boolean enProgres(String nom){
    boolean result=false;// oubli dans le sujet si pas d'evaluation : faux
    EvaluationCompetence[] resultats=getCompetencesParNom(nom);
    if (resultats.length==1){result=true;}
    else if(resultats.length>1){
        Acquisition dernier=resultats[resultats.length-1].getEtatAcquisition();
        Acquisition avantdernier=resultats[resultats.length-2].getEtatAcquisition();
        result= dernier.ordinal()>avantdernier.ordinal();
    }
    return result;
}
```

2 Questions et questionnaires

Question 6. Donnez le code Java de la méthode `déroulementQuestion` de la classe `Question`. Pour le chronométrage, on fera appel à la méthode statique `currentTimeMillis` présente dans la classe `java.lang.System` qui retourne le temps "système" à l'instant d'appel en millisecondes (sous forme de long). On rappelle que le comportement de cette méthode est décrit en début d'énoncé. On pourra utiliser toutes les méthodes présentes dans le modèle de la figure 1 sans les définir au préalable.
Réponse à la question 6 :

```
public void deroulementQuestion(){
    System.out.println(getEnonceGlobal());
    System.out.print(getQuestion());
    long chrono1 = java.lang.System.currentTimeMillis() ;
    acquerirReponse();
    long chrono2 = java.lang.System.currentTimeMillis() ;
    tempsReponse=chrono2-chrono1;
    System.out.println("");
}
```

Question 7. Ecrivez l'ensemble du code permettant d'implémenter la méthode `appreciation` dans les classes `Question` et `QuestionTable`. On pourra utiliser la méthode statique `abs` de la classe `Math` qui calcule la valeur absolue d'un nombre.
Réponse à la question 7 :

```
public abstract class Question{
public String appreciation(){
        String resultat;
        if (reponseCorrecte()){
            resultat="Bravo !";
        } else resultat="Mauvaise reponse ...";
        return resultat;     }}
public class QuestionTable{
public String appreciation(){
        String resultat=super.appreciation();
        float diff=Math.abs((float)reponseDonnee-(float)reponseAttendue())/(float)
            reponseAttendue();
        if (!reponseCorrecte()&&diff<0.05){
            resultat+=" mais tu n'es pas loin !";
        }
        return resultat;}}
```

Question 8. Expliquez la modélisation de l'énoncé général des questions (attribut `enonceGeneral:String` et méthodes `getEnonceGeneral():String` de `Question` et de `QuestionTable`).
Réponse à la question 8 :

On a un attribut de classe dans la sous classe : attribut de classe (et pas d'instance) car la valeur est partagée par toutes les instances de QuestionTable, n'est pas dans Question car la valeur n'est pas partagée par toutes les questions (pas les questions de conversion par exemple). On a une méthode d'accès à l'énoncé général dans Question car toutes les questions ont bien un énoncé. En revanche elle n'est pas implémentable dans Question car l'énoncé est introduit dans les sous-classes. Donc la définition concrete de cette méthode d'accès est dans QuestionTable.

Question 9. Donnez la signature et l'implémentation du constructeur de `QuestionTable`. On utilisera la classe `java.util.Random`. Cette classe est munie d'une méthode non statique `nextInt(int bound)` qui retourne un entier aléatoire compris entre 0 et bound.
Réponse à la question 9 :

```
public QuestionTable(){
        Random r=new Random();
        op1=r.nextInt(10);
        op2=r.nextInt(10);
    }
```

on pourrait faire un appel à super qui positionne tempsReponse à -1, mais on peut aussi le laisser implicite
on n'a pas de consigne pour reponseDonnee

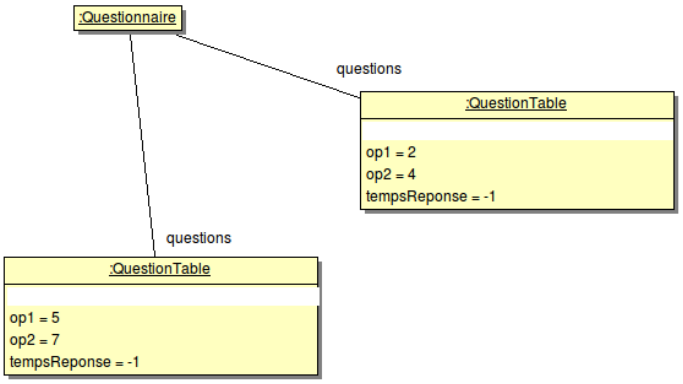
Question 10. Donnez en Java la traduction de l'association entre la classe `Questionnaire` et la classe `Question`.
Réponse à la question 10 :

```
private Vector<Question> questions=new Vector<Question>(); // ou new dans les constr
//ou un tableau car nous n'avions pas vu les vector ou arraylist :
private Question[] questions=new Question[30];// 30 est ici arbitraire
```

Question 11. Donnez en Java le code de la méthode `tempsReponse` de la classe `Questionnaire`. Ce temps est donné en secondes. C'est le cumul des temps de réponse à chaque question. Dès lors que l'une des questions a pour temps de réponse -1 (c'est-à-dire que l'élève n'y a pas répondu), la méthode `tempsReponse` de la classe `Questionnaire` doit retourner -1.
Réponse à la question 11 :

```
private long tempsReponse(){
        long cumul=0;
        for (Question q:questions){
            long temps= q.getTempsReponse();
            if (temps==-1) return -1;
            else {
            cumul+=temps;
            }
        }
        return cumul;
    }
}
```

Question 12. Donnez un diagramme d'objets représentant un questionnaire contenant deux questions sur les tables, non encore traitées par un élève, et correspondant respectivement aux calculs 2*4 et 5*7.
Réponse à la question 12 :



cette question pouvait être traitée avec des valeurs d'attributs plutôt que des liens car nous n'avons pas encore fait de liens en TD.

Question 13. Modélisez et implémentez un constructeur pour les questionnaires qui construit un questionnaire d'une taille voulue et d'un type de question (tables, conjugaisons, ou conversions) voulu, passés en paramètres. On supposera disposer d'une énumération `TypeQuestion` pour les types de question possibles (tables, conjugaisons, ou conversions).
Réponse à la question 13 :

```
public Questionnaire(TypeQuestion type, int taille){
        if (type==TypeQuestion.table){
            for (int i=0;i<taille;i++){
                Question q;
                q=new QuestionTable();
                questions.add(q);
            }
    //idem pour les autres types de question
        }
    }
```

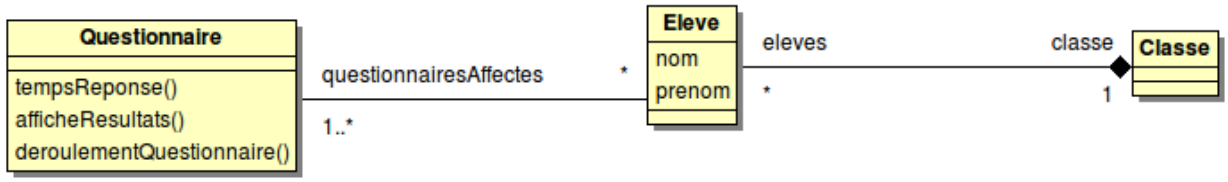
Question 14. Donnez en Java le code d'un `main` permettant d'obtenir un questionnaire à 2 questions sur les tables.
Réponse à la question 14 :

```
Questionnaire quest=new Questionnaire(TypeQuestion.table, 2);
```

3 Classes et élèves

On souhaite modéliser maintenant des classes et des élèves. Un élève a un nom et un prénom. Une classe se compose d'élèves (un élève n'appartient qu'à une seule classe). Chaque élève d'une classe se voit affecter un ensemble non vide de questionnaires.

Question 15. Modélisez les classes et les élèves avec un diagramme de classes intégrant uniquement les éléments de l'énoncé.
Réponse à la question 15 :



vous pouviez répondre avec des attributs au lieu des associations car nous n'avions pas fait d'association en TD-TP.