Examen terminal session 2 - Eléments de correction

Tous documents sur support papier autorisés. Durée : 2h.

Ce sujet développe quelques éléments d'un logiciel dédié au suivi de projets et de leurs versions successives. Une version met en place différentes fonctionnalités.

Question 1. Ecrivez en Java une énumération `Phase` décrivant les différentes états des versions d'un logiciel (prototype, alpha, bêta, release, stable).

Réponse à la question 1 :

```
public enum Phase {                          // 0,5 pt
    prototype, alpha, beta, release, stable  // 0,5 pt
}
```

Question 2. Ecrivez en Java une interface `IFonctionnalité`. A ce stade, une fonctionnalité est décrite par différentes informations :

- un nom,
- une année (l'année de développement de la version),
- un tableau associatif (`HashMap`) associant à une année (la clef), le coût horaire de développement (la valeur associée). Ce tableau est public et sa référence est constante. Son contenu peut cependant être modifié.
- une durée de développement prévisionnelle (DDP), donnée en heures
- un coût de développement prévisionnel (CDP), qui est le produit de la durée de développement prévisionnelle par le coût horaire. Si l'année de la version n'est pas enregistrée dans le tableau de coût horaires, un message d'erreur est affiché et le coût est nul.
- une méthode statique permettant d'inscrire dans le tableau associatif les coûts horaires des années 2017 et 2018, valant respectivement 150 pour 2017 et 155 pour 2018..

Réponse à la question 2 :

```
public interface IFonctionnalite {  //0,25 pt
    String getNom();     //0,25 pt
    int getAnnee();      //0,25 pt
    HashMap<Integer, Double> coutsHorairesParAnnee = new HashMap<>();  //1 pt
    int getDDP(); //0,25 pt
    default double CDP(){   //1 pt
       if (IFonctionnalite.coutsHorairesParAnnee.get(this.getAnnee())==null)
        {System.out.println("coût de l'année inconnu"); return 0;}
       else
        return IFonctionnalite.coutsHorairesParAnnee.get(this.getAnnee())*this.getDDP();
    }
   static void init(){ //1 pt
   IFonctionnalite.coutsHorairesParAnnee.put(2017, 150.0);
   IFonctionnalite.coutsHorairesParAnnee.put(2017, 155.0);
   }
}
```

Question 3. Ecrivez une première version de la classe `Fonctionnalité` qui respecte `IFonctionnalité`. Cette classe doit pouvoir être instanciée. Ecrivez la classe avec :

— les attributs nécessaires pour implémenter l'interface,
— les méthodes cohérentes avec l'interface,
— un constructeur avec paramètres.

Réponse à la question 3 :

```
public class Fonctionnalite implements IFonctionnalite { // 1 pt
    // pas abstract car elle doit pouvoir être instanciée

    // attributs 0,5 pt
    private String nom;
    private int annee;
    private int ddp;

    // methodes 0,75 pt
    @Override
    public String getNom() {
        return this.nom;
    }

    @Override
    public int getAnnee() {
        return this.annee;
    }

    @Override
    public int getDDP() {
        return this.ddp;
    }

    // constructeur 0,75 pt

    public Fonctionnalite(String nom, int annee, int ddp) {
        this.nom = nom;
        this.annee = annee;
        this.ddp = ddp;
    }
}
```

Question 4. Ecrire l'entête et les attributs d'une classe `Version`. Une version est décrite par un numéro, une phase, et un ensemble de fonctionnalités. Ajoutez-lui une méthode pour ajouter une fonctionnalité à une version et une méthode pour calculer le coût de développement prévisionnel.

Réponse à la question 5 :

```
public class Version {

    private String numero; // 0,25 pt
    private Phase phase; // 0,25 pt
    private ArrayList<IFonctionnalite> listeFct = new ArrayList<>(); // 1 pt

    public void ajoute(IFonctionnalite fct){ // 1pt
     if (this.listeFct.contains(fct))
     System.out.println("fonctionnalité déjà présente");
     else
     this.listeFct.add(fct);
    }
```

```
    public double coutDP(){ // 1,5 pt
     double coutTotal = 0;
     for (IFonctionnalite fct : this.listeFct)
     coutTotal += fct.CDP();
     return coutTotal;
    }
}
```

Question 5. Ecrire l'entête et les attributs d'une classe `VersionImplémentation`. Une version d'implémentation est une version qui est de plus décrite par un système d'exploitation cible et le coût de déploiement. Le calcul du coût prévisionnel est modifié pour intégrer le coût du déploiement.

Réponse à la question 5 :

```
public class VersionImplementation extends Version { // 1 pt

    private String systemeExploitation; // 0,25 pt
    private double coutDeploiement; // 0,25 pt

    public double coutDP(){ // 1,5 pt
     return super.coutDP() + this.coutDeploiement;
    }
```

Question 6. (4 pt, feuille annexe) Dessinez un diagramme de classes UML correspondant aux classes et à l'interface décrites précédemment. Les attributs dont le type est une classe doivent y apparaître sous forme d'associations. De plus ajoutez dans ce diagramme la notion de projet (un projet existe sous différentes versions) et de maquette utilisateur. Une maquette utilisateur est une version comprenant des spécifications (chaîne de caractères) et des types d'utilisateurs visés.

Question 7. (1 pt, feuille annexe) Dessinez un diagramme d'instances significatif comprenant un projet avec deux versions dont une maquette. Imaginez la valuation des attributs.