TP11 Correction

Matière : ATELIER DE PROGRAMMATION OBJET | **Classes : SEM21**

Exercice1

```
package ath;

public interface Resultat {
  public float getMeilleur();
  public String getUnite();
}

package ath;

public abstract class Athlete implements Resultat {
  protected String nom;
  protected String prenom;
  protected int age;
  protected float dernierResultat;
  protected String unite;
  protected float meilleurResultat;

  public Athlete(String nom, String prenom, int age, String unite) {
    this.nom = nom;
    this.prenom = prenom;
    this.age = age;
    this.unite = unite;
    dernierResultat = 0;
    meilleurResultat = 0;
  }

  public void afficher() {
    System.out.println("Nom:" + nom);
    System.out.println("Prénom:" + prenom);
    System.out.println("Age:" + age);
    System.out.println("Dernier résultat:" + dernierResultat);
    System.out.println("Unité:" + unite);
    System.out.println("Meilleur résultat:" + meilleurResultat);
  }

  public abstract void setDernierResultat(float dernierResultat);

  @Override
  public float getMeilleur() {
    return meilleurResultat;
  }

  @Override
  public String getUnite() {
    return unite;
  }

}
```

```
package ath;

public class AthleteLancement extends Athlete {

  private String typeCharge;
  private int poidsCharge;

  public AthleteLancement(String nom, String prenom, int age, String typeCharge,
int poidsCharge) {
    super(nom, prenom, age, "Mètre");
    dernierResultat = 0;
    meilleurResultat = 0;
    this.typeCharge = typeCharge;
    this.poidsCharge = poidsCharge;
  }

  @Override
  public void afficher() {
    super.afficher();
    System.out.println("Type charge:" + typeCharge);
    System.out.println("Poids charge:" + poidsCharge);
  }

  @Override
  public void setDernierResultat(float dernierResultat) {
    this.dernierResultat = dernierResultat;
    if (dernierResultat > meilleurResultat)
      meilleurResultat = dernierResultat;
  }
}
package ath;

public class AthleteCourse extends Athlete {

  private int longueurParcours;
  private boolean avecObstacle;
  private int nbObstacle;

  public AthleteCourse(String nom, String prenom, int age, int longueurParcours) {
    super(nom, prenom, age, "Seconde");
    dernierResultat = Float.POSITIVE_INFINITY;
    meilleurResultat = Float.POSITIVE_INFINITY;

    this.longueurParcours = longueurParcours;
    this.avecObstacle = false;
    this.nbObstacle = 0;
  }
  public AthleteCourse(String nom, String prenom, int age, int longueurParcours,
int nbObstacle) {
    super(nom, prenom, age, "Seconde");
    this.longueurParcours = longueurParcours;
    this.avecObstacle = true;
    this.nbObstacle = nbObstacle;
  }
  @Override
  public void afficher() {
    super.afficher();
    System.out.println("Longueur parcours:" + longueurParcours);
    System.out.println("Avec Obstacle?:" + avecObstacle);
    System.out.println("Nb Obstacle:" + nbObstacle);
  }
```

```
  @Override
  public void setDernierResultat(float dernierResultat) {
    this.dernierResultat = dernierResultat;
    if (dernierResultat < meilleurResultat)
      meilleurResultat = dernierResultat;
  }

  public int getLongueurParcours() {
    return longueurParcours;
  }
}
//Equipe version tableau
package ath;

public class EquipeTab {

   private String nom;
   private Athlete[] tAthlete;


   public EquipeTab(String nom) {
      this.nom = nom;
      tAthlete = new Athlete[10];
   }


   public void ajouter(Athlete athlete) {
      int i = 0;
      while (i < tAthlete.length)
         if (tAthlete[i] != null)
            i++;
      if (i < tAthlete.length)
         tAthlete[i] = athlete;
      else
         System.out.println("Le tableau est plein!");
   }


   public void afficherAgeMoyen() {
      int somme = 0;
      float moyenne;
      for (int i = 0; i < tAthlete.length; i++)
         somme += tAthlete[i].age;
      moyenne = ((float) somme) / tAthlete.length;
      System.out.println("Age moyen:" + moyenne);

   }


package ath;

import java.util.Vector;

public class Equipe {

  private String nom;
  private Vector<Athlete> vAthlete;

  public Equipe(String nom) {
    this.nom = nom;
```

```
    vAthlete = new Vector<Athlete>();
  }

  public void ajouter(Athlete athlete) {
    vAthlete.addElement(athlete);
  }

  public void afficherAgeMoyen() {
    int somme = 0;
    float moyenne;
    for (int i = 0; i < vAthlete.size(); i++)
      somme += vAthlete.elementAt(i).age;
    moyenne = ((float) somme) / vAthlete.size();
    System.out.println("Age moyen:" + moyenne);
  }

  public void afficherMeilleurResultatCourse(int longueurParcours) {
    boolean estPremier = true;
    float meilleurResultat = 0;
    for (int i = 0; i < vAthlete.size(); i++)
      if (vAthlete.elementAt(i) instanceof AthleteCourse) {
        AthleteCourse ac = (AthleteCourse) vAthlete.elementAt(i);
        if (ac.getLongueurParcours() == longueurParcours)
          if (estPremier) {
            meilleurResultat = ac.meilleurResultat;
            estPremier = false;
          } else if (ac.meilleurResultat > meilleurResultat)
            meilleurResultat = ac.meilleurResultat;
      }
    System.out.println("Meilleur résultat pour " + longueurParcours + ":" +
meilleurResultat);
  }

}
//Equipe version Vector
package ath;

import java.util.Vector;

public class EquipeVect {

   private String nom;
   private Vector<Athlete> vAthlete;
   public EquipeVect(String nom) {
      this.nom = nom;
      vAthlete = new Vector<Athlete>();
   }


   public void ajouter(Athlete athlete) {
      vAthlete.addElement(athlete);
   }

   public void afficherAgeMoyen() {
      int somme = 0;
      float moyenne;
      for (int i = 0; i < vAthlete.size(); i++)
         somme += vAthlete.elementAt(i).age;
      moyenne = ((float) somme) / vAthlete.size();
      System.out.println("Age moyen:" + moyenne);
   }

   public void afficherMeilleurResultatCourse(int longueurParcours) {
      boolean estPremier = true;
```

```
      float meilleurResultat = 0;
      for (int i = 0; i < vAthlete.size(); i++)
         if (vAthlete.elementAt(i) instanceof AthleteCourse) {
            AthleteCourse ac = (AthleteCourse) vAthlete.elementAt(i);
            if (ac.getLongueurParcours() == longueurParcours)
               if (estPremier) {
                  meilleurResultat = ac.meilleurResultat;
                  estPremier = false;
               } else if (ac.meilleurResultat > meilleurResultat)
                  meilleurResultat = ac.meilleurResultat;
         }
      System.out.println("Meilleur résultat pour " + longueurParcours + ":"
            + meilleurResultat);
   }

}
```

```
package ath;

public class TestEquipe {
  public static void main(String[] args) {
    Equipe eq=new Equipe("PO Sport");
    AthleteCourse ac1=new AthleteCourse("Ben Mohamed", "Saleh", 21, 100);
    ac1.setDernierResultat(11);
    eq.ajouter(ac1);

    AthleteLancement al=new AthleteLancement("Ben Saleh", "Ali", 22, "Disque", 6);
    eq.ajouter(al);

    AthleteCourse ac2=new AthleteCourse("Ben Salem", "Samir", 23, 100);
    ac2.setDernierResultat(10);
    eq.ajouter(ac2);

    eq.ajouter(new AthleteCourse("Ben Saleh", "Amir", 23, 1000));
    eq.ajouter(new AthleteCourse("Ben Ahmed", "Salem", 23, 100));

    eq.afficherAgeMoyen();
    eq.afficherMeilleurResultatCourse(100);
  }

}
```

Exercice2

```
public interface Statistique {
  public int getTotalFeuilleChargee();
  public int getTotalFeuilleImprimee();
}
public class Imprimante {
  protected String marque;
  protected int capacite;
  protected int nbFeuilleChargee;
  protected int totalImpression;
  protected int niveauEncre;
  public Imprimante(String marque, int capacite) {
    this.marque = marque;
    this.capacite = capacite;
    nbFeuilleChargee = 0;
    totalImpression = 0;
    niveauEncre = 100;
  }
  public String getMarque() {
    return marque;
  }
  public int getCapacite() {
    return capacite;
  }
  public int getNbFeuilleChargee() {
    return nbFeuilleChargee;
  }
  public int getTotalImpression() {
    return totalImpression;
  }
  public int getNiveauEncre() {
    return niveauEncre;
  }
  public void afficher() {
    System.out.println("Marque :" + marque);
    System.out.println("Capacite :" + capacite);
    System.out.println("Nb Feuille Chargée :" + nbFeuilleChargee);
    System.out.println("Total Impression :" + totalImpression);
    System.out.println("Niveau Encre :" + niveauEncre);
  }
  public void chargerFeuille(int nbFeuille) {
    if (nbFeuilleChargee + nbFeuille <= capacite)
      nbFeuilleChargee += nbFeuille;
    else
      nbFeuilleChargee = capacite;
  }
  public void chargerEncre() {
    niveauEncre = 100;
  }
      public void imprimer(int nbFeuille) {
    int min;
    if (nbFeuilleChargee <= nbFeuille)
      min = nbFeuilleChargee;
    else
      min = nbFeuille;
    nbFeuilleChargee -= min;
    totalImpression += min;
    int quantiteEncre = min * 2;
    if (niveauEncre <= quantiteEncre)
      niveauEncre = 0;
    else
      niveauEncre -= quantiteEncre;
  }
}
```

```
public class ImprimanteCouleur extends Imprimante {
  private int niveauEncreCouleur;
  public ImprimanteCouleur(String marque, int capacite) {
    super(marque, capacite);
    niveauEncreCouleur = 100;
  }
  public int getNiveauEncreCouleur() {
    return niveauEncreCouleur;
  }
  public void afficher() {
    super.afficher();
    System.out.println("Niveau Encre Couleur :" + niveauEncreCouleur);
  }
  public void chargerEncreCouleur() {
    niveauEncreCouleur = 100;
  }
  public void imprimer(int nbFeuille) {
    int min;
    if (nbFeuilleChargee <= nbFeuille)
      min = nbFeuilleChargee;
    else
      min = nbFeuille;
    nbFeuilleChargee -= min;
    totalImpression += min;
    int quantiteEncre = min;
    if (niveauEncre <= quantiteEncre)
      niveauEncre = 0;
    else
      niveauEncre -= quantiteEncre;
    if (niveauEncreCouleur <= quantiteEncre)
      niveauEncreCouleur = 0;
    else
      niveauEncreCouleur -= quantiteEncre;
  }
}
import java.util.Vector;
public class Imprimerie implements Statistique {
  private String nom;
  private Vector<Imprimante> vImp;
  public Imprimerie(String nom) {
    this.nom = nom;
    vImp = new Vector<Imprimante>();
  }
  public void ajouter(Imprimante imp) {
    vImp.addElement(imp);
  }
  public void supprimer(String marque) {
    for (int i = 0; i < vImp.size(); i++) {
      if (vImp.elementAt(i).getMarque().equals(marque)) {
        vImp.removeElement(i);
        return;
      }
    }
  }
  public void supprimerTout() {
    vImp.removeAllElements();
  }
  public void rechercher(String marque) {
    for (int i = 0; i < vImp.size(); i++) {
      if (vImp.elementAt(i).getMarque().equals(marque)) {
        vImp.elementAt(i).afficher();
        return;
      }
    }
  }
```

```
    public void afficher() {
    System.out.println("Nom :" + nom);
    for (int i = 0; i < vImp.size(); i++)
      vImp.elementAt(i).afficher();
  }
  public void charger() {
    for (int i = 0; i < vImp.size(); i++) {
      vImp.elementAt(i).chargerEncre();
      if (vImp.elementAt(i) instanceof ImprimanteCouleur)
        ((ImprimanteCouleur) vImp.elementAt(i)).chargerEncreCouleur();
    }
  }
  public void chargerFeuille(int indice, int nbFeuille) {
    vImp.elementAt(indice).chargerFeuille(nbFeuille);
  }
  public int getNbImprimante() {
    return vImp.size();
  }
  public int getNbImprimanteCouleur() {
    int nbImprimanteCouleur = 0;
    for (int i = 0; i < vImp.size(); i++) {
      if (vImp.elementAt(i) instanceof ImprimanteCouleur)
        nbImprimanteCouleur++;
    }
    return nbImprimanteCouleur;
  }
  public void imprimer(int indice, int nbFeuille) {
    vImp.elementAt(indice).imprimer(nbFeuille);
  }
  @Override
  public int getTotalFeuilleChargee() {
    int somme = 0;
    for (int i = 0; i < vImp.size(); i++)
      somme += vImp.elementAt(i).getNbFeuilleChargee();
    return somme;
  }
  @Override
  public int getTotalFeuilleImprimee() {
    int somme = 0;
    for (int i = 0; i < vImp.size(); i++)
      somme += vImp.elementAt(i).getTotalImpression();
    return somme;
  }
}
public class Test {
  public static void main(String[] args) {
    Imprimerie imprimerie;
    imprimerie = new Imprimerie("SuperPrint");
    Imprimante imp;
    imp = new Imprimante("HP", 150);
    imprimerie.ajouter(imp);
    ImprimanteCouleur impC;
    impC = new ImprimanteCouleur("Lexmark", 100);
    imprimerie.ajouter(impC);
    imprimerie.afficher();
    imprimerie.chargerFeuille(1, 50);
    imprimerie.imprimer(1, 20);
    System.out.println("Nb Imprimante : " + imprimerie.getNbImprimante());
    System.out.println("Total Feuille Chargée: "
        + imprimerie.getTotalFeuilleChargee());
    System.out.println("Total Feuille imprimée: "
        + imprimerie.getTotalFeuilleImprimee());
    imprimerie.supprimerTout();
  }
}
```