Examen Java
Juin 2016
Professeur Abdessamad Belangour
SMI S6

Nous souhaitons faire une petite application console qui gère les trains, leurs wagons et leurs conducteurs. Un train se compose d'un certain nombre de Wagons qui peuvent être des de passagers, de Minerai (phosphates, …..) ou de marchandises. Ainsi, un train peut être :

- Un train de Passagers s'il est composé de wagons passagers seulement,
- Un train de Minerai s'il est composé de wagons Minerai seulement,
- Un train de Marchandise s'il est composé de wagons Marchandise seulement,
- Un train de Passagers/Marchandise s'il est composé de wagons passagers et de wagons Marchandise,

Vous devez fournir uniquement les parties de code demandées sur votre feuille :

- Sans réécrire toutes les classes mais juste les méthodes ou constructeurs demandées !!
- En respectant les signatures des méthodes et des constructeurs fournis !!

```
public abstract class Wagon {
   private String code;
   private String description;
     // on suppose que les constructeurs, les getters et setters et toString() sont fournis
}
```

```
public class WagonMarchandise extends Wagon{
   private final static float volumeMax=72; // en mètres cube;
   private float volume;
   private static int nombreWagonsMarchandise=0;

   public WagonMarchandise(String code, String description, float volume) {  // à fournir  }

    // on suppose que le getter et setter du volume est fourni
   // getters et ou setters pour  VolumeMax et nombreWagonMarchandise à fournir
   // toString à fournir
}
```

```
public class WagonMinerai extends Wagon{
   private final static float tonnageMax=22;// en tonnes
   private float tonnage;
   private static int nombreWagonsMinerai=0;

// on suppose que les constructeurs et les getters et setters et toString sont fournis
```

```
}
```

```
public class WagonPassagers extends Wagon{
   private final static int nombrePassagersMax=50;
   private int nombrePassagers;
   private static int nombreWagonsPassagers=0;
  // on suppose que les constructeurs et les getters et setters et toString sont fournis
}
```

```
public enum TypeTrain {  Marchandise,    Passagers,    PassagersMarchandise,    Minerai  }
```

```
public class TrainWagonIncompatiblesException extends Exception{
   public TrainWagonIncompatiblesException(String message) {     super(message);    }  }
```

```
public class Personnel {
   private String matricule;
   private String nom;
   private String prenom;
     // on suppose que les constructeurs et les getters et setters et toString sont fournis
}
```

```
public class Train {
   private String code;
   private Personnel conducteur;
   private TypeTrain type;
   private int longueur;
   private ArrayList<Wagon> wagons;
   public Train(String code, Personnel conducteur, TypeTrain type) {    // à fournir    }
   public Train(String code, Personnel conducteur, TypeTrain type,
         ArrayList<Wagon> wagons)          {              // à fournir       }
   // getter et ou setter pour longueur à fournir
   // on suppose que les autres getters et setters sont fournis

   public void ajouterWagonPassagers(String code, String description, int nombrePassagers)

     throws   TrainWagonIncompatiblesException { // cette méthode est fournie   }

   public void ajouterWagonMinerai(String code, String description, float tonnage) throws
TrainWagonIncompatiblesException {   // à fournir    }


   public void ajouterWagonMarchandise(String code, String description, float volume) throws
TrainWagonIncompatiblesException {  // on suppose que le code de cette méthode est fourni    }

  public void ajouterWagon(Wagon wagon) throws TrainWagonIncompatiblesException {
      // à fournir    }

   public boolean supprimerWagon(String code) {     // à fournir      }

  // on suppose que toString est fourni

// la méthode afficherCharge() qui affiche la quantité, le nombre ou le volume de ce que transporte le train
selon son type.
```

```
public void AfficherCharge() {      // à fournir   }
```

Rappel des Questions :

i. Dans la classe **WagonMarchandise** :
   0. Donner le code du **constructeur**
   1. Donner le code du getter et ou setter pour l'attribut ***VolumeMax***
   2. Donner le code du getter et ou setter pour l'attribut ***nombreWagonMarchandise***
   3. Donner le code de la méthode ***toString***

ii. Dans la classe **Train** :
   4. Donner le code du **premier constructeur**
   5. Donner le code du ***deuxième constructeur***
   6. Donner le getter et ou setter pour l'attribut ***longueur***
   7. Donner le code de la méthode ***ajouterWagonMinerai*** (Attention : l'ajout d'un wagon d'un type dans un train qui n'est pas du même type déclenche une exception)
   8. Donner le code de la méthode ***ajouterWagon (***même remarque***)***
   9. Donner le code de la méthode ***supprimerWagon***
   10. Donner le code de la méthode ***afficherCharge*** de la classe Train qui affiche la quantité, le nombre ou le volume de ce que transporte le train selon son type.

```java
package Serie5.ma;

public class Personnel {
	private String matricule;
	 private String nom;
	 private String prenom;
	public Personnel(String matricule, String nom, String
prenom) {
		super();
		this.matricule = matricule;
		this.nom = nom;
		this.prenom = prenom;
	}

}
```

```java
package Serie5.ma;

public enum TypeTrain {
	Marchandise, Passagers, PassagersMarchandise, Minerai

}
```

```java
package Serie5.ma;

public abstract class Wagon {

	private String code;
	private String description;
	public Wagon(String code, String description) {

		this.code = code;
		this.description = description;
	}
	public String getCode() {
		return code;
	}
	public void setCode(String code) {
		this.code = code;
	}

}
```

```java
package Serie5.ma;

public class WagonMarchandise  extends Wagon {
```

```
      private final static float volumeMax=72; // en mètres cube;
      private float volume;
      private static int nombreWagonsMarchandise=0;
      public WagonMarchandise(String code, String description,
float volume) {

            super (code, description);
            this.volume=volume;
       }
     public float getVolume()
     {
          return (volume);
     }


     public void setVolume(float volume) {
          this.volume = volume;
     }
     public static float getVolumemax() {
          return volumeMax;
     }
     public static int getNombreWagonsMarchandise() {
          return nombreWagonsMarchandise;
     }
     public static void setNombreWagonsMarchandise(int
nombreWagonsMarchandise) {
          WagonMarchandise.nombreWagonsMarchandise =
nombreWagonsMarchandise;
     }
     @Override
     public String toString() {
          return "WagonMarchandise [" + super.getCode()+ "
volume= " + volume + " volumeMax= "+ volumeMax+ "
nombreWagonsMarchandise "+ nombreWagonsMarchandise+ "]";
     }


}
package Serie5.ma;

public class WagonMinerai  extends Wagon{

     private final static float tonnageMax=22;// en tonnes
      private float tonnage;
      private static int nombreWagonsMinerai=0;
     public WagonMinerai(String code, String description, float
tonnage) {
```

```
        super(code, description);
        this.tonnage = tonnage;
    }


    public void setTonnage(float tonnage) {
        this.tonnage = tonnage;
    }


    public float getTonnage() {
        return tonnage;
    }


    @Override
    public String toString() {
        return "WagonMinerai [tonnage=" + tonnage + "]";
    }
```

```
}
package Serie5.ma;

public class WagonPassagers  extends Wagon {

    private final static int nombrePassagersMax=50;
     private int nombrePassagers;
     private static int nombreWagonsPassagers=0;
    public WagonPassagers(String code, String description, int
nombrePassagers) {
        super(code, description);
        this.nombrePassagers = nombrePassagers;
    }
    public int getNombrePassagers() {
        // TODO Auto-generated method stub
        return nombrePassagers;
    }
    @Override
    public String toString() {
```

```
            return "WagonPassagers [nombrePassagers=" +
nombrePassagers + ", getNombrePassagers()=" +
getNombrePassagers()
                    + ", getCode()=" + getCode() + "]";
      }



}
package Serie5.ma;

import java.util.ArrayList;

public class Train {
      private String code;
      private Personnel conducteur;
      private TypeTrain type;
      private int longueur;
      private ArrayList<Wagon> wagons;


      public Train(String code, Personnel conducteur, TypeTrain
type) {

            this.code = code;
            this.conducteur = conducteur;
            this.type = type;
            this.longueur = 0;
            this.wagons = new ArrayList< >();
      }


      public Train(String code, Personnel conducteur, TypeTrain
type, ArrayList<Wagon> wagons) {

            this.code = code;
            this.conducteur = conducteur;
            this.type = type;
            this.longueur = wagons.size();
            this.wagons = new ArrayList <>(wagons);
      }


      public int getLongueur() {
            return longueur;
      }
```

```
    public void setLongueur(int longueur) {
        this.longueur = longueur;
    }


     public void ajouterWagonPassagers(String code, String
description, int nombrePassagers) throws
TrainWagonIncompatiblesException {
          if (this.type != TypeTrain.Passagers) { throw new
               TrainWagonIncompatiblesException("Type wagon non
autorisé pour ce type de train"); }
               wagons.add(new WagonPassagers(code, description,
nombrePassagers));
               this.longueur++;
          }

     public void ajouterWagonMinerai(String code, String
description, float tonnage) throws
TrainWagonIncompatiblesException
     {
          if (this.type != TypeTrain.Minerai) { throw new
               TrainWagonIncompatiblesException("Type wagon non
autorisé pour ce type de train"); }
               wagons.add(new WagonMinerai(code, description,
tonnage));
               this.longueur++;


     }
     public void ajouterWagonMarchandise(String code, String
description, float volume) throws
    TrainWagonIncompatiblesException {
          if (this.type != TypeTrain.Marchandise) { throw new
               TrainWagonIncompatiblesException("Type wagon non
autorisé pour ce type de train"); }
               wagons.add(new WagonMarchandise(code,
description, volume));
               this.longueur++;
          }


     public void ajouterWagon(Wagon wagon) throws
TrainWagonIncompatiblesException {
          if (((this.type == TypeTrain.Minerai)&& (wagon
instanceof WagonMinerai))||
```

```
                    ((this.type== TypeTrain.Marchandise) &&
(wagon instanceof WagonMarchandise)) ||
                    ((this.type ==
TypeTrain.PassagersMarchandise) && ((wagon instanceof
WagonMarchandise)
                              || this.type ==
TypeTrain.Passagers &&(wagon instanceof WagonPassagers))))
          {
          wagons.add(wagon);
          this.longueur++;
          }
          else
          {
               throw new TrainWagonIncompatiblesException("Type
wagon non autorisé pour ce type de train"); }
          }

     public boolean supprimerWagon(String code) {
          boolean supprimé = false;
          for (Wagon w : wagons) {
          if (w.getCode().equals(code)) {
          wagons.remove(w); supprimé = true; this.longueur--;
          //break;
          return supprimé;}
          }
          return supprimé;

          }
     public void AfficherCharge() {
          float volumeTotal = 0;
          float tonnageTotal = 0;
          int nombreTotal = 0;
          switch (type) {
          case Minerai:
          for (Wagon w : wagons) {
          tonnageTotal += ((WagonMinerai) w).getTonnage();
          }
          System.out.println("le tonnage total est " +
tonnageTotal); break;
          case Marchandise:
          for (Wagon w : wagons) {
          volumeTotal += ((WagonMarchandise) w).getVolume();
          }
          System.out.println("le volume total est " +
volumeTotal); break;
          case Passagers:
          for (Wagon w : wagons) {
```

```
            nombreTotal += ((WagonPassagers)
w).getNombrePassagers();
            }
            System.out.println("le nombre total est " +
nombreTotal); break;
            case PassagersMarchandise:
            for (Wagon w : wagons) {
            if (w instanceof WagonPassagers) {
            nombreTotal += ((WagonPassagers)
w).getNombrePassagers();
            } else {
            volumeTotal += ((WagonMarchandise) w).getVolume();
            }
            }
            System.out.println("le nombre total est " +
nombreTotal + " le volume total est" +
          volumeTotal); break;
            }


      }
}
```

```
package Serie5.ma;

import java.util.ArrayList;

public class Test_Train {

     public static void main(String[] args) {

          ArrayList<Wagon> wagon=new ArrayList();

          Wagon wagon1= new WagonMarchandise("R325", "train",
154);
          Wagon wagon2= new WagonMarchandise("L385", "train",
200);
          Wagon wagon3= new WagonMinerai("M325", "train", 300);
          Wagon wagon4= new WagonMinerai("N325", "train", 400);
          Wagon wagon5= new WagonPassagers("O325", "train", 500);
          Wagon wagon6= new WagonPassagers("P325", "train", 600);
          Wagon wagon7= new WagonMarchandise("Q325", "train",
700);
          Wagon wagon8= new WagonPassagers("Q25", "train", 800);
```

```
            //wagon.add(wagon5);
            wagon.add(wagon1);
            wagon.add(wagon2);

            //wagon.add(wagon3);
            //wagon.add(wagon4);

            //wagon.add(wagon6);
            wagon.add(wagon7);
            //wagon.add(wagon8);
            wagon.forEach(System.out::println);
            /*for (Wagon w: wagon)
            {
                 System.out.println (w);
            }*/
            Personnel c1=new Personnel("5879", "ali", "rafik");

            Train train =new Train ("T154", c1, TypeTrain.Minerai);
            try {
            //train.ajouterWagonMarchandise("Q25", "train", 800);
            //System.out.println(train.getLongueur());
            train.ajouterWagon(wagon3);
            System.out.println(train.getLongueur());


            }
            catch (TrainWagonIncompatiblesException e)
            {
                 System.out.println(e.getMessage());
            }

Personnel c2=new Personnel("5879", "mohame2", "rafik");

            Train train1 =new Train ("T154", c1,
TypeTrain.Marchandise, wagon);
            train1.AfficherCharge();

      }


}
```