TD 6 : Résolution de la surcharge

Objectifs pédagogiques : maîtrise des difficultés de l'héritage spécifiques à Java.

6.1 Exercice 1

On considère le code suivant :

```java
public class A {
  public void methode(A a) {
    System.out.println("methode a");
  }
}
```

```java
public class B extends A {
  private int i;
  public B(){i=0;}
  public void methode(A a) {
    System.out.println("methode a dans b");
  }
  public void methode(B b) {
    System.out.println("methode b");
  }
  public void modification() {
    i++;
  }
}
```

```java
public static void main(String[] args) {
  A[] tableau = {new A(), new B(), new B()};
  A a1 = tableau[0];
  B b1 = new B();
  B b2 = tableau[1];

  for(A a:tableau)
    a.methode(a1);

  b1.modification();
  b2.modification();
  b2.modification();
}
```

⇒ Une faute s'est glissée dans le code ci-dessus et empêche la compilation, **donnez une correction**.
⇒ **Donnez les affichages lors de l'exécution de ce code une fois corrigé.**
⇒ **Donnez le diagramme d'objets à l'issue de l'exécution.**

6.2 Exercice 2

On considère le code suivant :

```java
public class Fruit {
   public void af(Fruit f) {
      System.out.print(1);
```

```
   }
}
public class Kiwi extends Fruit {
   @Override
   public void af( Fruit f) {
      System .out. print (2);
   }

   public void af(Kiwi k) {
      System .out. print (3);
   }
}
```

Ainsi que le main suivant :

```
public class Appli {
   public static void main( String [] a) {
      Fruit f = new Fruit ();
      Fruit fk = new Kiwi ();
      Kiwi k = new Kiwi ();

      f.af(f); f.af(fk); f.af(k);

      fk.af(f); fk.af(fk); fk.af(k);

      k.af(f); k.af(fk); k.af(k);
   }
}
```

⇒ Donnez les affichages produits par ce programme.

6.3 Exercice 3

On considère une application dans laquelle un garagiste est amené à faire des comparaisons entre des voitures stockées dans son garage, qui peuvent être soit des voitures normales, soit des voitures diesel. Il s'intéresse à la similarité entre les voitures.

Pour cela, on suppose que l'on dispose des classes `Voiture` et `VoitureDiesel` ci-dessous :

```
package pobj.vroum;

public class Voiture {
  private final String modele;
  private final String couleur;
  private final int longueur;

  public Voiture(String modele, String couleur, int longueur) {
    this.modele = modele;
    this.couleur = couleur;
    this.longueur = longueur;
  }
  public String getModele() {
    return modele;
  }
  public String getCouleur() {
    return couleur;
  }
  public int getLongueur() {
    return longueur;
```

```
  }
  public boolean estSimilaire(Voiture v) {
    System.out.println("[appel de Voiture.estSimilaire(Voiture v)]");
    return getModele().equals(v.getModele())
      && getCouleur().equals(v.getCouleur())
      && getLongueur()==v.getLongueur();
  }
}
```

```
package pobj.vroum;

public class VoitureDiesel extends Voiture {
  private final boolean commonRail;

  public VoitureDiesel(String modele, String couleur, int longueur, boolean commonRail) {
    super(modele,couleur,longueur);
    this.commonRail = commonRail;
  }
  public boolean isCommonRail() {
    return commonRail;
  }
  public boolean estSimilaire(VoitureDiesel v) {
    System.out.println("[Appel de VoitureDiesel.estSimilaire(VoitureDiesel v)]");
    return super.estSimilaire(v)
      && isCommonRail()==v.isCommonRail();
    }
}
```

6.3.1 Synthèse des informations de type

⇒ **Donnez**, sous la forme d'un tableau récapitulatif, **les types complets synthétisés par le compilateur Java pour les classes `Voiture` et `VoitureDiesel`** (les héritages, les attributs et les signatures des méthodes disponibles).

⇒ **Représentez la structure du diagramme UML** (juste les noms de classe et la relation d'héritage) pour cet exemple **et indiquez la relation complète de sous-typage**. Quelle est la relation entre les types `Voiture`, `VoitureDiesel` et `Object` (`java.lang.Object`) ?

6.3.2 Invocations mystérieuses

⇒ Dans toutes les questions suivantes, **pour chaque invocation à une méthode de nom `estSimilaire()`**, on indiquera la signature présélectionnée par le compilateur. On indiquera ensuite la valeur calculée par la machine virtuelle sur ces invocations. Pour les cas difficiles, on donnera une représentation de la mémoire sous la forme de diagrammes d'objets.

Question 1

```
Voiture deuxCv1 = new Voiture("2cv", "bordeau", 3);
Voiture deuxCv2 = new Voiture("2cv", "bordeau", 3);
Voiture deuxCv3 = new Voiture("2cv", "bordeau", 4);
deuxCv1.estSimilaire(deuxCv1) ?
deuxCv1.estSimilaire(deuxCv2) ?
deuxCv1.estSimilaire(deuxCv3) ?
```

Question 2

```
VoitureDiesel twinga1 = new VoitureDiesel("twinga", "beige", 3, true);
VoitureDiesel twinga2 = new VoitureDiesel("twinga", "beige", 3, true);
VoitureDiesel twinga3 = new VoitureDiesel("twinga", "beige", 3, false);
twinga1.estSimilaire(twinga1) ?
twinga1.estSimilaire(twinga2) ?
twinga1.estSimilaire(twinga3) ?
```

Question 3

```
Voiture twinga4 = new Voiture("twinga", "beige", 3);
Voiture twinga5 = new Voiture("twinga", "rouge", 3);
twinga1.estSimilaire(twinga4) ?
twinga4.estSimilaire(twinga1) ?
twinga1.estSimilaire(twinga5) ?
twinga5.estSimilaire(twinga1) ?
```

Question 4

```
Voiture twinga6 = new VoitureDiesel("twinga", "beige", 3, true);
twinga1.estSimilaire(twinga6) ?
twinga6.estSimilaire(twinga1) ?
twinga3.estSimilaire(twinga6) ?
```

Question 5

Comment faire pour que le programme « réponde » correctement ?
⇒ **Proposez des modifications dans les classes `Voiture` et/ou `VoitureDiesel`** pour corriger le programme.

6.4 Exercice 3

On considère le code suivant :

```
package pobj.vroum;
public class Garage {
  private List<Voiture> mesVoitures;
  [...]
  public boolean checkPresence(Voiture v) {
    return mesVoitures.contains(v);
  }
  public void add(Voiture v) {
    mesVoitures.add(v);
  }
  public static void main(String args[]){
    Garage g = new Garage();
    Voiture v1 = new Voiture("twinga", "bleu", 3);
    Voiture v2 = new Voiture("twinga", "vert", 3);
    g.add(v1);
    g.add(v2);
```

```
    Voiture v3 = new Voiture("twinga", "vert", 3);
    System.out.println(g.checkPresence(v3));
  }
}
```

```
package pobj.vroum;
public class Voiture {
  [...]
  public boolean equals(Voiture v){
    if (!couleur. equals(v.couleur)) return false;
    if (!modele. equals(v.modele)) return false;
    if (longueur!=v.longueur) return false;
    return true;
  }
}
```

⇒ **Quel est l'affichage obtenu ? D'où vient le problème ? Comment faut-il le corriger ?**