

Génie Logiciel TD n° 5

Méta-Propriétés UML et correspondance avec Java

Objectif du TD

Maîtriser les méta-propriétés UML (propriétés des propriétés UML) :

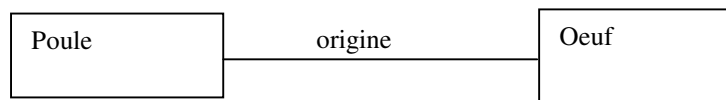
- propriété de classe et de propriété d'instance,
- visibilité des propriétés,
- attribut intrinsèque et attribut associatif.

Correspondance entre UML et POO :

- associations UML et références Java,

Exercice TRES FACILE

Cet exercice illustre le lien entre deux classes Java, la correspondance entre association UML et attributs références Java. On part de deux classes, Poule et Oeuf associée par l'association "origine". La poule a pour origine l'œuf qui a pour origine la poule... Pour simplifier, on part du diagramme de classe ci-dessous et on suppose que l'association binaire "origine" est un-un.



- Quelle est la manière de traduire le rôle Poule-Oeuf de l'association "origine" en Java ?
- Comment s'appellent les attributs monOeuf et maPoule ?
- Soit le programme Java suivant :

```

class Poule {
    public Oeuf monOeuf ;
    public void maMethode() { monOeuf.maMethode() ; }
}
class Oeuf { public :
    public Poule maPoule ;
    public void maMethode() { ; }
    public static void main() {
        Poule p = new Poule(); Oeuf o = new Oeuf();
        p->monOeuf = o; o->maPoule = p;
        p->maMethode(); o->maMethode();
    }
}
  
```

Exercice “ PROPRIETE DE CLASSE ET PROPRIETE D’INSTANCE ”

1) Que signifie “ PROPRIETE DE CLASSE ” ? “ PROPRIETE D’INSTANCE ” ?

Pour chaque propriété des classes UML suivantes, nous avons volontairement omis de la souligner quand cela était nécessaire.

2) En vous aidant du nom de la propriété et de sa signification, indiquer si la propriété est "d'instance" ou bien "de classe".

Objet
nombreObjets : int ; mesObjets : Liste ; numéro : int ;

Château
muraille : Muraille ; listeTours : Liste ; listeChateaux : Liste ;
void imprimerMuraille() ; void imprimerTours() ; void imprimerChateaux() ; void imprimer() ;

On peut souligner void imprimerChateaux() ; ou pas ; quelles sont les deux significations ?

On peut souligner void imprimer () ; ou pas ; quelles sont les deux significations ?

Exercice MOT CLE STATIC

- 1) Que signifie “ PROPRIETE DE CLASSE ” ? “ PROPRIETE D’INSTANCE ” ?
- 2) Quel mot-clé Java utilise-t-on pour spécifier une propriété de classe ? et d’instance ?

Soit le programme JAVA suivant :

```

class R {
    public static R notreR;
    public static int j;
    public int i;
    public R monR;
    public R(int a) { i = a; monR = null; j++; }
    public static void p(R r) {
        j = 0;
        notreR = new R(0);
        m(notreR);
    }
    public static void q(R r) {
        System.out.print("R.notreR = "); n(notreR);
        System.out.println("R.j = " + j);
    }
    public static void m(R r) {
        System.out.println("R, i = " + r.i);
    }
    public static void n(R r) {
        m(r);
        System.out.print("monR = ");
        if (r.monR!=null) r.monR.m(r.monR);
        else System.out.println("null");
    }
}

class S {
    public static void main() {
        R.p(null);
        R r1 = new R(1);
        R.notreR.monR = r1;
        R r2 = new R(2);
        System.out.print("r2 = ");
        R.n(r2);
        r1.monR = r2;
        System.out.print("r1 = ");
        R.n(r1);
        R.q(r1);
    }
}

```

- a) Donner la sortie du programme et dessiner les objets et leurs liens correspondant à l’exécution du programme.
- b) Le mot-clé ‘static’ est correctement placé en ce qui concerne les attributs mais pas en ce qui concerne les méthodes. Reprogrammer ce programme pour que sa sortie toujours la même, que les mot-clés ‘static’ soient partout correctement placés, et enfin que les paramètres inutiles soient enlevés.
- c) Donner le pour et le contre de la phrase : “ le constructeur est une méthode d’instance ”.

Exercice “ ABCDE ”

Soit le programme Java suivant :

```
// abcde.java
import java.io.*;
class A {
public A monA ;
    public void maMethode() { imprimer(" A ") ; }
    public void saMethode() { monA.maMethode(); }
    public void imprimer(String s) { System.out.println(s) ; }
}
class B extends A{
    public C monC ;
    public void saMethode() { monC.maMethode() ; }
}
class C extends A {
    public B monB ;
    public void maMethode() { imprimer(" C ") ; }
    public void saMethode() { monB.maMethode() ; }
}
class D extends B{
    public E monE;
    public void maMethode() { imprimer(" D ") ; }
    public void saMethode() { monE.maMethode() ; }
}
class E extends B {
    public D monD ;
    public void maMethode() { imprimer(" E ") ; }
}
class M {
    public static void main(String argv[]) {
        A a = new A();
        B b = new B();
        C c = new C();
        D d = new D();
        E e = new E();

        a.monA = b; b.monA = c; c.monA = d; d.monA = e; e.monA = a;
        b.monC = c; c.monB = b; d.monC = c; e.monC = c;
        d.monE = e; e.monD = d;

        a.maMethode();b.maMethode();c.maMethode();d.maMethode();e.maMethode();
        a.saMethode();b.saMethode();c.saMethode();d.saMethode();e.saMethode();
    }
}
```

1) Dessiner les diagrammes de généralisation, de classe-association et d’instances correspondant.

2) Donner la sortie du programme.

Exercice “ Reverse engineering de Java à UML ”

Le “ Reverse engineering ” est une technique de développement informatique qui consiste à concevoir un logiciel à partir du code source d’un logiciel existant. “ engineering ” signifie concevoir et “ reverse ” signifie que l’on travaille en sens inverse au sens habituel. Cet exercice est un exemple de reverse engineering. A partir d’un code Java, vous devez retrouver la conception UML.

```
import java.io.*;

class Objet {
    public String nom;
    public Objet(String n) { nom = n; }
    public Objet() { this(""); }
    public void imprimer() {
        System.out.println(toString());
    }
    public String toString() {
        return toIdent();
    }
    public String toIdent() {
        return nom;
    }
    public String toEtreVivant() {
        return "";
    }
}

class Demon extends Objet {
    public double puissance;
    public Liste mesLoges;
    public static Liste mesDemons = new Liste();
    public Demon(String n, double p) {
        super(n);
        puissance = p;
        mesLoges = new Liste();
        mesDemons.ajouter(this);
    }
    public void occuperEtreVivant(EtreVivant ev) {
        ev.maPremiereLoge.mesDemons.ajouter(this);
        mesLoges.ajouter(ev.maPremiereLoge);
    }
    public void occuperEtreHumain(EtreHumain ev) {
        ev.maSecondeLoge.mesDemons.ajouter(this);
        mesLoges.ajouter(ev.maSecondeLoge);
    }
    public String toString() {
        return nom + " puissance: " + puissance
            + " mesEV: " + mesLoges.toEtreVivant();
    }
}

class EtreVivant extends Objet {
    public Loge maPremiereLoge;
    public EtreVivant(String n) {
        super(n);
        maPremiereLoge = new Loge("", this);
    }
    public String toString() {
        return toIdent() + " lere loge: "
            + maPremiereLoge.mesDemons.toIdent();
    }
}
```

```

class EtreHumain extends EtreVivant {
    public Loge maSecondeLoge;
    public EtreHumain(String n) {
        super(n);
        maSecondeLoge = new Loge("", this);
    }
    public String toString() {
        return super.toString() + " 2eme loge: "
            + maSecondeLoge.mesDemons.toIdent();
    }
}

class Loge extends Objet {
    public EtreVivant monEtreVivant;
    public Liste mesDemons;
    public Loge(String n, EtreVivant ev) {
        super(n);
        monEtreVivant = ev;
        mesDemons = new Liste();
    }
    public String toString() {
        return "monEV: " + monEtreVivant.toIdent()
            + " mesDemons: " + mesDemons.toIdent();
    }
    public String toEtreVivant() {
        return monEtreVivant.toIdent();
    }
}

class Liste extends Objet {
    private Element debut;
    private Element courant;
    public Liste() { debut = null; courant = null; }
    public void ajouter(Objet machin) {
        Element nouveau = new Element();
        nouveau.suivant = debut;
        nouveau.contenu = machin;
        debut = nouveau;
        courant = nouveau;
    }
    public void imprimer() {
        Element cour = debut;
        while (cour!=null)
            { cour.contenu.imprimer(); cour = cour.suivant;}
    }
    public String toIdent() {
        String s = new String();
        Element cour = debut;
        while (cour!=null)
            { s = s + cour.contenu.toIdent() + " ";
              cour = cour.suivant; }
        return s;
    }
    public String toEtreVivant() {
        String s = new String();
        Element cour = debut;
        while (cour!=null)
            { s = s + cour.contenu.toEtreVivant() + " ";
              cour = cour.suivant; }
        return s;
    }
}

class Element {
    public Element suivant;
    public Objet contenu;
    public Element() { suivant = null; contenu = null; }
}

```

```

class Principal {
    public static void main(String argv[]) {

        System.out.println("Big Bang");

        System.out.println("Apparition des démons");
        Demon oRca = new Demon("oRca", 9.5);
        Demon eTranger = new Demon("eTranger", 5.5);
        Demon sOrath = new Demon("sOrath", 9);
        Demon cAracole1 = new Demon("cA1", 1);
        Demon cAracole2 = new Demon("cA2", 1);
        Demon pRocessionnaire1 = new Demon("pR1", 2);
        Demon pRocessionnaire2 = new Demon("pR2", 2);
        Demon hYpnotique1 = new Demon("hY1", 3);
        Demon hYpnotique2 = new Demon("hY2", 3);

        System.out.println("Apparition du premier etre vivant");
        EtreVivant dino = new EtreVivant("dino");
        sOrath.occuperEtreVivant(dino);
        hYpnotique1.occuperEtreVivant(dino);
        cAracole1.occuperEtreVivant(dino);
        pRocessionnaire1.occuperEtreVivant(dino);
        dino.imprimer();

        System.out.println("Apparition d'une Albspina");
        EtreHumain albspina = new EtreHumain("albspina");
        sOrath.occuperEtreVivant(albspina);
        hYpnotique2.occuperEtreVivant(albspina);
        cAracole2.occuperEtreVivant(albspina);
        pRocessionnaire2.occuperEtreVivant(albspina);
        albspina.imprimer();

        System.out.println("Apparition d'un Luisant");
        EtreHumain luisant = new EtreHumain("luisant");
        sOrath.occuperEtreVivant(luisant);
        hYpnotique2.occuperEtreVivant(luisant);
        cAracole2.occuperEtreVivant(luisant);
        pRocessionnaire2.occuperEtreVivant(luisant);
        oRca.occuperEtreHumain(luisant);
        luisant.imprimer();

        System.out.println("Apparition du premier Humain");
        EtreHumain adam = new EtreHumain("adam");
        sOrath.occuperEtreVivant(adam);
        hYpnotique1.occuperEtreVivant(adam);
        cAracole1.occuperEtreVivant(adam);
        pRocessionnaire1.occuperEtreVivant(adam);
        hYpnotique2.occuperEtreHumain(adam);
        cAracole2.occuperEtreHumain(adam);
        pRocessionnaire2.occuperEtreHumain(adam);
        adam.imprimer();

        System.out.println("Apparition de Mistral");
        EtreHumain mistral = new EtreHumain("mistral");
        oRca.occuperEtreVivant(mistral);
        oRca.occuperEtreHumain(mistral);
        mistral.imprimer();

        System.out.println("Occupation des demons:");
        Demon.mesDemons.imprimer();
        System.out.println("Fin du monde");
    }
}

```

- 1) Dessiner le de généralisation UML.
- 2) Dessiner un diagramme de classes UML. Placer les ordres de multiplicité.
- 3) Dessiner un diagramme d'objets UML.
- 4) Donner la sortie du programme.