TD #5-6
Diagrammes de séquences

Exercice 1. Caisse enregistreuse

Le déroulement normal du traitement d'un passage en caisse est le suivant :

- Un client arrive à la caisse avec des articles à payer.
- Le caissier enregistre les codes barres de chaque sortie d'article.
- La caisse affiche le prix de chaque article ainsi que son libellé.
- Lorsque tous les achats sont enregistrés, le caissier signale la fin de la vente.
- La caisse affiche le total du montant des achats.
- Le client peut présenter au caissier des réductions pour certains articles. Le caissier saisit alors les réductions et la caisse affiche le montant après réduction.
- Le client choisit de payer en liquide : le caissier encaisse l'argent reçu ; la caisse indique l'argent à rendre au client, le caissier rend la monnaie si nécessaire.
- La caisse enregistre la vente et imprime le ticket.
- Lorsque le paiement est terminé, la caisse transmet les informations sur le nombre d'articles vendus au système de gestion des stocks.

Proposez un diagramme de séquence décrivant le passage « nominal » d'un client en caisse.

Exercice 2. Protocoles de sécurité

Décrire l'attaque man-in-the-middle exploitant une faille connue du protocole initial Needham-Schröder Public Key (NSPK).

Pour mémoire :
Alice --> Bob {Na,Alice}_PK_A
Bob --> Alice {Nb, Na}_PK_B
Alice --> Bob {Nb}_PK_A

Exercice 3. Distributeur de billets

Décrire par un diagramme de séquences les interactions entre l'utilisateur et un distributeur de billets de banque. On modélisera le scénario suivant : l'utilisateur ne se trompe pas de code PIN, et il demande une somme d'argent dont le retrait est autorisé par sa banque.

Exercice 4. Flot d'exécution d'un interpréteur

Proposez un diagramme de séquence permettant de décrire le flot d'exécution du programme au principal ci-dessous.

```
public interface Expr {
  double interpreter();
}

public class Nombre implements Expr {
  double valeur;
  public Nombre(double v) { valeur = v; }
  public double interpreter() {
    return valeur;
  }
}

public class Multiplie implements Expr {
  Expr op1, op2;
  public Multiplie(Expr e1, Expr e2) {
    op1 = e1;
    op2 = e2;
  }
  public double interpreter() {
    return op1.interpreter() * op2.interpreter();
  }
}
```

```
public class Plus implements Expr {
  Expr op1, op2;
  public Multiplie(Expr e1, Expr e2) {
    op1 = e1;
    op2 = e2;
  }
  public double interpreter() {
    return op1.interpreter() + op2.interpreter();
  }
}

public class Program {
  public static void main(String[] args) {
    // e == (2 + 4) * 7
    Expr e1 = new Nombre(2.0);
    Expr e2 = new Nombre(4.0);
    Expr e3 = new Nombre(7.0);
    Expr e4 = new Plus(e1,e2);
    Expr e = new Multiplie(e3,e4);
    System.out.println(e.interpreter());
  }
}
```

Exercice 5. Flot d'exécution d'un visiteur

Même exercice avec le visiteur ci-après.

```
public interface Expr {
    Object accept(ExprVisiteur e);
}
```

```
public interface ExprVisiteur {
    Object visit(Nombre n);
    Object visit(Multiplie m);
    Object visit(Plus p);
}
```

```
public class Nombre implements Expr {
   double valeur;
   public Nombre(double v) { valeur = v; }
   public double getValeur() { return valeur; }
   public Object accept(ExprVisiteur ev) {
      return ev.visit(this);
   }
}

public class Multiplie implements Expr {
   Expr op1, op2;
   ...
   public Object accept(ExprVisiteur ev) {
     return ev.visit(this);
  }
}

public class Plus implements Expr {
   Expr op1, op2;
   ...
   public Object accept(ExprVisiteur ev) {
      return ev.visit(this);
   }
}
```

```
public class Interpreter implements ExprVisitor {

   Object visit(Nombre n) {
      return n.getValue();
   }

   Object visit(Multiplie m) {
      return (double) m.op1.accept(this) *
             (double) m.op2.accept(this);
   }

   Object visit(Plus p) {
      return (double) m.op1.accept(this) +
             (double) m.op2.accept(this);
   }
}


public class Program {
   public static void main(String[] args) {
      // e == (2 + 4) * 7
      Expr e1 = new Nombre(2.0);
      Expr e2 = new Nombre(4.0);
      Expr e3 = new Nombre(7.0);
      Expr e4 = new Plus(e1,e2);
      Expr e = new Multiplie(e3,e4);
      Interpreter i = new Interpreter();
      System.out.println(e.accept(i));
   }
}
```