

TD 1 - Introduction au Système Unix et à Java

Proteine.java - Demo 1.1

```
class Proteine {

    int nbPos;    // Attribut : Nombre d'AA charges positivement
    int nbNeg;    // Attribut : Nombre d'AA charges negativement
    int charge;   // Attribut : Charge nette a pH=7

    void calcCharge() {           // Methode
        charge = nbPos - nbNeg;  // instruction(s) de la methode
    }                             // fin de la methode

} // fin de la declaration de la classe Proteine
```

Milieu.java - Demo 1.1

```
class Milieu {

    public static void main(String args[]) {

        Proteine prot1 = new Proteine(); // Une instance de la classe Proteine

        prot1.nbPos = 4;                // Donne une valeur a un attribut
        prot1.nbNeg = 7;                // Donne une valeur a un attribut

        prot1.calcCharge();             // Appel d'une methode de la classe Proteine

        System.out.println("Charge =" + prot1.charge + " a pH=7.");

    } // fin de la methode main()

} // fin de la declaration de la classe Milieu
```

Milieu.java - Exercice 1.1

```
class Milieu {

    public static void main(String args[]) {

        Proteine prot1 = new Proteine(); // Declaration de
        Proteine prot2 = new Proteine(); // 2 instances de la classe Proteine

        prot1.nbPos = 4;
        prot1.nbNeg = 7;
        prot1.calcCharge();
        System.out.println("Proteine 1 : charge " + prot1.charge + " a pH=7");

        prot2.nbPos = 3;
        prot2.nbNeg = 1;
        prot2.calcCharge();
        System.out.println("Proteine 2 : charge " + prot2.charge + " a pH=7");

    }

}
```

Proteine.java - Exercice 1.2

```
class Proteine {

    int nbPos;    // Attribut : Nombre d'AA charges positivement
    int nbNeg;    // Attribut : Nombre d'AA charges negativement
    int charge;   // Attribut : Charge nette a pH=7
    int x,y;      // Attributs : Position

    void calcCharge() {
        charge = nbPos - nbNeg;
    }

}
```

```

void deplaceADroite() {
    x = x + 1;
}
void deplaceAGauche() {
    x = x - 1;
}
void deplaceEnBas() {
    y = y + 1;
}
void deplaceEnHaut() {
    y = y - 1;
}
}

```

Milieu.java - Exercice 1.2

```

class Milieu {

    public static void main(String args[]) {

        Proteine prot1 = new Proteine(); // Declaration de
        Proteine prot2 = new Proteine(); // 2 instances de la classe Proteine

        prot1.nbPos = 4;
        prot1.nbNeg = 7;
        prot1.calcCharge();
        prot1.x = 12;
        prot1.y = 21;
        System.out.println("Proteine 1 : charge "+prot1.charge+" a pH=7");
        System.out.println("Proteine 1 en "+prot1.x+" ; "+prot1.y);

        prot2.nbPos = 3;
        prot2.nbNeg = 1;
        prot2.calcCharge();
        prot2.x = 32;
        prot2.y = 45;
        System.out.println("Proteine 2 : charge "+prot2.charge+" a pH=7");
        System.out.println("Proteine 2 en "+prot2.x+" ; "+prot2.y);

        prot1.deplaceADroite();
        prot2.deplaceADroite();

        System.out.println("Proteine 1 en "+prot1.x+" ; "+prot1.y);
        System.out.println("Proteine 2 en "+prot2.x+" ; "+prot2.y);

        prot1.deplaceAGauche();
        prot1.deplaceEnHaut();
        prot2.deplaceEnBas();

        System.out.println("Proteine 1 en "+prot1.x+" ; "+prot1.y);
        System.out.println("Proteine 2 en "+prot2.x+" ; "+prot2.y);

    }

}

```

Proteine.java - Exercice 1.3

```

class Proteine {

    int nbPos;
    int nbNeg;
    int charge;
    int x,y;

    void calcCharge() {
        charge = nbPos - nbNeg;
    }

    void deplaceADroite() {
        x = x + 1;
    }
}

```

```

void deplaceAGauche() {
    x = x - 1;
}
void deplaceEnBas() {
    y = y + 1;
}
void deplaceEnHaut() {
    y = y - 1;
}

void afficheCharge() {
    System.out.println("Charge = "+charge+" a pH=7.");
}

void affichePosition() {
    System.out.println("Coordonnees : "+x+" ; "+y);
}
}

```

Milieu.java - Exercice 1.3

```

class Milieu {

    public static void main(String args[]) {

        Proteine prot1 = new Proteine();
        Proteine prot2 = new Proteine();

        prot1.afficheCharge();

        prot1.nbPos = 4;
        prot1.nbNeg = 7;
        prot1.calcCharge();
        prot1.x = 120;
        prot1.y = 210;
        prot1.afficheCharge();
        prot1.affichePosition();

        prot2.nbPos = 3;
        prot2.nbNeg = 1;
        prot2.calcCharge();
        prot2.x = 32;
        prot2.y = 45;
        prot2.afficheCharge();
        prot2.affichePosition();

        prot1.deplaceADroite();
        prot2.deplaceADroite();
        prot1.deplaceAGauche();
        prot1.deplaceEnHaut();
        prot2.deplaceEnBas();

        prot1.affichePosition();
        prot2.affichePosition();

    }
}

```


TD 2 - Encapsulation - Graphisme avec AWT

Absurd.java - Demo 2.1

```

class Absurd {
    public static void main(String args[]) {
        Proteine prot1 = new Proteine();

        prot1.nbPos = -4;           // Rien ne nous en empeche !
        prot1.nbNeg = 7;
        prot1.calcCharge();

        System.out.println(prot1.nbPos+" charges positives");
        System.out.println(prot1.nbNeg+" charges négatives");
        prot1.afficheCharge();

        System.out.println(); // Pour sauter une ligne

        prot1.nbPos = 4;
        prot1.charge = -2;
                                // On n'aurait pas oublie quelque chose, la ?

        System.out.println(prot1.nbPos+" charges positives");
        System.out.println(prot1.nbNeg+" charges négatives");
        prot1.afficheCharge();
    }
}

```

Proteine.java - Demo 2.2 et Exercice 2.1 (encapsulation 1)

```

class Proteine {
    private int nbPos;
    private int nbNeg;
    private int charge;
    private int x,y;

    public void setPos(int val) {
        nbPos = val;
    }

    public int getPos() {
        return nbPos;
    }

    public void setNeg(int val) {
        nbNeg = val;
    }

    public int getNeg() {
        return nbPos;
    }

    public void calcCharge() {
        charge = nbPos - nbNeg;
    }

    public int getCharge() {
        return charge;
    }

    public void afficheCharge() {
        System.out.println("Charge = "+charge+" a pH=7.");
    }

    public void setX(int abs) {
        x = abs;
    }

    public int getX() {
        return x;
    }
}

```

```

public void setY(int ord) {
    y = ord;
}

public int getY() {
    return y;
}

public void deplaceADroite() {
    x = x + 1;
}
public void deplaceAGauche() {
    x = x - 1;
}
public void deplaceEnBas() {
    y = y + 1;
}
public void deplaceEnHaut() {
    y = y - 1;
}

public void affichePosition() {
    System.out.println("Coordonnees : "+x+" ; "+y);
}
}

```

Milieu.java - Exercice 2.1 (encapsulation 1)

```

class Milieu {

    public static void main(String args[]) {

        Proteine prot1 = new Proteine();
        Proteine prot2 = new Proteine();

        prot1.afficheCharge();

        prot1.setPos(4);
        prot1.setNeg(7);
        prot1.calcCharge();
        prot1.setX(120);
        prot1.setY(210);
        prot1.afficheCharge();
        prot1.affichePosition();

        prot2.setPos(3);
        prot2.setNeg(1);
        prot2.calcCharge();
        prot2.setX(32);
        prot2.setY(45);

        System.out.println();
        prot2.afficheCharge();
        prot2.affichePosition();

        prot1.deplaceADroite();
        prot2.deplaceADroite();
        prot1.deplaceAGauche();
        prot1.deplaceEnHaut();
        prot2.deplaceEnBas();

        System.out.println();

        prot1.affichePosition();
        prot2.affichePosition();

    }
}

```

Proteine.java - Demo 2.3 (encapsulation 2)

```

class Proteine {
    private int nbPos = 0;
    private int nbNeg = 0;
    private int x,y;

    // public void calcCharge() {
    //     charge = nbPos - nbNeg;
    // }

    public int getCharge() {
        return nbPos - nbNeg;
    }

    public void afficheCharge() {
        System.out.println("Charge = +(nbPos-nbNeg)+\" a pH=7.\"");
    }

    // LE RESTE DE LA CLASSE NE CHANGE PAS
}

```

Milieu.java - Demo 2.3 (encapsulation 2)

```

class Milieu {
    public static void main(String args[]) {
        Proteine prot1 = new Proteine();
        Proteine prot2 = new Proteine();

        prot1.setPos(4);
        prot1.setNeg(7);
        // prot1.calcCharge(); On ne peut plus le faire !
        prot1.setX(120);
        prot1.setY(210);
        prot1.afficheCharge();
        prot1.affichePosition();

        prot2.setPos(3);
        prot2.setNeg(1);
        // prot2.calcCharge(); On ne peut plus le faire !
        prot2.setX(32);
        prot2.setY(45);
        System.out.println();
        prot2.afficheCharge();
        prot2.affichePosition();

        prot1.deplaceADroite();
        prot2.deplaceADroite();
        prot1.deplaceAGauche();
        prot1.deplaceEnHaut();
        prot2.deplaceEnBas();
        System.out.println();

        prot1.affichePosition();
        prot2.affichePosition();
    }
}

```

Proteine.java - Demo 2.4 et Exercice 2.2 (graphique)

```

import java.awt.*; // pour le graphisme

class Proteine {
    // Methode a ajouter
    public void dessine(Graphics g) {
        g.setColor (Color.red);
        g.fillOval (x-9, y-9, 19, 19);
    }
}

// Méthode à modifier... et modifier de même deplaceAGauche() deplaceEnHaut() deplaceEnBas()
public void deplaceADroite() {
    x = x + 10;
}

```

Milieu.java - Demo 2.4 et Exercice 2.2 (graphique)

```
import java.awt.*;

class Milieu {

    public static void delay (int ms) {
        long time = System.currentTimeMillis();
        while (System.currentTimeMillis() - time < ms);
    }

    public static void main(String args[]) {

        Frame fenetre = new Frame ("milieu");

        fenetre.setSize (600, 600);
        fenetre.setVisible (true);

        Graphics g = fenetre.getGraphics();

        Proteine prot1 = new Proteine();
        Proteine prot2 = new Proteine();

        delay (200);
        g.setColor (Color.white);
        g.fillRect (0,0,600,600);

        prot1.afficheCharge();

        prot1.setPos(4);
        prot1.setNeg(7);
        prot1.setX(120);
        prot1.setY(210);
        prot1.afficheCharge();
        prot1.affichePosition();
        prot1.dessine(g);

        prot2.setPos(3);
        prot2.setNeg(1);
        prot2.setX(32);
        prot2.setY(45);
        System.out.println();
        prot2.afficheCharge();
        prot2.affichePosition();
        prot2.dessine(g);

        prot1.deplaceADroite();
        prot2.deplaceADroite();
        prot1.deplaceAGauche();
        prot1.deplaceEnHaut();
        prot2.deplaceEnBas();

        delay (2000);
        System.out.println();
        prot1.affichePosition();
        prot2.affichePosition();

        g.setColor (Color.white);
        g.fillRect (0,0,600,600);
        prot1.dessine(g);
        prot2.dessine(g);
    }
}
```

Proteine.java - Exercice 2.3 (couleur)

```
import java.awt.*; // pour le graphisme

class Proteine {

    // Methode modifiée
    public void dessine(Graphics g) {
        g.setColor (new Color(127-10*getCharge(),0,127+10*getCharge()));
        g.fillOval (x-9, y-9, 19, 19);
    }
}
```


TD 3 - Tests

Proteine.java - Demo 3.1 (vérification)

```
// ON MODIFIE AINSI LA METHODE setPos() :
```

```
public void setPos(int val) {
    if ( val < 0 )
        val = 0;
    nbPos = val;
}
```

Proteine.java - Exercice 3.1 (tests)

```
// Méthodes modifiées
```

```
public void setNeg(int val) {
    if ( val < 0 )
        val = 0;
    nbNeg = val;
}

public void deplaceADroite() {
    x = x + 1;
    if ( x > 599 )
        x = 0;
}

public void deplaceAGauche() {
    x = x - 1;
    if ( x < 0 )
        x = 599;
}

public void deplaceEnBas() {
    y = y + 1;
    if ( y > 599 )
        y = 0;
}

public void deplaceEnHaut() {
    y = y - 1;
    if ( y < 0 )
        y = 599;
}
```

Proteine.java - Exercice 3.2 (config)

```
import java.awt.*;
```

```
class Proteine {

    private int nbPos = 0;
    private int nbNeg = 0;
    private int x,y;
    private char config;

    public void setPos(int val) {
        if ( val < 0 )
            val = 0;
        nbPos = val;
    }

    public void setNeg(int val) {
        if ( val < 0 )
            val = 0;
        nbNeg = val;
    }
}
```

```
public int getPos() {
    return nbPos;
}

public int getNeg() {
    return nbPos;
}

public int getCharge() {
    return nbPos - nbNeg;
}

public void setX(int abs) {
    x = abs;
}

public void setY(int ord) {
    y = ord;
}

public int getX() {
    return x;
}

public int getY() {
    return y;
}

public void deplaceADroite() {
    x = x + 10;
    if ( x > 599 )
        x = 0;
}

public void deplaceAGauche() {
    x = x - 10;
    if ( x < 0 )
        x = 599;
}

public void deplaceEnBas() {
    y = y + 10;
    if ( y > 599 )
        y = 0;
}

public void deplaceEnHaut() {
    y = y - 10;
    if ( y < 0 )
        y = 599;
}

public void setConfig(char c) {
    config = c;
}

public void changeConfig() {
    if (config=='P')
        config='N';
    else
        config='P';
}

public char getConfig() {
    return config;
}

public void afficheCharge() {
    System.out.println("Charge = "+(nbPos-nbNeg)+" a pH=7.");
}

public void affichePosition() {
    System.out.println("Coordonnees : "+x+" ; "+y);
}
```

```

public void afficheConfig() {
    if(config == 'P' )
        System.out.println("proteine phosphorylee");
    else
        System.out.println("proteine non phosphorylee");
}

public void dessine(Graphics g) {
    int iv = 0;
    if(config == 'P' )
        iv = 255;
    g.setColor (new Color(127-10*getCharge(),iv,127+10*getCharge()));
    g.fillOval (x-9, y-9, 19, 19);
}
}

```

Milieu.java - Exercice 3.2

```

import java.awt.*;

class Milieu {

    public static void delay (int ms) {
        long time = System.currentTimeMillis();
        while (System.currentTimeMillis() - time < ms);
    }

    public static void main(String args[]) {

        Frame fenetre = new Frame ("milieu");

        fenetre.setSize (600, 600);
        fenetre.setVisible (true);

        Graphics g = fenetre.getGraphics();

        Proteine prot1 = new Proteine();
        Proteine prot2 = new Proteine();

        delay (200);
        g.setColor (Color.white);
        g.fillRect (0,0,600,600);

        prot1.afficheCharge();

        prot1.setPos(4);
        prot1.setNeg(7);
        prot1.setX(375);
        prot1.setY(210);
        prot1.setConfig('N');
        prot1.afficheCharge();
        prot1.affichePosition();
        prot1.afficheConfig();
        prot1.dessine(g);
        prot2.setPos(3);
        prot2.setNeg(1);
        prot2.setX(32);
        prot2.setY(45);
        prot2.setConfig('P');
        System.out.println();
        prot2.afficheCharge();
        prot2.affichePosition();
        prot2.afficheConfig();
        prot2.dessine(g);

        prot1.deplaceADroite();
        prot1.deplaceADroite();
        prot1.deplaceADroite();
        prot1.deplaceADroite();
        prot2.deplaceAGauche();
        prot2.deplaceEnBas();
        prot2.changeConfig();
    }
}

```

```

delay (2000);
System.out.println();
prot1.affichePosition();
prot1.afficheConfig();
prot2.affichePosition();
prot2.afficheConfig();

g.setColor (Color.white);
g.fillRect (0,0,600,600);
prot1.dessine(g);
prot2.dessine(g);
}
}

```

Proteine.java - Exercice 3.3 (valeurs possibles)

// Méthodes modifiées

```

public void setX(int abs) {
    if ( ( abs > 0 ) && ( abs < 600 ) )
        x = abs;
    else {
        System.out.println("Valeur x="+abs+"impossible => x=300");
        x = 300;
    }
}

public void setY(int ord) {
    if ( ( ord > 0 ) && ( ord < 600 ) )
        y = ord;
    else {
        System.out.println("Valeur y="+ord+"impossible => y=300");
        y = 300;
    }
}

public void setConfig(char c) {
    if ( ( c == 'N' ) || ( c == 'P' ) )
        config = c;
    else {
        System.out.println("Configuration"+c+"impossible => N");
        config = 'N';
    }
}
}

```

TD 4 - Constructeurs et agrégation

Proteine.java - Demo 4.1 et Exercice 4.1 (constructeur)

```
import java.awt.*;
class Proteine {

    private int nbPos = 0;
    private int nbNeg = 0;
    private int x,y;
    private char config;

    public Proteine() {
        setX(300);
        setY(300);
        setConfig('N');
        setPos(0);
        setNeg(0);
    }

    // le reste de la classe ne change pas
}
```

Milieu.java - Exercice 4.1 (pour voir le résultat)

```
import java.awt.*;
class Milieu {

    ... // déclaration de la méthode delay()

    public static void main(String args[]) {

        ... // Création et initialisation de fenetre
        Graphics g = fenetre.getGraphics();

        Proteine prot1 = new Proteine(); // NB prot1.x et prot1.y valent 300, prot1.config='N'
        Proteine prot2 = new Proteine(); // NB prot2.x et prot2.y valent 300, prot2.config='N'

        // Remplissage de la fenêtre en blanc
        prot1.afficheCharge();

        prot1.setPos(4);          // On peut modifier ensuite
        prot1.setNeg(7);          // les valeurs par défaut
        // prot1.setX(375);       On peut garder la position
        // prot1.setY(210);       (300,300) si elle convient
        // prot1.setConfig('N');  inutile : c'est la configuration par défaut

        ... // affichages, dessin, etc.

        prot2.setPos(3);
        prot2.setNeg(1);
        prot2.setX(32);          // On peut changer ces variables si les
        prot2.setY(45);          // valeurs par défaut ne conviennent pas
        prot2.setConfig('P');    // On le garde !
        System.out.println();
        prot2.afficheCharge();
        prot2.affichePosition();
        prot2.afficheConfig();
        prot2.dessine(g);

        // etc. avec déplacements, changements de configuration, affichages...
    }
}
```

Proteine.java - Demo 4.2 (constructeur avec paramètres)

```
// Méthode à AJOUTER (garder l'autre constructeur)

public Proteine(int np, int ng) {
    setX(300);
    setY(300);
    setConfig('N');
    setPos(np);
    setNeg(ng);
}
```

Proteine.java - Exercice 4.2 (constructeur avec paramètres)

```
// Modifier le constructeur ainsi :
```

```
public Proteine(int np, int ng, int abs, int ord, char c) {
    setX(abs);
    setY(ord);
    setConfig(c);
    setPos(np);
    setNeg(ng);
}
```

Milieu.java - Exercice 4.2 (pour voir le résultat)

```
// Modifications :
```

```
Proteine prot1 = new Proteine(); // tous les parametres par default
Proteine prot2 = new Proteine(3,1,32,45,'P'); // tout est regle
```

```
// Remplissage de la fenetre en blanc
```

```
prot1.afficheCharge();
```

```
prot1.setPos(4);
prot1.setNeg(7);
prot1.afficheCharge();
prot1.affichePosition();
prot1.afficheConfig();
prot1.dessine(g);
```

```
// prot2.setPos(3);      inutile : le constructeur l'a fait
// prot2.setNeg(1);      inutile : le constructeur l'a fait
// prot2.setX(32);       inutile : le constructeur l'a fait
// prot2.setY(45);       inutile : le constructeur l'a fait
// prot2.setConfig('P'); inutile : le constructeur l'a fait
System.out.println();
prot2.afficheCharge();
```

Catal.java - Demo 4.3

```
import java.awt.*;
```

```
class Catal {
```

```
    private int x,y,taille;
    private boolean oqpN, oqpP;
```

```
    public Catal() {
        setX(300);
        setY(300);
        setTaille(50);
    }
```

```
    public Catal(int abs, int ord, int t) {
        setX(abs);
        setY(ord);
        setTaille(t);
    }
```

```
    public void setX(int abs) {
        if ( ( abs > 0 ) && ( abs < 600 ) )
            x = abs;
        else {
            System.out.println("Valeur x="+abs+"impossible => x=300");
            x = 300;
        }
    }
```

```
    public void setY(int ord) {
        if ( ( ord > 0 ) && ( ord < 600 ) )
            y = ord;
        else {
            System.out.println("Valeur y="+ord+"impossible => y=300");
            y = 300;
        }
    }
```

```

    }
}

public void setTaille(int val) {
    taille = val;
}

public int getX() {
    return x;
}

public int getY() {
    return y;
}

public void affichePosition() {
    System.out.println("Coordonnees : "+x+" ; "+y);
}

public int getXsite (char conf) {
    if (conf == 'P')
        return x - (taille/2);
    else
        return x;
}

public int getYsite (char conf) {
    if (conf == 'P')
        return y;
    else
        return y - (taille/2);
}

public void fixe (char conf) {
    if (conf == 'P') {
        oqpN = false;
        oqpP = true;
    }
    else {
        oqpN = true;
        oqpP = false;
    }
}

public void dessine(Graphics g) {
    g.setColor (Color.pink);
    g.fillRect (x-taille/2, y-taille/2, taille, taille);
}
}

```

Complexe.java - Demo 4.3

```

import java.awt.*;

class Complexe {

    private Catal cat;
    private Proteine sub;

    public Complexe (int posX, int posY, int tailleCat, int nbp, int nbn, char config) {
        cat = new Catal (posX, posY, tailleCat);
        cat.fixe (config);
        sub = new Proteine (nbp,nbn,cat.getXsite(config), cat.getYsite (config), config);
    }

    public void dessine (Graphics g) {
        cat.dessine (g);
        sub.dessine (g);
    }
}

```

Milieu.java - Demo 4.3

```
import java.awt.*;

class Milieu {

    public static void delay (int ms) {
        long time = System.currentTimeMillis();
        while (System.currentTimeMillis() - time < ms);
    }

    public static void main(String args[]) {

        Frame fenetre = new Frame ("milieu");

        fenetre.setSize (600,600);
        fenetre.setVisible (true);

        Graphics g = fenetre.getGraphics();

        Complexe cplx = new Complexe(300,300,50,3,1,'P');

        delay (200);
        g.setColor (Color.white);
        g.fillRect (0,0,600,600);

        cplx.dessine(g);

    }
}
```

Complexe.java - Exercice 4.3 (méthode transloc())

```
public void transloc () {
    sub.changeConfig();
    char newConf = sub.getConfig();
    cat.fixe (newConf);
    sub.setX (cat.getXsite (newConf));
    sub.setY (cat.getYsite (newConf));
}
```

Milieu.java - Exercice 4.3 (pour voir le résultat)

```
// Modifications :

    delay (200);
    g.setColor (Color.white);
    g.fillRect (0,0,600,600);
    cplx.dessine(g);

    cplx.transloc();

    delay (500);
    g.setColor (Color.white);
    g.fillRect (0,0,600,600);
    cplx.dessine(g);
```


TD 5 - Boucles, tableaux et chaînes de caractères

Proteine.java - Demo 5.1 et Exercice 5.1 (setX() avec boucle)

```
// Modifier la méthode setX() comme ceci :
```

```
public void setX(int abs) {
    while ( abs < 0 )
        abs = abs + 600;
    while ( abs > 599 )
        abs = abs - 600;
    x = abs;
}
```

Proteine.java - Exercice 5.2 (déplacement aléatoire)

```
// Ajouter à l'en-tête et aux déclarations
```

```
import java.util.*;
...
private Random alea = new Random();
...
```

```
// Ajouter la méthode :
```

```
public void bougeAlea(int nb) {
    setX(x+alea.nextInt(2*nb+1)-nb);
    setY(y+alea.nextInt(2*nb+1)-nb);
}
```

Reaction.java - Exercice 5.2

```
import java.awt.*;
import java.util.*; // Pour Random
```

```
class Reaction {
```

```
// Déclarer la méthode delay()
```

```
public static void main(String args[]) {
```

```
    Frame fenetre = new Frame ("reaction");
    fenetre.setSize (600,600);
    fenetre.setVisible (true);
    Graphics g = fenetre.getGraphics();
```

```
    Random alea = new Random();
    int rx = alea.nextInt (600);
    int ry = alea.nextInt (600);
```

```
    Proteine prot = new Proteine(5,2,rx,ry,'P');
```

```
    delay (200);
    g.setColor (Color.white);
    g.fillRect (0,0,600,600);
    prot.dessine(g);
```

```
    double dist = Math.sqrt((prot.getX()-300)*(prot.getX()-300)+(prot.getY()-300)*(prot.getY()-300));
    System.out.println(prot.getX()+" "+prot.getY()+"->" +dist);
```

```
    while(dist>40) {
        prot.bougeAlea(5);
        dist = Math.sqrt((prot.getX()-300)*(prot.getX()-300)+(prot.getY()-300)*(prot.getY()-300));
        System.out.println(prot.getX()+" "+prot.getY()+"->" +dist);
        g.setColor (Color.white);
        g.fillRect (0,0,600,600);
        prot.dessine(g);
        delay (10);
    }
```

```
    }
}
```

Milieu.java - Demo 5.2 (10 translocations)

```
// Répéter l'instruction de translocation comme suit :
for(int i=0;i<10;i++) {
    delay (500);
    cplx.transloc();
    g.setColor (Color.white);
    g.fillRect (0,0,600,600);
    cplx.dessine(g);
}
```

Reaction.java - Exercice 5.3 (100 pas)

```
// Modifier ainsi :
```

```
for(int i=1;i<100;i++) {
    prot.bougeAlea(5);
    dist = Math.sqrt((prot.getX()-300)*(prot.getX()-300)+(prot.getY()-300)*(prot.getY()-300));
    System.out.println(i+" "+prot.getX()+" "+prot.getY()+"->"+"dist);
    g.setColor (Color.white);
    g.fillRect (0,0,600,600);
    prot.dessine(g);
    delay (10);
}
```

Reaction.java - Demo 5.3 et Exercice 5.4 (tableaux)

```
// Modifier ainsi :
```

```
int npas = 100;
int xl[] = new int[npas+1];
int yl[] = new int[npas+1];
double distl[] = new double[npas+1];
Random alea = new Random();
int rx = alea.nextInt (600);
int ry = alea.nextInt (600);

Proteine prot = new Proteine(5,2,rx,ry,'P');

delay (200);
g.setColor (Color.white);
g.fillRect (0,0,600,600);
prot.dessine(g);

xl[0]=prot.getX();
yl[0]=prot.getY();
distl[0] = Math.sqrt((prot.getX()-300)*(prot.getX()-300)+(prot.getY()-300)*(prot.getY()-300));

for(int i=1;i<distl.length;i++) {
    prot.bougeAlea(5);
    xl[i]=prot.getX();
    yl[i]=prot.getY();
    distl[i] = Math.sqrt((prot.getX()-300)*(prot.getX()-300)+(prot.getY()-300)*(prot.getY()-300));
    g.setColor (Color.white);
    g.fillRect (0,0,600,600);
    prot.dessine(g);
    delay (10);
}

for(int i=0;i<distl.length;i++) {
    System.out.println(i+" : "+xl[i]+" "+yl[i]+"->"+"distl[i]);
}
}
```

Milieu.java - Demo 5.4 (plusieurs complexes)

```
import java.awt.*;

class Milieu {

    public static void delay (int ms) {
        long time = System.currentTimeMillis();
        while (System.currentTimeMillis() - time < ms);
    }
}
```

```

public static void main(String args[]) {

    Frame fenetre = new Frame ("milieu");

    fenetre.setSize (600,600);
    fenetre.setVisible (true);

    Graphics g = fenetre.getGraphics();

    Complexe cplx[] = new Complexe[4];

    for(int c=0;c<4;c++)
        cplx[c] = new Complexe(60*c+60,60*c+60,50,3,1,'P');

    delay (200);
    g.setColor (Color.white);
    g.fillRect (0,0,600,600);
    for(int c=0;c<4;c++)
        cplx[c].dessine(g);

    for(int i=0;i<50;i++) {
        delay (500);
        int c=i%4;
        cplx[c].transloc();
        cplx[c].dessine(g);
    }

}
}
}

```

AcAm.java - Demo 5.5

```

class AcAm {

    private char symbole;
    private int chph7;

    public AcAm(char s) {
        symbole = s;
        chph7 = 0;
        if ( (symbole=='D') || (symbole=='E') )
            chph7 = -1;
        if ( (symbole=='H') || (symbole=='K') || (symbole=='R') )
            chph7 = 1;
    }

    public char getSymb() {
        return symbole;
    }

    public int getChpH7() {
        return chph7;
    }

}

```

Proteine.java - Demo 5.5 (séquence)

```

import java.awt.*;
import java.util.*;
class Proteine {

    private AcAm[] sequence;
    private int x,y;
    private char config;
    private Random alea = new Random();

    public Proteine(String seq, int abs, int ord, char c) {
        sequence = new AcAm[seq.length()];
        for(int i=0;i<seq.length();i++)
            sequence[i] = new AcAm(seq.charAt(i));
        setX(abs);
        setY(ord);
    }
}

```

```

    setConfig(c);
}

public int getPos() {
    int n = 0;
    for (int a=0;a<sequence.length;a++)
        if(sequence[a].getChpH7()>0)
            n=n+1;
    return n;
}

public int getNeg() {
    int n = 0;
    for (int a=0;a<sequence.length;a++)
        if(sequence[a].getChpH7()<0)
            n=n+1;
    return n;
}

// Le reste ne change pas
}

```

Reaction.java - Demo 5.5 (séquence)

```

import java.awt.*;
import java.util.*;

class Reaction {

// Déclarer la méthode delay()

    public static void main(String args[]) {

        Frame fenetre = new Frame ("reaction");

        fenetre.setSize (600,600);
        fenetre.setVisible (true);
        Graphics g = fenetre.getGraphics();

        int npas = 100;
        int xl[] = new int[npas+1];
        int yl[] = new int[npas+1];
        double distl[] = new double[npas+1];
        Random alea = new Random();
        int rx = alea.nextInt (600);
        int ry = alea.nextInt (600);

        Proteine prot = new Proteine("MAVDLPKSWPALVEQKRDLA",rx,ry,'N');

// La suite ne change pas

    }

}

```

TD 6 - Algorithmique

Proteine.java - Demo 6.1

```
// Ajouter à Proteine les méthodes suivantes :
public int getLen() {
    return sequence.length;
}

public char aa(int p) {
    return sequence[p].getSymb();
}
```

Complexe.java - Demo 6.1

```
//Modifier ainsi le constructeur :
public Complexe (int posX, int posY, int tailleCat, String sq, char config) {
    cat = new Catal (posX, posY, tailleCat);
    cat.fixe (config);
    sub = new Proteine (sq,cat.getXsite(config), cat.getYsite (config), config);
}

// Ajouter les méthodes :
public int getProLen() {
    return sub.getLen();
}

public char aaPro(int p) {
    return sub.aa(p);
}
```

Milieu.java - Demo 6.1

```
import java.awt.*;

class Milieu {

// Déclarer la méthode delay()

public static void main(String args[]) {

    Frame fenetre = new Frame ("milieu");

    fenetre.setSize (600,600);
    fenetre.setVisible (true);
    Graphics g = fenetre.getGraphics();

    Complexe cplx = new Complexe(300,300,50,"DEQSAVACIGGLRKTINVALLDLKVGDYVILHVGFALQKLDEAEAQRT",'N');
    String motif = new String("ILHVG");
    int lp = cplx.getProLen();
    int lm = motif.length();
    if(lm>lp)
        System.out.println("Motif plus long que la proteine");
    else {
        delay (500);
        cplx.dessine(g);
        boolean trouve;
        for (int p=0; p<=lp-lm;p++) {
            trouve = true;
            for (int i=0; i<=lm-1;i++)
                trouve = trouve && (cplx.aaPro(p+i)==motif.charAt(i));
            if(trouve) {
                System.out.println("motif en "+p);
                for(int t=0;t<10;t++) {
                    delay (500);
                    cplx.transloc();
                    g.setColor (Color.white);
                    g.fillRect (0,0,600,600);
                    cplx.dessine(g);
                }
            }
        }
    }
}
}
```

Proteine.java - Exercice 6.1

```
// Ajouter la méthode :
public boolean motifPresent(String motif) {
    int lp = sequence.length;
    int lm = motif.length();
    boolean present = false;
    if(lm>lp)
        System.out.println("Motif plus long que la proteine");
    else {
        for (int p=0; p<=lp-lm;p++) {
            boolean trouve = true;
            int i=0;
            while( (i<=lm-1) && trouve ) {
                trouve = trouve && (aa(p+i)==motif.charAt(i));
                i++;
            }
            if(trouve) {
                System.out.println("Motif "+motif+" en "+p);
                present = true;
            }
        }
    }
    return present;
}
```

Complexe.java - Exercice 6.1

```
// Ajouter la méthode :
public boolean motInSub(String mot) {
    return sub.motifPresent(mot);
}
```

Milieu.java - Exercice 6.1

```
// Modifier ainsi :
if( cplx.motInSub(motif) )
    for(int t=0;t<10;t++) {
        delay (500);
        cplx.transloc();
        g.setColor (Color.white);
        g.fillRect (0,0,600,600);
        cplx.dessine(g);
    }
```

Catal.java - Exercice 6.2

```
// Ajouter la méthode :
public int getTaille() {
    return taille;
}
```

Tri.java - Exercice 6.2

```
import java.awt.*;
import java.util.*;
class Tri {

// Déclarer la méthode delay()

private static int posmin(Catal[] cl, int d) {
    int pm = d;
    for(int c=d+1;c<cl.length;c++) {
        if (cl[c].getTaille() < cl[pm].getTaille() )
            pm = c;
    }
    return pm;
}

private static void echange(Catal[] cl, int a, int b) {
    Catal t = cl[a];
    cl[a] = cl[b];
    cl[b] = t;
}
```

```
public static void main(String args[]) {

    Frame fenetre = new Frame ("tri");
    fenetre.setSize (600,600);
    fenetre.setVisible (true);
    Graphics g = fenetre.getGraphics();

    Random alea = new Random();
    int ncat = 12;
    int c, rt;
    Catal cats[] = new Catal[ncat];
    for ( c = 0; c < ncat ; c++) {
        rt = alea.nextInt(16) + 5;
        cats[c] = new Catal(c*600/ncat+300/ncat,40,rt);
    }

    delay (200);
    g.setColor (Color.white);
    g.fillRect (0,0,600,600);
    for ( c = 0; c < ncat ; c++)
        cats[c].dessine(g);

    int pp;
    for(int e=0;e<ncat-1;e++) {
        pp = posmin(cats,e);
        echange(cats,e,pp);
        delay (20);
        for ( c = 0; c < ncat ; c++) {
            cats[c].setX(c*600/ncat+300/ncat);
            cats[c].setY(cats[c].getY()+25);
            cats[c].dessine(g);
        }
    }
}
```


TD 7 - Interactions entre objets

Proteine.java - Exercice 7.1 (Observable)

```

import java.awt.*;
import java.util.*;

class Proteine extends Observable {

    private AcAm[] sequence;
    private int x,y;
    private char config;
    private Graphics g;
    private boolean libre;
    private Random alea = new Random();

    public Proteine(String seq, int abs, int ord, char c, Graphics gr) {
        sequence = new AcAm[seq.length()];
        for(int i=0;i<seq.length();i++)
            sequence[i] = new AcAm(seq.charAt(i));
        setX(abs);
        setY(ord);
        setConfig(c);
        g = gr;
        libre = true;
    }

    public static void delay (int ms) {
        long time = System.currentTimeMillis();
        while (System.currentTimeMillis() - time < ms);
    }

    public void setLiberte (boolean lib) {
        libre = lib;
    }

    public boolean getLiberte () {
        return libre;
    }

    public void efface() {
        g.setColor (Color.white);
        g.fillOval (x-9, y-9, 19, 19);
    }

    public void dessine() {
        int iv = 0;
        if(config == 'P' )
            iv = 255;
        g.setColor (new Color(127-10*getCharge(),iv,127+10*getCharge()));
        g.fillOval (x-9, y-9, 19, 19);
    }

    public void vivre () {
        while (true) {
            efface();
            if (libre) {
                bougeAlea(10);
            }
            dessine();
            setChanged();
            notifyObservers(); // le message est envoye a chaque tour,
            delay (100);      // que la proteine soit libre ou non
        }
    }

    // les autres méthodes ne changent pas
}

```

Catal.java - Exercice 7.1 (Observer)

```

import java.awt.*;
import java.util.*;

class Catal implements Observer {

    private int x,y,taille;
    private boolean oqpN, oqpP;
    private Graphics g;
    private Random r;

    public Catal() {
        setX(300);
        setY(300);
        setTaille(50);
        g = null;
        r = new Random();
    }

    public Catal(int abs, int ord, int t, Graphics gr) {
        setX(abs);
        setY(ord);
        setTaille(t);
        g = gr;
        r = new Random();
    }

    public void fixeProt (Proteine prot) {
        fixe(prot.getConfig());
        prot.setLiberte (false);
    }

    public void libereProt (Proteine prot) {
        oqpN = false;
        oqpP = false;
        prot.setLiberte (true);
    }

    public void dessine() {
        g.setColor (Color.pink);
        g.fillRect (x-taille/2, y-taille/2, taille, taille);
    }

    public void update (Observable obs, Object arg) {

        if( ((Proteine)obs).getLiberte() ) {
            int xMol = ((Proteine)obs).getX();
            int yMol = ((Proteine)obs).getY();
            if ( (xMol>x-taille/2) && (xMol<x+taille/2) // si la proteine est dans
                && (yMol>y-taille/2) && (yMol<y+taille/2)) { // la surface du catalyseur
                fixeProt((Proteine)obs);
            }
        }
        else {
            int d = r.nextInt(100); // liberation aleatoire 1 fois sur 20 tours
            if (d<5) {
                ((Proteine)obs).efface();
                ((Proteine)obs).setX (x -100);
                ((Proteine)obs).setY (y -100);
                libereProt((Proteine)obs);
            }
        }

        dessine();
    }

    // les autres méthodes ne changent pas
}

```

Observation.java - Exercice 7.1

```

import java.awt.*;

class Observation {

    public static void delay (int ms) {
        long time = System.currentTimeMillis();
        while (System.currentTimeMillis() - time < ms);
    }

    public static void main (String arg[]) {
        Proteine prot;
        Catal enz;

        Frame fenetre = new Frame ("Observation");
        fenetre.setSize (600,600);
        fenetre.setVisible (true);

        Graphics g = fenetre.getGraphics();
        delay(500);
        g.setColor (Color.white);
        g.fillRect (0,0,600,600);

        prot = new Proteine ("DEQSAVACIGGLRKTINVALLDLKVGDYVILHVGFGALQKLDEAEAQRT", 250, 250, 'N',g);
        enz = new Catal (300, 300, 80,g);

        prot.addObserver (enz);

        enz.dessine();
        prot.vivre();
    }
}

```

Proteine.java - Demo 7.1 (ActionListener)

```

import java.awt.*;
import java.util.*;
import java.awt.event.*;

class Proteine extends Observable implements ActionListener {

    // Ajouter l'attribut :
    private int attente = 100;

    // Modifier dans la méthode :
    public void vivre () {
        ...
        delay (attente);
        ...
    }

    // Ajouter :
    public void actionPerformed (ActionEvent e) {
        if (e.getActionCommand().equals("rapide"))
            attente = 50;
        if (e.getActionCommand().equals("lent"))
            attente = 500;
    }
}

```

Controle.java - Demo 7.1 (boutons)

```

import java.awt.*;
import java.util.*;

class Controle {

    public static void delay(int ms) {
        long time = System.currentTimeMillis();
        while(System.currentTimeMillis() - time < ms)
            ;
    }
}

```

```
public static void main (String arg[]) {

    Proteine prot;

    Frame fenetre = new Frame ("Controle");
    fenetre.setSize (600, 800);
    fenetre.setVisible (true);
    Graphics g = fenetre.getGraphics();
    delay(2000);
    g.setColor (Color.white);
    g.fillRect (0,0,600,600);
    prot = new Proteine ("DEQSAVACIGGLRKTINVALDDLLKVGDYVILHVG FALQKLDEAEAQRT", 300, 300, 'N',g);

    Button rapide = new Button ("rapide");
    Button lent = new Button ("lent");

    fenetre.add (rapide);
    fenetre.add (lent);

    // specification graphique des boutons
    rapide.setSize (100, 30);
    rapide.setLocation (100, 700);
    lent.setSize (100, 30);
    lent.setLocation (300, 700);

    // associer un nom de commande specifique
    rapide.setActionCommand("rapide");
    lent.setActionCommand("lent");

    // enregistrer l'ecouteur
    rapide.addActionListener (prot);
    lent.addActionListener (prot);

    rapide.setVisible (true);
    lent.setVisible (true);

    prot.vivre();

}

}
```

TD 8 - Héritage I

Molecule.java - Demo 8.1

```
import java.util.*;

class Molecule extends Observable {
    protected int x,y;
    public void setX(int abs) {
        while ( abs < 0 )
            abs = abs + 600;
        while ( abs > 599 )
            abs = abs - 600;
        x = abs;
    }
    public void setY(int ord) {
        while ( ord < 0 )
            ord = ord + 600;
        while ( ord > 599 )
            ord = ord - 600;
        y = ord;
    }
    public int getX() {
        return x;
    }
    public int getY() {
        return y;
    }
}
```

Proteine.java - Demo 8.1 (dérive de Molecule)

```
class Proteine extends Molecule implements ActionListener {
    // supprimer les membres x et y, et les méthodes écrites dans Molecule
}
```

Catal.java - Demo 8.1 (dérive de Molecule)

```
class Catal extends Molecule implements Observer {
    // supprimer les membres x et y, et les méthodes écrites dans Molecule
}
```

MolMob.java - Exercice 8.1

```
import java.util.*;
import java.awt.*;

class MolMob extends Molecule {
    protected Graphics g;
    protected boolean libre;
    protected int attente = 100;
    protected Random alea = new Random();
    public static void delay (int ms) {
        long time = System.currentTimeMillis();
        while (System.currentTimeMillis() - time < ms);
    }
    public void setLiberte (boolean lib) {
        libre = lib;
    }
    public boolean getLiberte () {
        return libre;
    }
    public void deplaceADroite() {
        x = x + 10;
        if ( x > 399 )
            x = 0;
    }
    public void deplaceAGauche() {
        x = x - 10;
        if ( x < 0 )
            x = 399;
    }
    public void deplaceEnBas() {
        y = y + 10;
    }
}
```

```

    if ( y > 399 )
        y = 0;
}
public void deplaceEnHaut() {
    y = y - 10;
    if ( y < 0 )
        y = 399;
}
public void bougeAlea(int nb) {
    setX(x+alea.nextInt(2*nb+1)-nb);
    setY(y+alea.nextInt(2*nb+1)-nb);
}
public void affichePosition() {
    System.out.println("Coordonnees : "+x+" ; "+y);
}
}

```

Phosphate.java - Exercice 8.1

```

class Phosphate extends MolMob {
    public Phosphate(int abs, int ord, Graphics gr) {
        setX(abs);
        setY(ord);
        g = gr;
        libre = true;
    }
    public void efface() {
        g.setColor (Color.white);
        g.fillOval (x-3, y-3, 7, 7);
    }
    public void dessine() {
        g.setColor (new Color(0,127,0));
        g.fillOval (x-3, y-3, 7, 7);
    }
    public void vivre () {
        efface();
        if (libre)
            bougeAlea(30);
        dessine();
        delay (attente);
    }
}

```

Proteine.java - Exercice 8.1

```

class Proteine extends MolMob implements ActionListener {
    // supprimer les membres et méthodes repris dans MolMob
    public void vivre () {
        efface();
        if (libre)
            bougeAlea(10);
        dessine();
        setChanged();
        notifyObservers();
        delay (attente);
    }
}

```

Observation.java - Exercice 8.1 (Proteine et Phosphate)

```

import java.awt.*;

class Observation {
    public static void delay (int ms) {
        long time = System.currentTimeMillis();
        while (System.currentTimeMillis() - time < ms);
    }

    public static void main (String arg[]) {
        Phosphate phos;
        Proteine prot;
        Catal enz;

        Frame fenetre = new Frame ("Observation");
    }
}

```

```

fenetre.setSize (600,600);
fenetre.setVisible (true);

Graphics g = fenetre.getGraphics();
delay(500);
g.setColor (Color.white);
g.fillRect (0,0,600,600);

phos = new Phosphate (350,350,g);
prot = new Proteine ("DEQSAVACIGGLRKTINVALDDLDKVGDYVILHVGFFALQKLDEAEAQRT", 250, 250, 'N',g);
enz = new Catal (300, 300, 80,g);

prot.addObserver (enz);

while(true) {
    enz.dessine();
    phos.vivre();
    prot.vivre();
}
}
}

```

MolMob.java - Demo 8.2 (constructeur MolMob avec paramètres)

```

// Ajouter le constructeur :
protected MolMob(int abs, int ord, Graphics gr) {
    setX(abs);
    setY(ord);
    g = gr;
    libre = true;
}

```

Proteine.java - Demo 8.2 (constructeur MolMob avec paramètres)

```

// Modifier le constructeur :
public Proteine(String seq, int abs, int ord, char c, Graphics gr) {
    super(abs,ord,gr);
    sequence = new AcAm[seq.length()];
    for(int i=0;i<seq.length();i++)
        sequence[i] = new AcAm(seq.charAt(i));
    setConfig(c);
}

```

Phosphate.java - Demo 8.2 (constructeur MolMob avec paramètres)

```

// Modifier le constructeur :
public Phosphate(int abs, int ord, Graphics gr) {
    super(abs,ord,gr);
}

```

Molecule.java - Exercice 8.2

```

// Ajouter :
protected Graphics g;

protected Molecule(int abs, int ord, Graphics gr) {
    setX(abs);
    setY(ord);
    g = gr;
}

```

MolMob.java - Exercice 8.2

```

// Enlever le membre Graphics gr
// Modifier ainsi le constructeur :
protected MolMob(int abs, int ord, Graphics gr) {
    super(abs,ord,gr);
    libre = true;
}

```

Catal.java - Exercice 8.2

```

// Enlever le membre Graphics gr
// Modifier ainsi les constructeurs :
public Catal() {
    super(300,300,null);
    setTaille(50);
}

```

```

    r = new Random();
}
public Catal(int abs, int ord, int t, Graphics gr) {
    super(abs,ord,gr);
    setTaille(t);
    r = new Random();
}

```

Molecule.java - Exercice 8.3

```

// Constructeur avec positionnement choisi
protected Molecule(int x, int y, Graphics gr) {
    setX(x);
    setY(y);
    g = gr;
}
// Constructeur avec positionnement aléatoire
protected Molecule(Graphics gr) {
    setX(alea.nextInt(600));
    setY(alea.nextInt(600));
    g = gr;
}

```

MolMob.java - Exercice 8.3

```

// Supprimer protected Random alea = new Random();
// Ajouter :
protected int speed;
protected MolMob(Graphics gr) {
    super(gr);
    libre = true;
}
// Modifier ainsi :
public void bougeAlea() {
    setX(x+alea.nextInt(2*speed+1)-speed);
    setY(y+alea.nextInt(2*speed+1)-speed);
}

```

Proteine.java - Exercice 8.3

```

// Nouveau constructeur :
public Proteine(String seq, char c, Graphics gr) {
    super(gr);
    sequence = new AcAm[seq.length()];
    for(int i=0;i<seq.length();i++)
        sequence[i] = new AcAm(seq.charAt(i));
    setConfig(c);
    speed = 10;
}
// Constructeur modifié :
public Proteine(String seq, int abs, int ord, char c, Graphics gr) {
    super(abs,ord,gr);
    sequence = new AcAm[seq.length()];
    for(int i=0;i<seq.length();i++)
        sequence[i] = new AcAm(seq.charAt(i));
    setConfig(c);
    speed = 10;
}
// Dans vivre(), modifier :
    bougeAlea();

```

Phosphate.java - Exercice 8.3

```

// Constructeurs modifiés :
public Phosphate(Graphics gr) {
    super(gr);
    speed = 30;
}
public Phosphate(int abs, int ord, Graphics gr) {
    super(abs,ord,gr);
    speed = 30;
}
// Dans vivre(), modifier :
    bougeAlea();

```


TD 9 - Héritage II

Complexe.java - Demo 9.1 (complexe Observer)

```

import java.awt.*;
import java.util.*;

class Complexe implements Observer {

    private Catal cat;
    private Proteine sub;
    private Random r;

    public Complexe (int posX, int posY, int tailleCat, String sq, char config, Graphics gr) {
        cat = new Catal (posX, posY, tailleCat, gr);
        cat.fixe (config);
        sub = new Proteine (sq, cat.getXsite (config), cat.getYsite (config), config, gr);
        r = new Random();
    }
    // constructeur Complexe (int posX, int posY, int tailleCat, Graphics gr) inchangé

    public void transloc () {
        if(sub!=null) {
            sub.changeConfig();
            char newConf = sub.getConfig();
            cat.fixe (newConf);
            sub.setX (cat.getXsite (newConf));
            sub.setY (cat.getYsite (newConf));
        }
    }

    public void dessine () {
        cat.dessine ();
        if(sub!=null)
            sub.dessine ();
    }

    // méthodes getProLen(), aaPro(int p) inchangées

    public boolean motInSub(String mot) {
        if(sub!=null)
            return sub.motifPresent(mot);
        else
            return false;
    }

    public void update (Observable obs, Object arg) {
        int x = cat.getX();
        int y = cat.getY();
        int t = cat.getTaille();
        if( ((Proteine)obs)!=sub && sub==null ) {
            int xMol = ((Proteine)obs).getX();
            int yMol = ((Proteine)obs).getY();
            if ( (xMol>x-t/2) && (xMol<x+t/2) // si la proteine est dans
                && (yMol>y-t/2) && (yMol<y+t/2)) { // la surface du catalyseur
                cat.fixeProt((Proteine)obs);
                sub=(Proteine)obs;
            }
        }
        else {
            int d = r.nextInt(100); // liberation aleatoire 1 fois sur 20 tours
            if (d<5) {
                ((Proteine)obs).efface();
                ((Proteine)obs).setX (x-100);
                ((Proteine)obs).setY (y-100);
                cat.libereProt((Proteine)obs);
                sub = null;
            }
            else
                transloc();
        }
    }
}

```

Catal.java - Demo 9.1 (complexe Observer)

```
// Supprimer de l'en-tête : implements Observer
// Supprimer la méthode update()
```

Observation.java - Demo 9.1 (complexe Observer)

```
// Remplacer la déclaration Catal enz; par :
    Complexe cplx;
// Remplacer l'initialisation enz = new Catal (300, 300, 80,g); par :
    cplx = new Complexe(300,300,80,g);
// Remplacer prot.addObserver (enz); par :
    prot.addObserver (cplx);
// Remplacer la boucle infinie par :
    while(true) {
        g.setColor (Color.white);
        g.fillRect (0,0,600,600);
        cplx.dessine();
        phos.vivre();
        prot.vivre();
    }
```

MolMob.java - Demo 9.1 (tableau de MolMob)

```
// Ajouter la méthode :
    public void vivre() {
    }
```

Proteine.java et Phosphate.java - Demo 9.1 (tableau de MolMob)

```
// Enlever :
    delay (attente);
```

Observation.java - Demo 9.1 (tableau de MolMob)

```
public static void main (String arg[]) {
    MolMob molb[];
    Complexe cplx;

    Frame fenetre = new Frame ("Observation");
    fenetre.setSize (600,600);
    fenetre.setVisible (true);

    Graphics g = fenetre.getGraphics();
    delay(500);
    g.setColor (Color.white);
    g.fillRect (0,0,600,600);

    cplx = new Complexe(300,300,80,g);
    molb = new MolMob[100];
    for(int i=0;i<100;i+=2) {
        molb[i] = new Phosphate (g);
        molb[i].addObserver (cplx);
        molb[i+1] = new Proteine ("DEQSAVACIGGLRKTINVALLDLKVGDYVILHVGFALQKLDEAEAQRT", 'N',g);
        molb[i+1].addObserver (cplx);
    }

    while(true) {
        g.setColor (Color.white);
        g.fillRect (0,0,600,600);
        cplx.dessine();
        for(int i=0;i<100;i++)
            molb[i].vivre();
        delay(150);
    }
}
```

Catal.java - Exercice 9.1 (réaction)

```
public void libereProt (Proteine prot) {
    oqpN = false;
    oqpP = false;
    prot.setX (x-100);
    prot.setY (y-100);
    prot.setLiberte (true);
}
```

Phosphate.java - Exercice 9.1 (réaction)

```

public void vivre () {
    if (libre) {
        bougeAlea();
        dessine();
    }
    setChanged();
    notifyObservers();
}

```

Proteine.java - Exercice 9.1(réaction)

```

class Proteine extends MolMob implements ActionListener {

    private AcAm[] sequence;
    private Phosphate pho;

    public Proteine(String seq, Phosphate p, Graphics gr) {
        super(gr);
        sequence = new AcAm[seq.length()];
        for(int i=0;i<seq.length();i++)
            sequence[i] = new AcAm(seq.charAt(i));
        setConfig(p);
        speed = 10;
    }

    public Proteine(String seq, int abs, int ord, Phosphate p, Graphics gr) {
        super(abs,ord,gr);
        sequence = new AcAm[seq.length()];
        for(int i=0;i<seq.length();i++)
            sequence[i] = new AcAm(seq.charAt(i));
        setConfig(p);
        speed = 10;
    }

    // Les méthodes getPos(), getNeg(), getCharge(), afficheCharge(), efface(),
    // getLen(), aa(), motifPresent(), actionPerformed() ne changent pas.
    // transloc() disparaît

    public void setConfig(Phosphate p) {
        if(pho==null) {
            if(p!=null) {
                pho=p;
                pho.setLiberte(false);
            }
        }
        else {
            pho.setLiberte(true);
            pho=null;
        }
    }

    public char getConfig() {
        if(pho==null)
            return 'N';
        else
            return 'P';
    }

    public void afficheConfig() {
        if(pho!=null)
            System.out.println("proteine phosphorylee");
        else
            System.out.println("proteine non phosphorylee");
    }

    public void vivre () {
        if (libre)
            bougeAlea();
        dessine();
        setChanged();
        notifyObservers();
    }
}

```

Complexe.java - Exercice 9.1 (réaction)

```
// Modifier :
public Complexe (int posX, int posY, int tailleCat, String sq, char config, Graphics gr) {
    cat = new Catal (posX, posY, tailleCat, gr);
    cat.fixe (config);
    sub = new Proteine (sq,cat.getXsite(config), cat.getYsite (config), null, gr);
    r = new Random();
}

public Complexe (int posX, int posY, int tailleCat, Graphics gr) {
    cat = new Catal (posX, posY, tailleCat, gr);
    sub = null;
    r = new Random();
}

public void update (Observable obs, Object arg) {
    int x = cat.getX();
    int y = cat.getY();
    int t = cat.getTaille();
    int xMol = ((MolMob)obs).getX();
    int yMol = ((MolMob)obs).getY();
    if ( (xMol>x-t/2) && (xMol<x+t/2) // si la molecule est dans
        &&(yMol>y-t/2) && (yMol<y+t/2)) { // la surface du catalyseur
        if(obs instanceof Proteine) {
            if (obs!=sub) {
                if (sub==null) {
                    if (((Proteine)obs).getConfig()=='N') {
                        cat.fixeProt((Proteine)obs);
                        sub=(Proteine)obs;
                    }
                    else {
                        ((Proteine)obs).setConfig(null);
                        cat.libereProt((Proteine)obs);
                    }
                }
            }
        }
        else {
            int d = r.nextInt(100); // liberation aleatoire 1 fois / 20 tours
            if (d<5) {
                cat.libereProt((Proteine)obs);
                sub = null;
            }
        }
    }
    else {
        if ( (sub!=null) && (sub.getConfig()=='N') ) {
            sub.setConfig((Phosphate)obs);
            cat.libereProt(sub);
            sub = null;
        }
    }
}
}
```

Observation.java - Exercice 9.1(réaction)

```
// Modifier :
molb[i+1] = new Proteine ("DEQSAVACIGGLRKTINVALDDLKVGDYVILHVGFALQKLDEAEAQRT", null,g);
```

MolMob.java - Exercice 9.2 (classes abstraites)

```
// Modifier :
abstract class MolMob extends Molecule {
    ...
    public abstract void vivre();
}
```

Molecule.java - Exercice 9.2 (classes abstraites)

```
// Modifier :
abstract class Molecule extends Observable {
    ...
    public abstract void dessine();
}
```

TD 10 - Échange de données avec un programme

Catal.java - Demo 10.1 (avec motif)

```
// Ajouter le membre :
private String motif;
// Modifier le constructeur :
public Catal(int abs, int ord, int t, String mot, Graphics gr) {
    super(abs,ord,gr);
    setTaille(t);
    motif = new String(mot);
    r = new Random();
}
// Ajouter la méthode :
public String getMot() {
    return motif;
}
```

Complexe.java - Demo 10.1 (avec motif)

```
// Modifier les constructeurs :
public Complexe (int posX, int posY, int tCat, String sq, Phosphate ph, String mot, Graphics gr) {
    cat = new Catal (posX, posY, tCat, mot, gr);
    sub = new Proteine (sq,cat.getXsite(ph), cat.getYsite (ph), ph, gr);
    cat.fixe (ph);
}
public Complexe (int posX, int posY, int tCat, String mot, Graphics gr) {
    cat = new Catal (posX, posY, tCat, mot, gr);
    sub = null;
}
// Modifier la méthode :
public boolean motInSub() {
    if(sub!=null)
        return sub.motifPresent(cat.getMot());
    else
        return false;
}
// Modifier la méthode :
public void update (Observable obs, Object arg) {
    ...
    if ( (sub!=null) && (sub.getConfig()=='N') && (motInSub())) {
    ...
    }
}
```

Observation.java - Demo 10.1 (avec motif)

```
// Modifier l'initialisation :
cplx = new Complexe(300,300,80,arg[0],g);
```

Observation.java - Demo 10.2 (nombre de tours en paramètre)

```
// Modifier la boucle principale :
int ntrs = Integer.parseInt(arg[1]);
for(int t=0;t<ntrs;t++) {
    ...
}
```

Observation.java - Exercice 10.1 (nombre d'objets en paramètre)

```
// Modifier :
int nobj = 2*Integer.parseInt(arg[2]);
molb = new MolMob[nobj];
for(int i=0;i<nobj;i+=2) {
    ...
}
...
for(int i=0;i<nobj;i++)
    molb[i].vivre();
...
```

Observation.java - Demo 10.3 (vérification du nombre de tours)

```
// Modifier :
int ntrs;
try {
    ntrs = Integer.parseInt(arg[1]);
}
catch(NumberFormatException e) {
    System.out.println("<" + arg[1] + "> incorrect : simulation pendant 1000 tours");
    ntrs = 1000;
}
```

Observation.java - Exercice 10.2 (vérification du nombre d'objets)

```
// Modifier :
int nobj;
try {
    nobj = 2*Integer.parseInt(arg[2]);
}
catch(NumberFormatException e) {
    System.out.println("<" + arg[2] + "> incorrect : 100 objets");
    nobj = 100;
}
```

Observation.java - Demo 10.4 (fichier de sortie)

```
// Ajouter :
import java.io.*;
// Modifier :
try {
    File f_out = new File("ProPhos");
    FileWriter fw = new FileWriter(f_out);
    PrintWriter pw = new PrintWriter(fw);
    for(int t=0;t<ntrs;t++) {
        g.setColor (Color.white);
        g.fillRect (0,0,600,600);
        cplx.dessine();
        for(int i=0;i<nobj;i++)
            molb[i].vivre();
        int nbProP=0;
        for(int i=0;i<nobj;i++)
            if((molb[i] instanceof Proteine) && (((Proteine)molb[i]).getConfig()=='P') )
                nbProP++;
        pw.println(t+" "+nbProP);
    }
    pw.close();
}
catch (FileNotFoundException e) {
    System.out.println("Erreur de fichier : ProPhos");
}
catch (IOException e) {
    System.out.println("Erreur d'entree / sortie : ProPhos");
}
```

Observation.java - Demo 10.5 (fichier d'entrée)

```
// Modifier :
try {
    FileReader r = new FileReader("Prots.fasta");
    BufferedReader br = new BufferedReader(r);
    String str;
    for(int i=0;i<nobj;i+=2) {
        molb[i] = new Phosphate (g);
        molb[i].addObserver (cplx);
        str = br.readLine(); // Pour lire l'en-tete (on n'en fait rien)
        str = br.readLine();
        molb[i+1] = new Proteine (str, null,g);
        molb[i+1].addObserver (cplx);
    }
    r.close();
}
catch (FileNotFoundException e) {
    System.out.println("Erreur de fichier : Prots.fasta");
}
```

```
catch (IOException e) {
    System.out.println("Erreur d'entree / sortie : Prots.fasta");
}
```

Observation.java - Exercice 10.3 (question à l'utilisateur)

```
// Modifier :
int nobj;
String strk = "";
try {
    InputStreamReader isr = new InputStreamReader(System.in);
    BufferedReader flux = new BufferedReader(isr);
    System.out.print("Nombre de Proteines ? = ");
    strk = flux.readLine();
    nobj = 2*Integer.parseInt(strk);
}
catch(IOException e) {
    System.out.println("<"+strk+"> incorrect : 100 objets");
    nobj = 100;
}
```

