MOOC Intro POO Java
Corriges semaine 1

Les corrigés proposés correspondent à l'ordre des apprentissages : chaque corrigé correspond à la solution à laquelle vous pourriez aboutir au moyen des connaissances acquises jusqu'à la semaine correspondante.

Exercice 1 : cercle (classes et objets)

Voici une solution possible :

```
class TestCercle
{
/**
 * Programme principal testanr les fonctionnalités de la classe Cercle
 */
    public static void main(String[] args)
        {
            Cercle c1 = new Cercle();
            Cercle c2 = new Cercle();
            Cercle c3 = new Cercle();

            c1.setCentre(1.0, 2.0);
            c1.setRayon(Math.sqrt(5.0)); // passe par (0, 0)

            c2.setCentre(-2.0, 1.0);
            c2.setRayon(2.25);  // 2.25 > sqrt(5) => inclus le point (0, 0)

            c3.setCentre(-2.0, -5.0);
            c3.setRayon(1.0);

            System.out.println("Surface de c1 : " +  c1.surface());
            System.out.println("Surface de c2 : " +  c2.surface());
            System.out.println("Surface de c3 : " +  c3.surface());

            afficherPosition("c1", c1, 0.0, 0.0);
            afficherPosition("c2", c2, 0.0, 0.0);
            afficherPosition("c3", c3, 0.0, 0.0);
        }

    /* Méthode utilitaire affichant si un point est en dehors ou en dedans
     * d'un cercle donné
     * @param c : le cercle
     * @param x, y: les coordonnées du point
     * @paramm nom : le nom donné au cercle pour l'affichage
     */
    static void afficherPosition(String nom, Cercle c, double x, double y)
        {
            System.out.print("Position du point (" + x + ", " + y + ") : ");

            if (c.estInterieur(x,y))
            {
                System.out.print("dans ");
            }
            else
            {
                System.out.print("hors de ");
            }
            System.out.println(nom);
        }

}

/* Classe Cercle
 */

class Cercle {
    private double rayon;
    private double x;  // abscisse du centre
    private double y;  // ordonnée du centre

    // calcul de la surface du cercle
```

```
    public double surface() { return Math.PI * rayon * rayon; }

    /* méthode testant la position d'un point par rapport au cercle
     * @return : true si le point de coordonnées unX et unY est
     * dans le cercle
     */

    public boolean estInterieur(double unX, double unY) {
        return (((unX-x) * (unX-x) +
                 (unY-y) * (unY-y))
                <= rayon * rayon);
    }

    public void setCentre(double unX, double unY) {
       x = unX;
       y = unY;
    }


    public void setRayon(double r) {
       if (r < 0.0) r = 0.0;
       rayon = r;
    }

}
```

Exercice 2 : tour de magie (classes et objets)

Voici une solution possible :

```
import java.util.Scanner;

class Magie
{
  // L'histoire générale :
    public static void main(String[] args) {

        Spectateur thorin = new Spectateur();    // Il était une fois un spectateur...
        thorin.arriver();                        // ...qui venait voir un spectacle (!!)...

        Magicien gandalf = new Magicien();       // ...où un magicien...
        Assistant bilbo = new Assistant();       // ...et son assistant...
        gandalf.tourDeMagie(bilbo, thorin);      // ...lui firent un tour fantastique
    }
}

// un bout de papier... pour ce tour de magie
class Papier {

    // les données sur le papier
    private  int age;
    private int argent;

    // on peut écrire sur le papier
    public void ecrire(int unAge, int unMontant) {
        age    = unAge;
        argent = unMontant;
  }

    // et on peu lire les données depuis le papier
    public int lireAge() { return age   ; }
    public int lireArgent() { return argent; }
}

// ---------------------------------------
class Assistant {
    /* l'assistant mémorise dans son cerveau les valeurs lues
     * et le resultat du calcul.
     */
    private int ageLu;
    private int argentLu;
    private int resultat;

    public  void lire(Papier billet) {

        System.out.println("[Assistant] (je lis le papier)");
        ageLu    = billet.lireAge();
        argentLu = billet.lireArgent();
    }

    public void calculer() {
        System.out.println("[Assistant] (je calcule mentalement)");
        resultat = ageLu * 2;
        resultat += 5;
        resultat *= 50;
        resultat += argentLu;
        resultat -= 365;
    }

    public int annoncer() {
        System.out.println("[Assistant] J'annonce : " + resultat + " !" );
        return resultat;
    }
}

// ---------------------------------------
class Spectateur {
    // pour le moment a prendre comme tel (vu dans le MOOC précédent)
    // nous y reviendrons dans ce cours:
    private final static Scanner clavier = new Scanner(System.in);
```

```
    // Les spécificités du spectateur
    private int age;
    private int argent;


    // lorsqu'il entre dans la salle (avant il n'existe pas pour nous)

    public void arriver() {
        System.out.println("[Spectateur] (j'entre en scène)");
        System.out.print("Quel âge ai-je ? ");

        age = clavier.nextInt();

        do {
            System.out.print("Combien d'argent ai-je en poche (<100) ? ");
            argent = clavier.nextInt();
        } while (argent >= 100);
        System.out.println("[Spectateur] (j'ai un montant qui convient)");
    }

    /* Dans cette modélisation on suppose que le papier
       n'appartient à personne : il a été par exemple trouvé
       dans la salle de spectacle
    */

    // écrit sur un papier
    public void ecrire(Papier billet) {
        System.out.println("[Spectateur] (j'écris le papier)");
        billet.ecrire(age, argent);
    }
}


// ---------------------------------------
class Magicien {

    // ce que le magicien doit deviner:
    private int ageDevine;
    private int argentDevine;

    // pour faire son tour, le magicien a besoin d'au moins
    // un spectateur et d'un assistant
    public void tourDeMagie(Assistant fidele, Spectateur quidam) {
        Papier billet = new Papier();
        System.out.println("[Magicien] un petit tour de magie...");
        // le magicien donne ses instructions :
        quidam.ecrire(billet);
        fidele.lire(billet);
        fidele.calculer();
        calculer(fidele.annoncer());
        annoncer();
    }
    // partie privée ici car seul le magicien sait ce qu'il doit
    // faire dans son tour
    private void calculer(int resultatRecu) {
        resultatRecu += 115;
        ageDevine    = resultatRecu / 100;
        argentDevine = resultatRecu % 100;
    }

    private void annoncer() {
        System.out.println("[Magicien] "
                           + "  - hum... je vois que vous êtes agé de "
                           + ageDevine + " ans ");
        System.out.println("    et que vous avez " + argentDevine + " francs  suisses en poche !");
}

}
```

Exercice 3 : géométrie (classes et objets)

Voici une solution possible :

```
import java.util.Scanner;

class Geometrie {

/**
 * Le programme principal se contente de construire un
 * triangle, d'afficher son périmètre et d'afficher
 * s'il est isocèle ou non.
 */
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        Point p1 = new Point();
        Point p2 = new Point();
        Point p3 = new Point();
        Triangle t = new Triangle();

        initPoint(p1);
        initPoint(p2);
        initPoint(p3);

        t.setSommets(p1, p2, p3);

        double perimetre = t.calculerPerimetre();
        System.out.println("Perimetre : " + perimetre);
        boolean isocele = t.testerIsocele();
        if (isocele)
            System.out.println("Le triangle est isocèle");
        else
            System.out.println("Le triangle n'est pas isocèle");
    }

    /* Initialisation d'un point
     * on fait le choix de traiter les données entrées par l'utilisateur
     * en dehors de la méthode d'initialisation de l'objet Point
     * ceci permettrait de garantir que l'on ne fournit que des valeurs
     * valides à la méthode initilalisant le point
     * (non traités ici mais que vous pouvez ajouter en extension)
     */
    static void  initPoint(Point p) {
        double x = 0;
        double y = 0;
        System.out.println("Construction d'un nouveau point");
        System.out.print("   Veuillez entrer x : ");
        x = scanner.nextDouble();
        System.out.print("   Veuillez entrer y : ");
        y = scanner.nextDouble();
        // eventuellement des tests d'intégrité des données lues
        // et donner plusieurs chances de saisie à l'utilisateur
        p.init(x,y);
    }
}


class Triangle {

    private Point p1, p2, p3;
    private double longueur1, longueur2, longueur3;

    /**
     * Affecte des valeurs aux sommets
     * à remplacer impérativement par un constructeur adéquat
     *  dès la semaine prochaine
     */
    public void setSommets(Point point1, Point point2, Point point3) {
        p1 = point1;
        p2 = point2;
        p3 = point3;
        // Les distances sont calculées et stockées dans des
        // attributs. Les méthodes calculerPerimetre et testerIsocele
        // peuvent ainsi accéder aux valeurs précalculées et nous évitons
```

```
        // de les recalculer plusieurs fois.

        longueur1 = p1.calculerDistance(p2);
        longueur2 = p2.calculerDistance(p3);
        longueur3 = p3.calculerDistance(p1);
    }

    /**
     * Calcul du perimètre de l'instance courante (this).
     * @return le perimetre sous la forme d'un double
     */
    public double calculerPerimetre() {
        return (longueur1 + longueur2 + longueur3);
    }

    /**
     * Teste si l'instance courante (this) est un triangle isocèle
     * @return true si le triangle est isocèle et false sinon
     */
    public boolean testerIsocele() {
        return (longueur1 == longueur2
                    || longueur2 == longueur3
                    || longueur3 == longueur1);
    }
}



class Point {

    private double x, y;

    /**
     * encore un constructeur en devenir
     */
    public void init(double x, double y) {
        this.x = x;
        this.y = y;
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

    /**
     * Calcule la distance entre this et un point p
     * @param p un Point par rapport auquel on calcule la distance
     * @return la distance de this à p
     */
    public double calculerDistance(Point p) {
        // Calcule la distance entre deux points. Le premier point est
        // l'objet actuel (this). Le deuxième point (p) est envoyé en
        // paramètre.
        double x1 = this.x;
        double y1 = this.y;

        double x2 = p.getX();
        double y2 = p.getY();

        double xdiff = x1 - x2;
        double ydiff = y1 - y2;

        double somme = xdiff * xdiff + ydiff * ydiff;
        double distance = Math.sqrt(somme);

        return distance;
    }
}
```
