1) *instanceof*

Donner le résultat de l'exécution de ce programme.

%%

```
public class TstPcol3 {
public static void main (String args[]){
    Point pc1 = new Point(3,5);
    System.out.print("PC1:");
    pc1.affiche();

    Pointcol pc2 = new Pointcol(5,8,(byte)2);
    System.out.println("");
    System.out.print("PC2:");
    pc2.affiche();
    System.out.println("");
    System.out.println("PC2 deplace 1,-3:");
    pc2.deplace(1,-3);
    pc2.affiche();

    Point pc3 = new Pointcol(50,80,(byte)20);
    System.out.println("");
    System.out.print("PC3:");
    pc3.affiche();

    System.out.println("pc1 est une instance de Object " + (pc1 instanceof Object));
    System.out.println("pc1 est une instance de Point " + (pc1 instanceof Point));
    System.out.println("pc1 est une instance de Pointcol " + (pc1 instanceof Pointcol));

    System.out.println("pc2 est une instance de Object " + (pc2 instanceof Object));
    System.out.println("pc2 est une instance de Point " + (pc2 instanceof Point));
    System.out.println("pc2 est une instance de Pointcol " + (pc2 instanceof Pointcol));

    System.out.println("pc3 est une instance de Object " + (pc3 instanceof Object));
    System.out.println("pc3 est une instance de Point " + (pc3 instanceof Point));
    System.out.println("pc3 est une instance de Pointcol " + (pc3 instanceof Pointcol));
}
}
```

%%

2) Méthode *toString()*

Proposer une classe *Personne* proposant 4 données membres : *String nom*, *String prenom*, *Personne père*, *Personne mere* et deux constructeurs permettant d'initialiser avec des paramètres d'entrée soit les 2 premières données soit toutes les 4. Proposer une méthode *toString()* qui sera appelée de manière implicite. Proposer ensuite une autre classe *TestPersonnes* avec la méthode principale créant des objets.

3) Transtypage

Donner le résultat de l'exécution de ce programme.

%%

```
import java.io.*;
class Mere
{   int a, b;
    Mere(){ a=1; b=2;       }
}
class Fille extends Mere
{   int c;
    Fille() { a=10;         b=20;           c=30;   }
}
public class Transtypage6
{   public static void main(String args[])
    {       Mere om1=new Mere();
            Mere om2=new Mere();
            Mere om3=new Mere();
```

```
                Fille of1=new Fille();
                Fille of2=new Fille();
                Fille of3=new Fille();
                System.out.println("om1 "+om1.a + " " +om1.b );
                System.out.println("om2 "  +om2.a + " " +om2.b );
                System.out.println("of2 "+of2.a + " " +of2.b + " " +of2.c);
                om1=of1;
                System.out.println("om1 "+om1.a + " " +om1.b );
                om2=(Mere)of1;
                System.out.println("om2 "+om2.a + " " +om2.b );

                of2=(Fille)om3;
                of3=om1;
        }
}
```

%%%

4) Transtypage

Donner le résultat de l'exécution de ce programme.

%%%

```
import java.io.*;
class Mere
{    int a, b;
            Mere() {        a=1; b=2;       }
}
class Fille extends Mere
{    int c;
            Fille() { a=10;         b=20;           c=30;   }
}
public class Transtypage6
{    public static void main(String args[])
     {      Mere om1=new Mere();
                    Mere om2=new Mere();
                    Mere om3=new Mere();
                    Fille of1=new Fille();
                    Fille of2=new Fille();
                    Fille of3=new Fille();
                    System.out.println("om1 "+om1.a + " " +om1.b );
                    System.out.println("om2 "  +om2.a + " " +om2.b );
                    System.out.println("of2 "+of2.a + " " +of2.b + " " +of2.c);
                    om1=of1;
                    System.out.println("om1 "+om1.a + " " +om1.b );
                    om2=(Mere)of1;
                    System.out.println("om2 "+om2.a + " " +om2.b );
                    of2=(Fille)om3;
                    of3=om1;
            }
}
```

%%%

5) Classes abstraites *abstract*

Proposer une classe ***Affichable*** proposant 1 donnée membre : ***String nom*** et 1 méthode abstraite *affiche()* ne retournant rien. Proposer une autre classe *Entier* héritant de *Affichable* proposant un constructeur, la méthode ***affiche()*** affichant les données membres et une donnée membre valeur entière. Proposer ensuite une dernière classe *Test_Abstract* avec la méthode principale créant des objets des deux classes et appelant la méthode ***affiche()***.

6) Classes abstraites *abstract*

Reprenez l'exercice précédent avec la classe ***Affichable*** et ***Entier***. Compléter en ajoutant une autre classe ***Reel*** héritant aussi de ***Affichable*** proposant un constructeur, la méthode affiche et une donnée membre valeur réelle.

Proposer ensuite une dernière classe ***Test_Abstract2*** avec la méthode principale créant un tableau d'objets des trois autres classes et appelant la méthode ***affiche()***.

7) Classes abstraites

Proposer une classe abstraite ***FormeGeom*** proposant :

- 2 données membres ***posX*** et ***posY***,
- 2 méthodes
  - ***deplacer(double, double)*** initialisant les données membres,
  - ***afficher()*** permettant d'afficher le contenu de mes données membres.
- 2 méthodes abstraites ***perimetre()*** et ***surface()*** retournant des doubles.

Proposer deux classes non abstraites qui héritent de la classe ***FormeGeom*** .

- Rectangle a :
  - 2 données membres ***largeur*** et ***longueur***,
  - 2 constructeurs
    - Celui sans paramètre d'entrée permettant d'initialiser à des valeurs par défaut les données membres.
    - Celui avec 4 paramètres d'entrée permettant d'initialiser à des valeurs passées en paramètre d'entrée les données membres.
- Cercle a
  - une seule donnée membre ***rayon***.
  - 2 constructeurs
    - Celui sans paramètre d'entrée permettant d'initialiser à des valeurs par défaut les données membres.
    - Celui avec 3 paramètres d'entrée permettant d'initialiser à des valeurs passées en paramètre d'entrée les données membres.

Dans la classe ***ProgGeom***, contenant la méthode principale, créer des objets et afficher les résultats de l'appel de ces méthodes qui retournent le résultat.

8) Interface

- Proposer une interface ***Forme*** proposant deux méthodes ***perimetre()*** et ***surface()***.
- Proposer deux classes qui implémentent cette interface.
  - ***Rectangle*** a de données membres ***largeur*** et ***longueur***,
  - ***Cercle*** a une seule donnée membre ***rayon***.
- Dans la classe ***TestInterface***, créer des objets et afficher les résultats de l'appel de ces méthodes qui retournent le résultat.