

Solutions Premiers Pas

Solution 1:

```
public class HelloWorld {
public static void main(String[] args) {
System.out.println("Hello World!");
}
}
```

Solution 2 :

```
public class HelloWorld {
public static void main(String[] args) {
System.out.println(args[0]);
}
}
```

Solution 3 :

```
public class HelloWorldWithMethod{
public void Hello(){
System.out.print("You're Welcome!");
}
public static void main(String[] args) {
HelloWorldWithMethod s=new HelloWorldWithMethod();
s.hello();
}
}
```

Solution 4 :

```
public class HelloWorldWithAttribut {
String maChaine;
public void Hello(){
System.out.print(maChaine);
}
public static void main(String[] args) {
HelloWorldWithAttribut s=new HelloWorldWithAttribut ();
s.maChaine="salut";
s.hello();
}
}
```

Solutions Généralités

Solution 1 :

```
class Somme{
    public static void main(String[] arg){
        int i, somme = 0;
        for (i = 1; i <= 100; i++){
            somme += i;
        }
        System.out.println("Voila la somme des 100 " + " premiers entiers : " + somme);
    }
}
```

Solution 2

```
public class Segment{
    int extr1, extr2;
    Segment(int e1, int e2){
        extr1 = e1;
        extr2 = e2;
    }
    void ordonne(){
        if (extr1 > extr2){
            int tampon;
            tampon = extr1;
            extr1 = extr2;
            extr2 = tampon;
        }
    }
    int longueur(){
        ordonne();
        return extr2 - extr1;
    }
    boolean appartient(int x){
        return (x - extr1) * (x - extr2) <= 0;
    }
    public String toString(){
        ordonne();
        return "segment [" + extr1 + ", " + extr2 + "]";
    }
}

public class EssaiSegment{
    public static void main(String[] argv){
        Segment s = new Segment(Integer.parseInt(argv[0]), Integer.parseInt(argv[1]));
        int point;
        System.out.println("Longueur du " + s + " : " + s.longueur());
        point = Integer.parseInt(argv[2]);
        if (s.appartient(point))
            System.out.println(point + " appartient au " + s);
        else
            System.out.println(point + " n'appartient pas au " + s);
    }
}
```

Solution 4

```
public class Factorielle{
    public static void main(String[] argv){
        int n, factorielle = 1 ;
        n = Integer.parseInt(argv[0]);
        for (int i = 2; i <= n; i++)
            factorielle *= i;
        System.out.println("Voila la factorielle de " + n + " : " + factorielle);
    }
}
```

Solution 5

Première méthode

```
public class Palindrome{
    static String inverse(String s){
        int longueur = s.length();
        char [] envers = new char[longueur];
        int i;
        for (i = 0; i < longueur; i++)
            envers[i] = s.charAt(longueur - i - 1);
        return new String(envers);
    }
    public static void main(String[] arg)    {
        String chaine = arg[0];
        String autre = inverse(chaine);
        System.out.println("L'inverse de " + chaine + " est " + autre);
        if (autre.equals(chaine))
            System.out.println(chaine + " est un palindrome");
        else
            System.out.println(chaine + " n'est pas un palindrome");
    }
}
```

Deuxième méthode

```
import java.io.*;
public class Palindrome2{
    static String inverse(String s){
        int longueur = s.length();
        char [] envers = new char[longueur];
        int i;
        for (i = 0; i < longueur; i++)
            envers[i] = s.charAt(longueur - i - 1);
        return new String(envers);
    }
    public static void main(String[] arg) throws IOException{
        BufferedReader entree = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Indiquez la chaîne de caractères");
        String chaine = entree.readLine();
        String autre = inverse(chaine);
        System.out.println("L'inverse de \" + chaine + "\" est \" + autre + "\"");
        if (autre.equals(chaine))
            System.out.println("\" + chaine + "\" est un palindrome");
        else
            System.out.println("\" + chaine + "\" n'est pas un palindrome");
    }
}
```

Solution 6

```
public class Vecteur{
    int abs, ord;
    Vecteur(int _abs, int _ord){
        abs = _abs;
        ord = _ord;
    }
    double longueur(){
        return Math.sqrt(abs * abs + ord * ord);
    }
    void addition(Vecteur v){
        abs = v.abs + abs;
        ord = v.ord + ord;
    }
    static Vecteur addition(Vecteur v1, Vecteur v2){
        return new Vecteur(v1.abs + v2.abs, v1.ord + v2.ord);
    }
    boolean plusPetitQue(Vecteur v){
        return longueur() < v.longueur();
    }
    public String toString(){
        return "vecteur (" + abs + ", " + ord + ")";
    }
}

public class EssaiVecteur{
    public static void main(String[] argv){
        Vecteur v1 = new Vecteur(Integer.parseInt(argv[0]), Integer.parseInt(argv[1]));
        System.out.println("v1 : " + v1);
        System.out.println("Longueur de " + v1 + " : " + v1.longueur());
        Vecteur v2 = new Vecteur(Integer.parseInt(argv[2]), Integer.parseInt(argv[3]));
        System.out.println("v2 : " + v2);
        if (v1.plusPetitQue(v2))
            System.out.println("Le " + v1 + " est plus petit que le " + v2);
        else
            System.out.println("Le " + v1 + " est au moins aussi grand que " + "le " + v2);
        v1.addition(v2);
        System.out.println("v1 après addition de v2 : " + v1);
        Vecteur v3 = Vecteur.addition(v1, v2);
        System.out.println("v3 = v1 + v2 : " + v3);
    }
}
```

Solution 7 :

```
class ExceptionPileVide extends Exception{
    public String toString(){
        return "la pile est vide";
    }
}
```

Méthode1

```
public class Pile1{
    private int taille = 3;
    private int increment = 2;
    private int[] tableau = new int[taille];
    private int hauteur; // nombre d'entiers rangés
```

```

void empiler(int n){
    if (hauteur == taille){
        taille = taille + increment;
        int[] tableauBis = new int[taille];
        for (int i = 0; i < hauteur; i++)
            tableauBis[i] = tableau[i];
        tableau = tableauBis;
    }
    tableau[hauteur] = n;
    hauteur++;
}
int depiler() throws ExceptionPileVide{
    int cle;
    if (estVide()) throw new ExceptionPileVide();
    hauteur--;
    return tableau[hauteur];
}
boolean estVide(){
    return hauteur == 0;
}
}

```

Méthode 2

```

public class Maillon {
    int donnee;
    Maillon suivant;
    Maillon(int donnee, Maillon suivant) {
        this.donnee = donnee; this.suivant = suivant;
    }
}
public class Pile2{
    private Maillon debut;
    void empiler(int n) {
        debut = new Maillon(n, debut);
    }
    int depiler() throws ExceptionPileVide {
        int cle;
        if (estVide()) throw new ExceptionPileVide();
        cle = debut.donnee;
        debut = debut.suivant;
        return cle;
    }
    boolean estVide() {
        return debut == null;
    }
}
}

```