TD N °1

JAVA

M2207 Consolidation des bases de la programmation

ABDOUL-AZID Amine

Mercredi 30 jan

Exercice 1 :

CODE :

```
public class Bonjour {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Bonjour");
    }

}
```

Résultat :

```
Bonjour
```

Il suffit d'utiliser un System.out.print ()

Exercice 2 : Correction des programmes

Exercice 2) a) : Scope

Code :

```
1  public class scope {
2      public static void main(String[] args) {
3              int i=0;
4              for ( i=0 ; i<5 ; i++) {System.out.print(i+",");}
5              System.out.print("\n");
6  }
7  }
```

Résultat :

```
0,1,2,3,4,
```

Car la variable était déclaré en double on a corrigé en retirant le *int* dans le *for* ou bien le *int* déclaré avant le *for*.

2ème méthode : Avec le int uniquement dans la boucle for :

```java
public class scope {
    public static void main(String[] args) {
            for ( int i=0 ; i<5 ; i++) {System.out.print(i+",");}
            System.out.print("\n");
}
}
```

Exercice 2) b) : Variables

1ère méthode :

```java
public class Variables {

    public static void main(String[] args) {
            double a=3.0 ;
            float b=4;
            double c;
            c=Math.sqrt(a*a+b*b);
            System.out.print("c="+c);

    }

}
```

Résultat :

```
c=5.0
```

Car c est de type double il faut que a soit aussi de type double ce qui est le plus simple car c'est le seul qui est décimal.

2ème méthode :

Le type double est affecté à b avec la virgule

```java
public class Variables {

    public static void main(String[] args) {
            float a=3 ;
            double b=4.0;
            double c;
            c=Math.sqrt(a*a+b*b);
            System.out.print("c="+c);

    }

}
```

3ème méthode : On utilise le caste par ex : pour a

```java
public class Variables {

    public static void main(String[] args) {
            float a=(float) 3.0 ;
            double b=4;
            double c;
            c=Math.sqrt(a*a+b*b);
            System.out.print("c="+c);

    }

}
```

4ème méthode : avec un caste devant le double de sqrt :

```java
public class Variables {

    public static void main(String[] args) {
            float a=(float) 3.0 ;
            double b=4;
            double c;
            c=(float)Math.sqrt(a*a+b*b);
            System.out.print("c="+c);

    }

}
```

Exercice 2) a) : Promote

```java
public class Promote {

    public static void main(String[] args) {
            byte b=42;
            char c= 'a';
            short s=1024;
            int i =50000;
            float f =5.67f;
            double d=.1234;
            double resultat=(f*b)+(i/c)-(d*s);
            System.out.print((f*b)+"+"+(i/c)+"-"+(d*s));
            System.out.println("="+resultat);

            byte b2=10;
            short b3= (short) (b2*b);
            System.out.println("b3="+b3);
    }

}
```

Résultat :

```
238.14+515-126.3616=626.7784146484375
b3=420
```

Les types *short* ou *int* ou *long* peuvent être utilisés pour « caster » la valeur car la valeur nécessite 2 octets.

Voire de cette manière :

```java
public class Promote {

    public static void main(String[] args) {
            byte b=42;
            char c= 'a';
            short s=1024;
            int i =50000;
            float f =5.67f;
            double d=.1234;
            double resultat=(f*b)+(i/c)-(d*s);
            System.out.print((f*b)+"+"+(i/c)+"-"+(d*s));
            System.out.println("="+resultat);

            byte b2=10;
            int b3= (short) (b2*b);
            System.out.println("b3="+b3);
    }

}
```

Affecter le type *int* à b3 et caster avec un *short*.

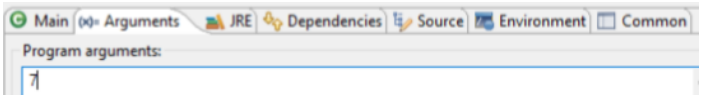
Afin que le résultat occupe les bits pour s'afficher correctement

Il faut absolument des + de 2 octets d'allocations.

Exercice 3 :

Exercice 1) a) : Arg= 2

On saisit la valeur dans arguments :



Alors si on lance la commande :

$java Array 2

Arg=2

```
1  public class Array {
2      public static void main(String[] args) {
3              String jour_semaine[] = new String [7];
4              int jour=Integer.parseInt(args[0]);
5              jour_semaine[0]="dimanche";
6              jour_semaine[1]="lundi";
7              jour_semaine[2]="mardi";
8              jour_semaine[3]="mercredi";
9              jour_semaine[4]="jeudi";
10             jour_semaine[5]="vendredi";
11             jour_semaine[6]="samedi";
12             System.out.println ( "Moi, je préfère le " +jour_semaine[jour]);
13     }
14
15 }
```

Résultat :

```
Moi, je préfère le mardi
```

La chaîne est concaténée avec la valeur de l'index 2 est affiché c'est-à-dire mardi.

Exercice 1) b) : Arg=7

Alors si on lance la commande :

$java Array 7

Arg=7

```
1  public class Array {
2      public static void main(String[] args) {
3              String jour_semaine[] = new String [7];
4              int jour=Integer.parseInt(args[0]);
5              jour_semaine[0]="dimanche";
6              jour_semaine[1]="lundi";
7              jour_semaine[2]="mardi";
8              jour_semaine[3]="mercredi";
9              jour_semaine[4]="jeudi";
10             jour_semaine[5]="vendredi";
11             jour_semaine[6]="samedi";
12             System.out.println ( "Moi, je préfère le " +jour_semaine[jour]);
13     }
14
15 }
```

Résultat :

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 7
        at Array.main(Array.java:12)
```

Un message d'erreur est affiché. C'est normal car il est hors index. Car le [8] n'est pas défini.

Exercice 4 :

Exercice 1) a) : Méthode BuildString appelant KeepChar

1ère solution :

```
public class ex4a {

    public static void main(String[] args) {
            String machaine= buildString("Microsoft");
            System.out.println(machaine);


    }
    public static String buildString(String s) {
        String r="";
        for(int i=0;i<s.length();i++) {
            if(keepChar(s.charAt(i),i)) {
                    r=r+s.charAt(i);
            }
        }
        return r;
    }

    public static boolean keepChar(char c, int position) {
        return true;
    }
}
```

Résultat :

```
Microsoft
```

La méthode keepChar retourne vrai pour toutes les positions et les caractères.

Cette méthode buildString permet de placer l'intégralité d'une chaîne de caractère dans sa chaîne vide en faisant appel à la méthode keepChar

La chaîne est vide puis elle ajoute la valeur du caractère pour chaque position jusqu'à la valeur finale (grâce à la longueur de la chaîne transmise à buildString) sachant que la position/i est initialisée à 0 elle ne peut retourner que les jusqu'au dernier caractère. Ici il y a 9 caractères, le dernier i vaut 8.

On vient d'effectuer de la manipulation de chaîne de caractère.

2ème solution :

```
public class ex4a {

    public static void main(String[] args) {
        System.out.println(buildString("Microsoft"));


    }
    public static String buildString(String s) {
        String r="";
        for(int i=0;i<s.length();i++) {
            if(keepChar(s.charAt(i),i)) {
                    r=r+s.charAt(i);
            }
        }
        return r;
    }

    public static boolean keepChar(char c, int position) {

            return true;
        }
}
```

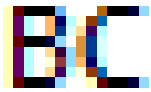
Elle évite la déclaration d'un String et d'une ligne …

En transmettant la valeur (chaîne de caractère à la méthode buildString à l'intérieur du print (""))

Exercice 2)a) : Garder un caractère sur deux :

```
2  public class ex4a {
3
4      public static void main(String[] args) {
5              String machaine= buildString("NBVCX");
6              System.out.println(machaine);
7
8
9      }
10     public static String buildString(String s) {
11         String r="";
12         for(int i=0;i<s.length();i++) {
13             if(keepChar(s.charAt(i),i)) {
14                     r=r+s.charAt(i);
15             }
16         }
17         return r;
18     }
19
20     public static boolean keepChar(char c, int position) {
21         if (position%2==1) {
22         return true;
23         }else {
24             return false;
25         }
26
27 }
```

Résultat :



Le résultat de cette chaîne est : BC

La méthode buildString : Retourne les caractères d'index impair en d'autres termes 1 caractère sur 2

Les caractères sont initialisés à la position 1.

Si l'on change : la valeur correspond à celle de la position `if (position%2==1)`

Exercice 3)a) : Garder un caractère sur trois

```
1  public class ex4a {
2
3      public static void main(String[] args) {
4              String machaine= buildString("ABCDEF");
5              System.out.println(machaine);
6
7
8      }
9      public static String buildString(String s) {
10         String r="";
11         for(int i=0;i<s.length();i++) {
12             if(keepChar(s.charAt(i),i)) {
13                     r=r+s.charAt(i);
14             }
15         }
16         return r;
17     }
18
19     public static boolean keepChar(char c, int position) {
20         if (position%3==1) {
21         return true;
22         }else {
23             return false;
24         }
25         }
26 }
27
```

Résultat :

```
BE
```

On effectue un modulo par 3 pour afficher 1 caractère sur 3 de la chaîne paramètre.

Exercice 3)b) : Renvoie que les caractères

```
1  public class ex4a {
2
3      public static void main(String[] args) {
4              String machaine= buildString("AB2C3D6EF");
5              System.out.println(machaine);
6
7
8      }
9      public static String buildString(String s) {
10         String r="";
11         for(int i=0;i<s.length();i++) {
12             if(keepChar(s.charAt(i),i)) {
13                     r=r+s.charAt(i);
14             }
15         }
16         return r;
17     }
18
19     public static boolean keepChar(char c, int position) {
20         if (Character.isDigit(c)) {
21         return false;
22         }else {
23             return true;
24         }
25         }
26 }
```

Résultat :

```
ABCDEF
```

Il faut absolument modifier la condition en false puis en true car dans le cas inverse il ne conserve que les chiffres.

Les chiffres ne sont plus affichés, les @ ou les autres caractères sont toujours affichées. En effet les chiffres sont exclus de l'affichage par la méthode buildString.

Grâce à la méthode isDigit de la classe Character de la librairie par défaut java.lang

Annexes :

Exercice 2)a) : Garder un caractère sur deux : Modulo de Position par 2 égale à 0

```
public class ex4b {

public static void main(String[] args) {
    String machaine=buildString("MICROSOFT");
    System.out.println(machaine);

    }

    public static String buildString (String s) {
            String r="";
            for (int i=0;i<s.length();i++) {
                        if(keepChar(s.charAt(i),i)) {
                            r=r+s.charAt(i);
                        }
            }
            return r;
        }
    private static boolean keepChar(char c, int position) {
        if(position%2==0) {
            return true;
        }   else {
            return false;
        }
    }
}
```

```
<terminated> ex4b [Java Application] /usr/lib/jvm/java-8-openjdk-i3
MCOOT
```

Exercice 3)b) : Renvoie que les caractères : caractère spéciaux @

```
public class ex4d {

    public static void main(String[] args) {
    String machaine=buildString("AB2C3D6EF@");
    System.out.println(machaine);

    }

    public static String buildString (String s) {
            String r="";
            for (int i=0;i<s.length();i++) {
                        if(keepChar(s.charAt(i),i)) {
                            r=r+s.charAt(i);
                        }
            }
            return r;
        }
    private static boolean keepChar(char c, int position) {
        if(Character.isDigit(c)) {
            return false;/* false indispensable pour n'affi
        }   else {
            return true;
        }
    }

}
```

```
Problems  @ Javadoc  Declaration  Console
<terminated> ex4d [Java Application] /usr/lib/jvm/java-8-
ABCDEF@
```