# [CI2] Cours 5: Traduction de programmes II

Daniela Petrișan Université Paris Cité, IRIF





#### Dans ce cours

Nous poursuivons la traduction de programmes et nous nous intéressons aux appels de fonctions.

#### Dans ce cours

Nous poursuivons la traduction de programmes et nous nous intéressons aux appels de fonctions.

Nous avons déjà vu que lors d'un appel de fonction, il est nécessaire de transmettre les paramètres (appel par valeur) et de mémoriser une adresse de retour.

#### Dans ce cours

Nous poursuivons la traduction de programmes et nous nous intéressons aux appels de fonctions.

Nous avons déjà vu que lors d'un appel de fonction, il est nécessaire de transmettre les paramètres (appel par valeur) et de mémoriser une adresse de retour.

Dans le cours précédent, on a fait la traduction d'un appel de fonction très simple, pour une fonction sans paramètres. Dans ce cas, il suffisait de conserver l'adresse de retour dans une pile d'entiers.

```
public class Tiny {
 public static void f() {
   System.out.println("Hello"); // ic = 500
 } // ic = 501 : la sortie de l'appel de f
 public static void main(String[] args) {
   f(); // ic = 0 : appel a la fonction f
                                                        while(true)
        // ic = 1 : point de retour, nettoyage
 } // ic = 2 : sortie du programme
                                                        switch(ic){
```

```
import java.util.Stack;
public class TinyTrad {
  public static void main(String[] args) {
    int ic = 0; //compteur instruction
    Stack<Integer> p = new Stack<Integer>();
      case 0: p.push(ic+1); ic = 500; break;
             //@retour=1 sur la pile +
             //saut incond vers 500
     case 1: p.pop(); ic++; break;
             //on dépile
     case 2: System.exit(0):
      case 500: System.out.println("Hello");
                ic++: break:
     case 501: ic = p.peek(); break;
} } }
```

En général, chaque appel de fonction nécessite la mémorisation de certaines informations :

- l'adresse de retour, c'est-à-dire le numéro de l'instruction qui doit être exécutée au retour de l'appel
- les paramètres de la fonction initialisés avec les valeurs des paramètres effectifs
- la valeur de retour de la fonction (si le type de retour n'est pas vide)
- les variables locales de la fonction appelée

En général, chaque appel de fonction nécessite la mémorisation de certaines informations :

- l'adresse de retour, c'est-à-dire le numéro de l'instruction qui doit être exécutée au retour de l'appel
- les paramètres de la fonction initialisés avec les valeurs des paramètres effectifs
- la valeur de retour de la fonction (si le type de retour n'est pas vide)
- les variables locales de la fonction appelée

Ces informations forment le bloc d'activation (stack frame) de l'appel de fonction.

En général, chaque appel de fonction nécessite la mémorisation de certaines informations :

- l'adresse de retour, c'est-à-dire le numéro de l'instruction qui doit être exécutée au retour de l'appel
- les paramètres de la fonction initialisés avec les valeurs des paramètres effectifs
- la valeur de retour de la fonction (si le type de retour n'est pas vide)
- les variables locales de la fonction appelée

Ces informations forment le bloc d'activation (stack frame) de l'appel de fonction.

A priori, le compilateur ne connaît pas le nombre d'appels de fonction, donc ces blocs d'activation seront stockés dans une pile d'exécution (ou pile d'appels).

En général, chaque appel de fonction nécessite la mémorisation de certaines informations :

- l'adresse de retour, c'est-à-dire le numéro de l'instruction qui doit être exécutée au retour de l'appel
- les paramètres de la fonction initialisés avec les valeurs des paramètres effectifs
- la valeur de retour de la fonction (si le type de retour n'est pas vide)
- les variables locales de la fonction appelée

Ces informations forment le bloc d'activation (stack frame) de l'appel de fonction.

A priori, le compilateur ne connaît pas le nombre d'appels de fonction, donc ces blocs d'activation seront stockés dans une pile d'exécution (ou pile d'appels).

La durée de vie d'un bloc d'activation dépend de l'appel de fonction auquel il correspond. Le bloc est empilé lorsque l'appel de fonction est effectué et détruit après la fin de l'appel.

#### Bloc d'activation minimal

Le bloc d'activation minimal ne contient que l'adresse de retour, par exemple pour une fonction sans paramètres, avec le type de retour void et sans variables locales

```
public class Bloc {
   private int adresseRetour;

   public Bloc(int adresseRetour) {
      this.adresseRetour = adresseRetour;
   }

   public int getAdresse() {
      return this.adresseRetour;
   }
}
```

# La structure générale de la traduction d'un appel de fonction simple

```
public class AppelSimpleTrad {
  static int ic = 0:
  static Stack<Bloc> p = new Stack<Bloc>();
  public static void main(String[] args) {
  while(true) {
     switch(ic){
       . . .
       case n: p.push(new Bloc(ic+1)); // l'appel de fonction est lancé: on prépare le retour et
              ic = m: break:
                                                                  //on fait un saut inconditionnel
       case n+1: p.pop(); //le point de retour après l'appel et nettoyage : on dépile le bloc
                ic++: break:
       . . .
                                                                    //début de la fonction appelée
       case m:
       . . .
       case m': ic = p.peek().getAdress(); break; //sortie de la fonction appelée: on récupère
                                                          //l'adresse de retour et on fait le saut
1111
```

```
public class AppelSimple {
 public static void f() {
   System.out.println("Entree f()");
   g();
   System.out.println("Sortie f()");
 public static void g() {
   System.out.println("Je suis g()");
 public static void main(String[] args) {
   System.out.println("Entree main"):
   f();
   System.out.println("Sortie main");
```

```
public class AppelSimple {
 public static void f() {
   System.out.println("Entree f()");
   g();
   System.out.println("Sortie f()");
 public static void g() {
   System.out.println("Je suis g()");
 public static void main(String[] args) {
   System.out.println("Entree main"):
   f();
   System.out.println("Sortie main");
```

```
public class AppelSimple {
 public static void f() {
   System.out.println("Entree f()");
   g();
   System.out.println("Sortie f()");
 public static void g() {
   System.out.println("Je suis g()");
 public static void main(String[] args) {
                                                                   110
   System.out.println("Entree main"):
   f();
   System.out.println("Sortie main");
```

```
public class AppelSimple {
 public static void f() {
   System.out.println("Entree f()");
   g();
   System.out.println("Sortie f()");
 public static void g() {
   System.out.println("Je suis g()");
 public static void main(String[] args) {
   System.out.println("Entree main");
   f():
             // 1 empiler le bloc d'activation et saut inconditionnel
                    // 2 point de retour : dépiler le bloc d'activation
   System.out.println("Sortie main");
```

```
public class AppelSimple {
 public static void f() {
   System.out.println("Entree f()");
   g();
   System.out.println("Sortie f()");
 public static void g() {
   System.out.println("Je suis g()");
 public static void main(String[] args) {
   System.out.println("Entree main");
                                                                  110
   f():
             // 1 empiler le bloc d'activation et saut inconditionnel
                    // 2 point de retour : dépiler le bloc d'activation
   System.out.println("Sortie main");
                                                                  1/3
```

```
public class AppelSimple {
 public static void f() {
   System.out.println("Entree f()");
   g();
   System.out.println("Sortie f()");
 public static void g() {
   System.out.println("Je suis g()");
 public static void main(String[] args) {
   System.out.println("Entree main");
   f():
              // 1 empiler le bloc d'activation et saut inconditionnel
                    // 2 point de retour : dépiler le bloc d'activation
   System.out.println("Sortie main");
                                                                   1/3
                                                   // 4 sortie de main
```

```
public class AppelSimple {
 public static void f() {
   System.out.println("Entree f()");
                                                                // 100
   g();
   System.out.println("Sortie f()");
 public static void g() {
   System.out.println("Je suis g()");
 public static void main(String[] args) {
   System.out.println("Entree main");
   f():
             // 1 empiler le bloc d'activation et saut inconditionnel
                    // 2 point de retour : dépiler le bloc d'activation
   System.out.println("Sortie main");
                                                                   1/3
                                                   // 4 sortie de main
```

```
public class AppelSimple {
 public static void f() {
   System.out.println("Entree f()");
                                                                // 100
   g(): // 101 empiler le bloc d'activation et saut inconditionnel
                   // 102 point de retour: dépiler le bloc d'activation
   System.out.println("Sortie f()");
 public static void g() {
   System.out.println("Je suis g()");
 public static void main(String[] args) {
   System.out.println("Entree main");
                                                                  110
   f(); // 1 empiler le bloc d'activation et saut inconditionnel
                    // 2 point de retour : dépiler le bloc d'activation
   System.out.println("Sortie main");
                                                                  1/3
                                                   // L sortie de main
```

```
public class AppelSimple {
 public static void f() {
   System.out.println("Entree f()");
                                                                // 100
   g(): // 101 empiler le bloc d'activation et saut inconditionnel
                   // 102 point de retour: dépiler le bloc d'activation
   System.out.println("Sortie f()");
                                                                // 103
                   // 104 sortie de f (on récupère l'adresse de retour)
 public static void g() {
   System.out.println("Je suis g()");
 public static void main(String[] args) {
   System.out.println("Entree main");
                                                                  110
   f(); // 1 empiler le bloc d'activation et saut inconditionnel
                    // 2 point de retour : dépiler le bloc d'activation
   System.out.println("Sortie main");
                                                                  1/3
                                                   // L sortie de main
```

```
public class AppelSimple {
 public static void f() {
   System.out.println("Entree f()");
                                                                // 100
   g(): // 101 empiler le bloc d'activation et saut inconditionnel
                   // 102 point de retour: dépiler le bloc d'activation
   System.out.println("Sortie f()");
                                                                // 103
                   // 104 sortie de f (on récupère l'adresse de retour)
 public static void g() {
   System.out.println("Je suis g()");
                                                                11 200
                  // 201 sortie de q (on récupère l'adresse de retour)
 public static void main(String[] args) {
   System.out.println("Entree main");
                                                                  110
   f(); // 1 empiler le bloc d'activation et saut inconditionnel
                    // 2 point de retour : dépiler le bloc d'activation
   System.out.println("Sortie main");
                                                                  1/3
                                                   // L sortie de main
```

```
public class AppelSimple {
                                               import java.util.Stack;
 public static void f() {
                                               public class AppelSimpleTrad {
   System.out.println("Entree f()"); //100
                                                 static int ic = 0;
   g(); // 101 empiler bloc + saut incond.
                                                 static Stack<Bloc> p = new Stack<Bloc>();
         // 102 point de retour:
                                                 public static void main(String[] args) {
          // dépiler le bloc d'activation
                                                  while(true) {
   System.out.println("Sortie f()"); //103
                                                   switch(ic) {
                         //104 sortie de f
                                                   case 0:
                                                   case 1:
 public static void g() {
                                                   case 2:
   System.out.println("Je suis g()")://200
                                                   case 3:
                          //201 sortie de q
                                                   case 4:
 public static void main(String[] args) {
                                                   case 100:
   System.out.println("Entree main"); //0
                                                   case 101:
   f(): //1 empiler bloc + saut incond.
                                                   case 102:
          //2 point de retour :
                                                   case 103:
          // dépiler le bloc d'activation
                                                   case 104:
   System.out.println("Sortie main"); //3
                       //4 sortie de main
                                                   case 200:
                                                   case 201:
                                               1111
```

```
public class AppelSimple {
                                               import java.util.Stack;
 public static void f() {
                                               public class AppelSimpleTrad {
   System.out.println("Entree f()"); //100
                                                 static int ic = 0;
   g(); // 101 empiler bloc + saut incond.
                                                 static Stack<Bloc> p = new Stack<Bloc>();
         // 102 point de retour:
                                                 public static void main(String[] args) {
          // dépiler le bloc d'activation
                                                  while(true) {
   System.out.println("Sortie f()"); //103
                                                   switch(ic) {
                         //104 sortie de f
                                                   case 0: ...println("Entree main"); ic++; break;
                                                   case 1:
 public static void g() {
                                                   case 2:
   System.out.println("Je suis g()")://200
                                                   case 3:
                          //201 sortie de q
                                                   case 4:
 public static void main(String[] args) {
                                                   case 100:
   System.out.println("Entree main"); //0
                                                   case 101:
   f(): //1 empiler bloc + saut incond.
                                                   case 102:
          //2 point de retour :
                                                   case 103:
          // dépiler le bloc d'activation
                                                   case 104:
   System.out.println("Sortie main"); //3
                       //4 sortie de main
                                                   case 200:
                                                   case 201:
                                               1111
```

```
public class AppelSimple {
                                               import java.util.Stack;
 public static void f() {
                                               public class AppelSimpleTrad {
   System.out.println("Entree f()"); //100
                                                 static int ic = 0;
   g(); // 101 empiler bloc + saut incond.
                                                 static Stack<Bloc> p = new Stack<Bloc>();
         // 102 point de retour:
                                                 public static void main(String[] args) {
          // dépiler le bloc d'activation
                                                  while(true) {
   System.out.println("Sortie f()"); //103
                                                   switch(ic) {
                         //104 sortie de f
                                                   case 0: ...println("Entree main"); ic++; break;
                                                   case 1: p.push(new Bloc(ic+1)); ic = 100; break;
 public static void g() {
                                                   case 2:
   System.out.println("Je suis g()")://200
                                                   case 3:
                          //201 sortie de q
                                                   case 4:
 public static void main(String[] args) {
                                                   case 100:
   System.out.println("Entree main"); //0
                                                   case 101:
   f(): //1 empiler bloc + saut incond.
                                                   case 102:
          //2 point de retour :
                                                   case 103:
          // dépiler le bloc d'activation
                                                   case 104:
   System.out.println("Sortie main"); //3
                       //4 sortie de main
                                                   case 200:
                                                   case 201:
                                               1111
```

```
public class AppelSimple {
                                               import java.util.Stack;
 public static void f() {
                                               public class AppelSimpleTrad {
   System.out.println("Entree f()"); //100
                                                 static int ic = 0;
   g(); // 101 empiler bloc + saut incond.
                                                 static Stack<Bloc> p = new Stack<Bloc>();
         // 102 point de retour:
                                                 public static void main(String[] args) {
          // dépiler le bloc d'activation
                                                  while(true) {
   System.out.println("Sortie f()"); //103
                                                   switch(ic) {
                         //104 sortie de f
                                                   case 0: ...println("Entree main"); ic++; break;
                                                   case 1: p.push(new Bloc(ic+1)); ic = 100; break;
 public static void g() {
                                                   case 2: p.pop(); ic++; break;
   System.out.println("Je suis g()")://200
                                                   case 3:
                         //201 sortie de q
                                                   case 4:
 public static void main(String[] args) {
                                                   case 100:
   System.out.println("Entree main"); //0
                                                   case 101:
   f(): //1 empiler bloc + saut incond.
                                                   case 102:
          //2 point de retour :
                                                   case 103:
          // dépiler le bloc d'activation
                                                   case 104:
   System.out.println("Sortie main"); //3
                       //4 sortie de main
                                                   case 200:
                                                   case 201:
                                               1111
```

```
public class AppelSimple {
                                               import java.util.Stack;
 public static void f() {
                                               public class AppelSimpleTrad {
   System.out.println("Entree f()"); //100
                                                 static int ic = 0;
   g(); // 101 empiler bloc + saut incond.
                                                 static Stack<Bloc> p = new Stack<Bloc>();
         // 102 point de retour:
                                                 public static void main(String[] args) {
          // dépiler le bloc d'activation
                                                  while(true) {
   System.out.println("Sortie f()"); //103
                                                   switch(ic) {
                         //104 sortie de f
                                                   case 0: ...println("Entree main"); ic++; break;
                                                   case 1: p.push(new Bloc(ic+1)); ic = 100; break;
 public static void g() {
                                                   case 2: p.pop(); ic++; break;
   System.out.println("Je suis g()")://200
                                                   case 3: ...println("Sortie main"); ic++; break;
                         //201 sortie de q
                                                   case 4:
 public static void main(String[] args) {
                                                   case 100:
   System.out.println("Entree main"); //0
                                                   case 101:
   f(): //1 empiler bloc + saut incond.
                                                   case 102:
          //2 point de retour :
                                                   case 103:
          // dépiler le bloc d'activation
                                                   case 104:
   System.out.println("Sortie main"); //3
                       //4 sortie de main
                                                   case 200:
                                                   case 201:
                                               1111
```

```
public class AppelSimple {
                                               import java.util.Stack;
 public static void f() {
                                               public class AppelSimpleTrad {
   System.out.println("Entree f()"); //100
                                                 static int ic = 0;
   g(); // 101 empiler bloc + saut incond.
                                                 static Stack<Bloc> p = new Stack<Bloc>():
         // 102 point de retour:
                                                 public static void main(String[] args) {
          // dépiler le bloc d'activation
                                                  while(true) {
   System.out.println("Sortie f()"); //103
                                                   switch(ic) {
                         //104 sortie de f
                                                   case 0: ...println("Entree main"); ic++; break;
                                                   case 1: p.push(new Bloc(ic+1)); ic = 100; break;
 public static void g() {
                                                   case 2: p.pop(); ic++; break;
   System.out.println("Je suis g()")://200
                                                   case 3: ...println("Sortie main"); ic++; break;
                         //201 sortie de q
                                                   case 4: System.exit(0);
 public static void main(String[] args) {
                                                   case 100:
   System.out.println("Entree main"); //0
                                                   case 101:
   f(): //1 empiler bloc + saut incond.
                                                   case 102:
          //2 point de retour :
                                                   case 103:
          // dépiler le bloc d'activation
                                                   case 104:
   System.out.println("Sortie main"); //3
                       //4 sortie de main
                                                   case 200:
                                                   case 201:
                                               1 1 1 1
```

```
public class AppelSimple {
                                               import java.util.Stack;
 public static void f() {
                                               public class AppelSimpleTrad {
   System.out.println("Entree f()"); //100
                                                 static int ic = 0;
   g(); // 101 empiler bloc + saut incond.
                                                 static Stack<Bloc> p = new Stack<Bloc>();
         // 102 point de retour:
                                                 public static void main(String[] args) {
          // dépiler le bloc d'activation
                                                  while(true) {
   System.out.println("Sortie f()"); //103
                                                   switch(ic) {
                         //104 sortie de f
                                                   case 0: ...println("Entree main"); ic++; break;
                                                   case 1: p.push(new Bloc(ic+1)); ic = 100; break;
 public static void g() {
                                                   case 2: p.pop(); ic++; break;
   System.out.println("Je suis g()")://200
                                                   case 3: ...println("Sortie main"); ic++; break;
                         //201 sortie de q
                                                   case 4: System.exit(0);
 public static void main(String[] args) {
                                                   case 100: ...println("Entree f()"); ic++; break;
   System.out.println("Entree main"); //0
                                                   case 101:
   f(): //1 empiler bloc + saut incond.
                                                   case 102:
          //2 point de retour :
                                                   case 103:
          // dépiler le bloc d'activation
                                                   case 104:
   System.out.println("Sortie main"); //3
                       //4 sortie de main
                                                   case 200:
                                                   case 201:
                                               1 1 1 1
```

```
public class AppelSimple {
                                               import java.util.Stack;
 public static void f() {
                                               public class AppelSimpleTrad {
   System.out.println("Entree f()"); //100
                                                 static int ic = 0;
   g(); // 101 empiler bloc + saut incond.
                                                 static Stack<Bloc> p = new Stack<Bloc>();
         // 102 point de retour:
                                                 public static void main(String[] args) {
          // dépiler le bloc d'activation
                                                  while(true) {
   System.out.println("Sortie f()"); //103
                                                   switch(ic) {
                         //104 sortie de f
                                                   case 0: ...println("Entree main"); ic++; break;
                                                   case 1: p.push(new Bloc(ic+1)); ic = 100; break;
 public static void g() {
                                                   case 2: p.pop(); ic++; break;
   System.out.println("Je suis g()")://200
                                                   case 3: ...println("Sortie main"); ic++; break;
                         //201 sortie de q
                                                   case 4: System.exit(0);
 public static void main(String[] args) {
                                                   case 100: ...println("Entree f()"); ic++; break;
   System.out.println("Entree main"); //0
                                                   case 101: p.push(new Bloc(ic+1)); ic = 200; break;
   f(): //1 empiler bloc + saut incond.
                                                   case 102:
          //2 point de retour :
                                                   case 103:
          // dépiler le bloc d'activation
                                                   case 104:
   System.out.println("Sortie main"); //3
                       //4 sortie de main
                                                   case 200:
                                                   case 201:
                                               1 1 1 1
```

```
public class AppelSimple {
                                               import java.util.Stack;
 public static void f() {
                                               public class AppelSimpleTrad {
   System.out.println("Entree f()"); //100
                                                 static int ic = 0;
   g(); // 101 empiler bloc + saut incond.
                                                 static Stack<Bloc> p = new Stack<Bloc>();
         // 102 point de retour:
                                                 public static void main(String[] args) {
          // dépiler le bloc d'activation
                                                  while(true) {
   System.out.println("Sortie f()"); //103
                                                   switch(ic) {
                         //104 sortie de f
                                                   case 0: ...println("Entree main"); ic++; break;
                                                   case 1: p.push(new Bloc(ic+1)); ic = 100; break;
 public static void g() {
                                                   case 2: p.pop(); ic++; break;
   System.out.println("Je suis g()")://200
                                                   case 3: ...println("Sortie main"); ic++; break;
                         //201 sortie de q
                                                   case 4: System.exit(0);
 public static void main(String[] args) {
                                                   case 100: ...println("Entree f()"); ic++; break;
   System.out.println("Entree main"); //0
                                                   case 101: p.push(new Bloc(ic+1)); ic = 200; break;
   f(): //1 empiler bloc + saut incond.
                                                   case 102: p.pop(); ic++; break:
          //2 point de retour :
                                                   case 103:
          // dépiler le bloc d'activation
                                                   case 104:
   System.out.println("Sortie main"); //3
                       //4 sortie de main
                                                   case 200:
                                                   case 201:
                                               1 1 1 1
```

```
public class AppelSimple {
                                               import java.util.Stack;
 public static void f() {
                                               public class AppelSimpleTrad {
   System.out.println("Entree f()"); //100
                                                 static int ic = 0;
   g(); // 101 empiler bloc + saut incond.
                                                 static Stack<Bloc> p = new Stack<Bloc>();
         // 102 point de retour:
                                                 public static void main(String[] args) {
          // dépiler le bloc d'activation
                                                  while(true) {
   System.out.println("Sortie f()"); //103
                                                   switch(ic) {
                         //104 sortie de f
                                                   case 0: ...println("Entree main"); ic++; break;
                                                   case 1: p.push(new Bloc(ic+1)); ic = 100; break;
 public static void g() {
                                                   case 2: p.pop(); ic++; break;
   System.out.println("Je suis g()")://200
                                                   case 3: ...println("Sortie main"); ic++; break;
                         //201 sortie de q
                                                   case 4: System.exit(0);
 public static void main(String[] args) {
                                                   case 100: ...println("Entree f()"); ic++; break;
   System.out.println("Entree main"); //0
                                                   case 101: p.push(new Bloc(ic+1)); ic = 200; break;
   f(): //1 empiler bloc + saut incond.
                                                   case 102: p.pop(); ic++; break:
          //2 point de retour :
                                                   case 103:
          // dépiler le bloc d'activation
                                                   case 104:
   System.out.println("Sortie main"); //3
                       //4 sortie de main
                                                   case 200:
                                                   case 201:
                                               1 1 1 1
```

```
public class AppelSimple {
                                               import java.util.Stack;
 public static void f() {
                                               public class AppelSimpleTrad {
   System.out.println("Entree f()"); //100
                                                 static int ic = 0;
   g(); // 101 empiler bloc + saut incond.
                                                 static Stack<Bloc> p = new Stack<Bloc>();
         // 102 point de retour:
                                                 public static void main(String[] args) {
          // dépiler le bloc d'activation
                                                  while(true) {
   System.out.println("Sortie f()"); //103
                                                   switch(ic) {
                         //104 sortie de f
                                                   case 0: ...println("Entree main"); ic++; break;
                                                   case 1: p.push(new Bloc(ic+1)); ic = 100; break;
 public static void g() {
                                                   case 2: p.pop(); ic++; break;
   System.out.println("Je suis g()")://200
                                                   case 3: ...println("Sortie main"); ic++; break;
                         //201 sortie de q
                                                   case 4: System.exit(0);
 public static void main(String[] args) {
                                                   case 100: ...println("Entree f()"); ic++; break;
   System.out.println("Entree main"); //0
                                                   case 101: p.push(new Bloc(ic+1)); ic = 200; break;
                                                   case 102: p.pop(); ic++; break;
   f(): //1 empiler bloc + saut incond.
          //2 point de retour :
                                                   case 103: ...println("Sortie f()"); ic++; break;
          // dépiler le bloc d'activation
                                                   case 104:
   System.out.println("Sortie main"); //3
                       //4 sortie de main
                                                   case 200:
                                                   case 201:
                                               1 1 1 1
```

```
public class AppelSimple {
                                               import java.util.Stack;
 public static void f() {
                                               public class AppelSimpleTrad {
   System.out.println("Entree f()"); //100
                                                 static int ic = 0;
   g(); // 101 empiler bloc + saut incond.
                                                 static Stack<Bloc> p = new Stack<Bloc>();
         // 102 point de retour:
                                                 public static void main(String[] args) {
          // dépiler le bloc d'activation
                                                  while(true) {
   System.out.println("Sortie f()"); //103
                                                   switch(ic) {
                         //104 sortie de f
                                                   case 0: ...println("Entree main"); ic++; break;
                                                   case 1: p.push(new Bloc(ic+1)); ic = 100; break;
 public static void g() {
                                                   case 2: p.pop(); ic++; break;
   System.out.println("Je suis g()")://200
                                                   case 3: ...println("Sortie main"); ic++; break;
                         //201 sortie de q
                                                   case 4: System.exit(0);
 public static void main(String[] args) {
                                                   case 100: ...println("Entree f()"); ic++; break;
   System.out.println("Entree main"); //0
                                                   case 101: p.push(new Bloc(ic+1)); ic = 200; break;
                                                   case 102: p.pop(); ic++; break;
   f(): //1 empiler bloc + saut incond.
          //2 point de retour :
                                                   case 103: ...println("Sortie f()"); ic++; break;
          // dépiler le bloc d'activation
                                                   case 104: ic = p.peek().getAdresse(): break:
   System.out.println("Sortie main"); //3
                       //4 sortie de main
                                                   case 200:
                                                   case 201:
                                               1 1 1 1
```

```
public class AppelSimple {
                                               import java.util.Stack;
 public static void f() {
                                               public class AppelSimpleTrad {
   System.out.println("Entree f()"); //100
                                                 static int ic = 0;
   g(); // 101 empiler bloc + saut incond.
                                                 static Stack<Bloc> p = new Stack<Bloc>();
         // 102 point de retour:
                                                 public static void main(String[] args) {
          // dépiler le bloc d'activation
                                                  while(true) {
   System.out.println("Sortie f()"); //103
                                                   switch(ic) {
                         //104 sortie de f
                                                   case 0: ...println("Entree main"); ic++; break;
                                                   case 1: p.push(new Bloc(ic+1)); ic = 100; break;
 public static void g() {
                                                   case 2: p.pop(); ic++; break;
   System.out.println("Je suis g()")://200
                                                   case 3: ...println("Sortie main"); ic++; break;
                         //201 sortie de q
                                                   case 4: System.exit(0);
 public static void main(String[] args) {
                                                   case 100: ...println("Entree f()"); ic++; break;
   System.out.println("Entree main"); //0
                                                   case 101: p.push(new Bloc(ic+1)); ic = 200; break;
                                                   case 102: p.pop(); ic++; break;
   f(): //1 empiler bloc + saut incond.
          //2 point de retour :
                                                   case 103: ...println("Sortie f()"); ic++; break;
                                                   case 104: ic = p.peek().getAdresse(); break;
          // dépiler le bloc d'activation
   System.out.println("Sortie main"); //3
                       //4 sortie de main
                                                   case 200: ...println("Je suis g()"); ic++; break;
                                                   case 201:
                                               1 1 1 1
```

```
public class AppelSimple {
                                               import java.util.Stack;
 public static void f() {
                                               public class AppelSimpleTrad {
   System.out.println("Entree f()"); //100
                                                 static int ic = 0;
   g(); // 101 empiler bloc + saut incond.
                                                 static Stack<Bloc> p = new Stack<Bloc>();
         // 102 point de retour:
                                                 public static void main(String[] args) {
          // dépiler le bloc d'activation
                                                  while(true) {
   System.out.println("Sortie f()"); //103
                                                   switch(ic) {
                         //104 sortie de f
                                                   case 0: ...println("Entree main"); ic++; break;
                                                   case 1: p.push(new Bloc(ic+1)); ic = 100; break;
 public static void g() {
                                                   case 2: p.pop(); ic++; break;
   System.out.println("Je suis g()")://200
                                                   case 3: ...println("Sortie main"); ic++; break;
                         //201 sortie de q
                                                   case 4: System.exit(0);
 public static void main(String[] args) {
                                                   case 100: ...println("Entree f()"); ic++; break;
   System.out.println("Entree main"); //0
                                                   case 101: p.push(new Bloc(ic+1)); ic = 200; break;
                                                   case 102: p.pop(); ic++; break;
   f(): //1 empiler bloc + saut incond.
          //2 point de retour :
                                                   case 103: ...println("Sortie f()"); ic++; break;
                                                   case 104: ic = p.peek().getAdresse(); break;
          // dépiler le bloc d'activation
   System.out.println("Sortie main"); //3
                       //4 sortie de main
                                                   case 200: ...println("Je suis g()"); ic++; break;
                                                   case 201: ic = p.peek().getAdresse(); break;
                                               1 1 1 1
```

```
public class AppelSimpleDeux {
 static int k = 1;
 public static void g() {
   System.out.println(k+ " éléphants se balançaient\n"
        + "Sur une toile, toile, toile\n"
       + "Toile d'araignée");
   k++:
 public static void main(String[] args) {
   while (k \le 10) {
     g();
   System.out.println("C'était un jeu tellement\n"
       + "Tellement amusant\n"
        + "Que tout à coup\n"
       + "Ba da boum"):
 } }
```

```
public class AppelSimpleDeux {
 static int k = 1; // 0: init mem\lceil 0 \rceil
 public static void g() {
   System.out.println(k+ " éléphants se balançaient\n"
        + "Sur une toile, toile, toile\n"
       + "Toile d'araignée");
   k++:
 public static void main(String[] args) {
   while (k \le 10)
     g();
   System.out.println("C'était un jeu tellement\n"
       + "Tellement amusant\n"
       + "Que tout à coup\n"
       + "Ba da boum"):
 } }
```

```
public class AppelSimpleDeux {
  static int k = 1; // 0: init mem\lceil 0 \rceil
 public static void g() {
    System.out.println(k+ " éléphants se balançaient\n"
        + "Sur une toile, toile, toile\n"
        + "Toile d'araignée");
   k++:
 public static void main(String[] args) {
   while (k <= 10 ) {
     // 1: saut conditionnel</pre>
     g();
    System.out.println("C'était un jeu tellement\n"
        + "Tellement amusant\n"
        + "Que tout à coup\n"
        + "Ba da boum"):
 } }
```

```
public class AppelSimpleDeux {
 public static void g() {
   System.out.println(k+ " éléphants se balançaient\n"
       + "Sur une toile, toile, toile\n"
       + "Toile d'araignée");
   k++:
 public static void main(String[] args) {
   while (k <= 10 ) {
     // 1: saut conditionnel</pre>
     g();
                        // 2: appel à q
                            // 3: point de retour après l'appel
   System.out.println("C'était un jeu tellement\n"
       + "Tellement amusant\n"
       + "Que tout à coup\n"
       + "Ba da boum"):
```

```
public class AppelSimpleDeux {
 public static void g() {
   System.out.println(k+ " éléphants se balançaient\n"
       + "Sur une toile, toile, toile\n"
       + "Toile d'araignée");
   k++:
 public static void main(String[] args) {
   while (k <= 10 ) {
     // 1: saut conditionnel</pre>
     g();
                         // 2: appel à q
                            // 3: point de retour après l'appel
                            // 4: saut inconditionnel
   System.out.println("C'était un jeu tellement\n"
       + "Tellement amusant\n"
       + "Que tout à coup\n"
       + "Ba da boum"):
```

```
public class AppelSimpleDeux {
 public static void g() {
   System.out.println(k+ " éléphants se balançaient\n"
      + "Sur une toile, toile, toile\n"
      + "Toile d'araignée");
   k++:
 public static void main(String[] args) {
   while (k <= 10 ) {
     // 1: saut conditionnel</pre>
    g();
                       // 2: appel à q
                          // 3: point de retour après l'appel
                          // 4: saut inconditionnel
   System.out.println("C'était un jeu tellement\n"
      + "Tellement amusant\n"
      + "Que tout à coup\n"
      + "Ba da boum"); // 5: affichage
```

```
public class AppelSimpleDeux {
 public static void g() {
   System.out.println(k+ " éléphants se balançaient\n"
      + "Sur une toile, toile, toile\n"
      + "Toile d'araignée"); // 100: affichage
   k++:
 public static void main(String[] args) {
   while (k <= 10 ) {
     // 1: saut conditionnel</pre>
    g();
                       // 2: appel à q
                          // 3: point de retour après l'appel
                          // 4: saut inconditionnel
   System.out.println("C'était un jeu tellement\n"
      + "Tellement amusant\n"
      + "Que tout à coup\n"
      + "Ba da boum"); // 5: affichage
```

```
public class AppelSimpleDeux {
 public static void g() {
   System.out.println(k+ " éléphants se balançaient\n"
      + "Sur une toile, toile, toile\n"
      + "Toile d'araignée"); // 100: affichage
                          // 101: modif mem[0]
   k++:
 public static void main(String[] args) {
   while (k <= 10 ) {
     // 1: saut conditionnel</pre>
    g();
                       // 2: appel à q
                          // 3: point de retour après l'appel
                          // 4: saut inconditionnel
   System.out.println("C'était un jeu tellement\n"
      + "Tellement amusant\n"
      + "Que tout à coup\n"
      + "Ba da boum"); // 5: affichage
```

```
public class AppelSimpleDeux {
 public static void g() {
   System.out.println(k+ " éléphants se balançaient\n"
      + "Sur une toile, toile, toile\n"
      + "Toile d'araignée"); // 100: affichage
                         // 101: modif mem[0]
   k++:
                          // 102: sortie de q
 public static void main(String[] args) {
   while (k <= 10 ) {
     // 1: saut conditionnel</pre>
    g();
                       // 2: appel à q
                          // 3: point de retour après l'appel
                          // 4: saut inconditionnel
   System.out.println("C'était un jeu tellement\n"
      + "Tellement amusant\n"
      + "Que tout à coup\n"
      + "Ba da boum"); // 5: affichage
```

```
public class AppelSimpleDeux {
                                             import java.util.Stack;
 static int k = 1;  // 0: init mem[0]
                                             public class AppelSimpleDeuxTrad {
                                               static int ic = 0; int[] mem = new int[1000];
 public static void g() {
                                               static Stack<Bloc> p = new Stack<Bloc>();
   System.out.println(k+ " éléphants ...
                                              public static void main(String[] args) {

→ d'araignée"): // 100

                                               while(true) {
   k++; // 101: modif mem[0]
                                                switch(ic) {
    // 102: sortie q
                                                 case 0: mem[0] = 1; ic++; break; // init. de k
                                                case 1:
 public static void main(String[] args) {
   while (k \le 10)  \{ // 1: saut cond. \}
                                                case 2:
     g(): // 2: appel à q
              // 3: point de retour
                                                case 3:
   } // 4: saut incond.
                                                case 4:
   System.out.println("C'était un jeu
                                                case 5:
    → tellement\n"
                                                case 6:
       + "Tellement amusant\n"
                                                case 100:
       + "Que tout à coup\n"
                                                case 101:
       + "Ba da boum"); // 5: affichage
                                                case 102:
 } }
      // 6: sortie
                                             1111
```

```
public class AppelSimpleDeux {
                                            import java.util.Stack;
 static int k = 1;  // 0: init mem[0]
                                            public class AppelSimpleDeuxTrad {
                                              static int ic = 0; int[] mem = new int[1000];
 public static void g() {
                                              static Stack<Bloc> p = new Stack<Bloc>();
   System.out.println(k+ " éléphants ...
                                              public static void main(String[] args) {

→ d'araignée"): // 100

                                               while(true) {
   k++; // 101: modif mem[0]
                                                switch(ic) {
    // 102: sortie q
                                                case 0: mem[0] = 1; ic++; break; // init. de k
                                                case 1: if(mem[0] > 10){ic += 4;} else {ic++;};
 public static void main(String[] args) {
                                                       break: //saut cond.
   while (k \le 10)  \{ // 1: saut cond. \}
                                                case 2:
     g(): // 2: appel à q
              // 3: point de retour
                                                case 3:
   } // 4: saut incond.
                                                case 4:
   System.out.println("C'était un jeu
                                                case 5:
    → tellement\n"
                                                case 6:
       + "Tellement amusant\n"
                                                case 100:
       + "Que tout à coup\n"
                                                case 101:
       + "Ba da boum"); // 5: affichage
                                                case 102:
 } }
      // 6: sortie
                                            1111
```

```
public class AppelSimpleDeux {
                                            import java.util.Stack;
 static int k = 1;  // 0: init mem[0]
 public static void g() {
   System.out.println(k+ " éléphants ...

→ d'araignée"): // 100

                                               while(true) {
                                                switch(ic) {
   k++: // 101: modif mem[0]
    // 102: sortie g
 public static void main(String[] args) {
   while (k \le 10)  \{ // 1: saut cond. \}
     g(): // 2: appel à q
              // 3: point de retour
                                                case 3:
   } // 4: saut incond.
                                                case 4:
   System.out.println("C'était un jeu
                                                case 5:
    → tellement\n"
                                                case 6:
       + "Tellement amusant\n"
                                                case 100:
       + "Que tout à coup\n"
                                                case 101:
       + "Ba da boum"); // 5: affichage
                                                case 102:
 } }
      // 6: sortie
                                            1 1 1 1
```

```
public class AppelSimpleDeuxTrad {
  static int ic = 0; int[] mem = new int[1000];
  static Stack<Bloc> p = new Stack<Bloc>();
 public static void main(String[] args) {
   case 0: mem[0] = 1; ic++; break; // init. de k
   case 1: if(mem[0] > 10){ic += 4;} else {ic++;};
           break;
                              //saut. cond.
   case 2: p.push(new Bloc(ic+1)); ic = 100; break;
           //empiler le bloc d'act. + saut incond.
```

```
public class AppelSimpleDeux {
 static int k = 1;  // 0: init mem[0]
 public static void g() {
   System.out.println(k+ " éléphants ...

→ d'araignée"); // 100

  k++: // 101: modif mem[0]
     // 102: sortie g
 public static void main(String[] args) {
   while (k \le 10)  \{ // 1: saut cond. \}
     g(): // 2: appel à q
              // 3: point de retour
   } // 4: saut incond.
   System.out.println("C'était un jeu
    → tellement\n"
       + "Tellement amusant\n"
       + "Que tout à coup\n"
       + "Ba da boum"); // 5: affichage
 } }
      // 6: sortie
                                           1 1 1 1
```

```
import java.util.Stack;
public class AppelSimpleDeuxTrad {
  static int ic = 0; int[] mem = new int[1000];
  static Stack<Bloc> p = new Stack<Bloc>();
 public static void main(String[] args) {
  while(true) {
   switch(ic) {
    case 0: mem[0] = 1; ic++; break; // init. de k
    case 1: if(mem[0] > 10){ic += 4;} else {ic++;};
           break;
                               //saut. cond.
    case 2: p.push(new Bloc(ic+1)); ic = 100; break;
           //empiler le bloc d'act. + saut incond.
    case 3: p.pop(); ic++; break;// dépiler le bloc
    case 4:
   case 5:
   case 6:
    case 100:
   case 101:
   case 102:
```

```
public class AppelSimpleDeux {
 static int k = 1;  // 0: init mem[0]
 public static void g() {
   System.out.println(k+ " éléphants ...

→ d'araignée"); // 100

  k++; // 101: modif mem[0]
    // 102: sortie g
 public static void main(String[] args) {
   while (k \le 10)  \{ // 1: saut cond. \}
     g(): // 2: appel à q
              // 3: point de retour
   } // 4: saut incond.
   System.out.println("C'était un jeu
    → tellement\n"
       + "Tellement amusant\n"
       + "Que tout à coup\n"
       + "Ba da boum"); // 5: affichage
 } }
      // 6: sortie
                                           1111
```

```
import java.util.Stack;
public class AppelSimpleDeuxTrad {
  static int ic = 0; int[] mem = new int[1000];
  static Stack<Bloc> p = new Stack<Bloc>();
 public static void main(String[] args) {
  while(true) {
   switch(ic) {
   case 0: mem[0] = 1; ic++; break; // init. de k
   case 1: if(mem[0] > 10){ic += 4;} else {ic++;};
           break: //saut cond.
   case 2: p.push(new Bloc(ic+1)); ic = 100; break;
           //empiler le bloc d'act. + saut incond.
   case 3: p.pop(); ic++; break;// dépiler le bloc
   case 4: ic -= 3; break: //saut incond.
   case 5:
   case 6:
   case 100:
   case 101:
   case 102:
```

```
public class AppelSimpleDeux {
 static int k = 1;  // 0: init mem[0]
 public static void g() {
   System.out.println(k+ " éléphants ...

→ d'araignée"); // 100

                                              while(true) {
   k++; // 101: modif mem[0]
                                               switch(ic) {
    // 102: sortie g
 public static void main(String[] args) {
   while (k \le 10)  \{ // 1: saut cond. \}
     g(): // 2: appel à q
              // 3: point de retour
   } // 4: saut incond.
   System.out.println("C'était un jeu
    → tellement\n"
                                               case 6:
       + "Tellement amusant\n"
                                               case 100:
       + "Que tout à coup\n"
                                               case 101:
       + "Ba da boum"); // 5: affichage
                                               case 102:
 } }
      // 6: sortie
                                            1111
```

```
import java.util.Stack;
public class AppelSimpleDeuxTrad {
  static int ic = 0; int[] mem = new int[1000];
  static Stack<Bloc> p = new Stack<Bloc>();
 public static void main(String[] args) {
   case 0: mem[0] = 1; ic++; break; // init. de k
   case 1: if(mem[0] > 10){ic += 4;} else {ic++;};
           break: //saut cond.
   case 2: p.push(new Bloc(ic+1)); ic = 100; break;
           //empiler le bloc d'act. + saut incond.
   case 3: p.pop(); ic++; break;// dépiler le bloc
   case 4: ic -= 3: break: //saut incond.
   case 5: S.out.println("C'était ..."); ic++; break;
```

```
public class AppelSimpleDeux {
 static int k = 1;  // 0: init mem[0]
 public static void g() {
   System.out.println(k+ " éléphants ...

→ d'araignée"); // 100

                                               while(true) {
   k++; // 101: modif mem[0]
                                                switch(ic) {
     // 102: sortie g
 public static void main(String[] args) {
   while (k \le 10)  \{ // 1: saut cond. \}
     g(): // 2: appel à q
              // 3: point de retour
   } // 4: saut incond.
   System.out.println("C'était un jeu
    → tellement\n"
       + "Tellement amusant\n"
                                                case 100:
       + "Que tout à coup\n"
                                                case 101:
       + "Ba da boum"); // 5: affichage
                                                case 102:
 } }
       // 6: sortie
                                            1 1 1 1
```

```
import java.util.Stack;
public class AppelSimpleDeuxTrad {
  static int ic = 0; int[] mem = new int[1000];
  static Stack<Bloc> p = new Stack<Bloc>();
 public static void main(String[] args) {
   case 0: mem[0] = 1; ic++; break; // init. de k
   case 1: if(mem[0] > 10){ic += 4;} else {ic++;};
           break: //saut cond.
   case 2: p.push(new Bloc(ic+1)); ic = 100; break;
           //empiler le bloc d'act. + saut incond.
   case 3: p.pop(); ic++; break;// dépiler le bloc
   case 4: ic -= 3: break: //saut incond.
   case 5: S.out.println("C'était ..."); ic++; break;
   case 6: System.exit(0);
```

```
public class AppelSimpleDeux {
                                            import java.util.Stack;
 static int k = 1: // 0: init mem [0]
                                            public class AppelSimpleDeuxTrad {
                                              static int ic = 0; int[] mem = new int[1000];
 public static void g() {
                                              static Stack<Bloc> p = new Stack<Bloc>():
                                              public static void main(String[] args) {
   System.out.println(k+ " éléphants ...

    d'araignée"); // 100

                                               while(true) {
   k++; // 101: modif mem[0]
                                                switch(ic) {
        // 102: sortie q
                                                case 0: mem[0] = 1; ic++; break; // init. de k
                                                case 1: if(mem[0] > 10){ic += 4;} else {ic++;};
 public static void main(String[] args) {
                                                       break: //saut cond.
   while (k \le 10) \{ // 1: saut cond. \}
                                                case 2: p.push(new Bloc(ic+1)); ic = 100; break;
     g(); // 2: appel à q
                                                       //empiler le bloc d'act. + saut incond.
              // 3: point de retour
                                                case 3: p.pop(); ic++; break:// dépiler le bloc
   } // 4: saut incond.
                                                case 4: ic -= 3; break; //saut incond.
   System.out.println("C'était un jeu
                                                case 5: S.out.println("C'était ..."); ic++; break;
    → tellement\n"
                                                case 6: System.exit(0);
       + "Tellement amusant\n"
                                                case 100: S.o.println(mem[0]+"éléphants...");
       + "Que tout à coup\n"
                                                → ic++: break:
       + "Ba da boum"); // 5: affichage
                                                case 101:
 case 102:
                                            1 1 1 1
```

```
public class AppelSimpleDeux {
                                            import java.util.Stack;
                                            public class AppelSimpleDeuxTrad {
 static int k = 1: // 0: init mem [0]
                                              static int ic = 0; int[] mem = new int[1000];
 public static void g() {
                                              static Stack<Bloc> p = new Stack<Bloc>():
                                              public static void main(String[] args) {
   System.out.println(k+ " éléphants ...

    d'araignée"); // 100

                                               while(true) {
   k++; // 101: modif mem[0]
                                                switch(ic) {
        // 102: sortie q
                                                case 0: mem[0] = 1: ic++: break: // init. de k
                                                case 1: if(mem[0] > 10){ic += 4;} else {ic++;};
 public static void main(String[] args) {
                                                       break: //saut cond.
   while (k \le 10) \{ // 1: saut cond. \}
                                                case 2: p.push(new Bloc(ic+1)); ic = 100; break;
     g(); // 2: appel à q
                                                       //empiler le bloc d'act. + saut incond.
              // 3: point de retour
                                                case 3: p.pop(); ic++; break:// dépiler le bloc
   } // 4: saut incond.
                                                case 4: ic -= 3; break; //saut incond.
   System.out.println("C'était un jeu
                                                case 5: S.out.println("C'était ..."); ic++; break;
    → tellement\n"
                                                case 6: System.exit(0);
       + "Tellement amusant\n"
                                                case 100: S.o.println(mem[0]+"éléphants...");
       + "Que tout à coup\n"
                                                → ic++: break:
       + "Ba da boum"); // 5: affichage
                                                case 101: mem[0]++; ic++; break;
 case 102:
                                            1 1 1 1
```

```
public class AppelSimpleDeux {
                                            import java.util.Stack;
 static int k = 1;  // 0: init mem[0]
                                            public class AppelSimpleDeuxTrad {
                                              static int ic = 0; int[] mem = new int[1000];
 public static void g() {
                                              static Stack<Bloc> p = new Stack<Bloc>();
   System.out.println(k+ " éléphants ...
                                              public static void main(String[] args) {

→ d'araignée"); // 100

                                               while(true) {
   k++; // 101: modif mem[0]
                                                switch(ic) {
           // 102: sortie q
                                                case 0: mem[0] = 1; ic++; break; // init. de k
                                                case 1: if(mem[0] > 10){ic += 4;} else {ic++;};
 public static void main(String[] args) {
                                                        break: //saut cond.
   while (k <= 10 ) { // 1: saut cond.
                                                case 2: p.push(new Bloc(ic+1)); ic = 100; break;
     g(); // 2: appel à q
                                                       //empiler le bloc d'act. + saut incond.
                                                case 3: p.pop(); ic++; break;// dépiler le bloc
              // 3: point de retour
         // 4: saut incond.
                                                case 4: ic -= 3; break; //saut incond.
   System.out.println("C'était un jeu
                                                case 5: S.out.println("C'était ..."); ic++; break;
    → tellement\n"
                                                case 6: System.exit(0);
                                                case 100: S.o.println(mem[0]+"éléphants...");
       + "Tellement amusant\n"

→ ic++; break;

       + "Que tout à coup\n"
       + "Ba da boum"); // 5: affichage
                                                case 101: mem[0]++; ic++; break;
 case 102: ic = p.peek().getAdresse(); break;
                                                         //sortie de q: saut incondionnel vers
                                                         → l'adresse de retour
```

2 2 2 2

# Bloc d'activation pour les fonctions avec un paramètre

Nous considérons maintenant une classe modélisant le bloc d'activation d'une fonction avec un paramètre, mais sans valeur de retour ni variables locales.

```
public class BlocAvecPar {
   private int adresseRetour;
   private int param;
   // constructeur: l'appelant doit spécifier l'adresse de retour et la valeur du paramètre
   public BlocAvecPar(int adr, int param) {
     this.adresseRetour = adr:
     this.param = param:
   // accesseurs utilisés par le code appelé
   public int getAdresse() {
     return this.adresseRetour;
   public int getParam() {
     return this.param;
```

```
public class AppelSimple3 {
 public static void g(int k) {
   System.out.println(k+ " éléphants ...");
 public static void main(String[] args) {
   int k = 1;  // 0: init mem[0]
   while (k <= 10 ) { // 1: saut incondinnel vers 6
     g(k++);
   System.out.println("... ba da boum");
```

```
public class AppelSimple3 {
 public static void g(int k) {
   System.out.println(k+ " éléphants ...");
 public static void main(String[] args) {
   int k = 1: // 0: init mem[0]
   while (k <= 10 ) { // 1: saut incondinnel vers 6
     g(k++); // 2: appel \tilde{a} q
                     1/ 3: retour
                     // 4: incrémentation de k
   System.out.println("... ba da boum");
```

```
public class AppelSimple3 {
 public static void g(int k) {
   System.out.println(k+ " éléphants ...");
 public static void main(String[] args) {
   int k = 1: // 0: init mem[0]
   while (k <= 10 ) { // 1: saut incondinnel vers 6
     g(k++); // 2: appel \tilde{a} q
                     // 3: retour
                     // 4: incrémentation de k
                  // 5: saut inconditionnel vers 1
   System.out.println("... ba da boum"); // 6
 } // 7: sortie du programme
```

```
public class AppelSimple3 {
 public static void g(int k) {
   System.out.println(k+ " éléphants ..."); // 100
 } //101: sortie du q, saut incondionnel vers l'adresse de retour
 public static void main(String[] args) {
   int k = 1: // 0: init mem[0]
   while (k <= 10 ) { // 1: saut incondinnel vers 6
     g(k++); // 2: appel \tilde{a} q
                     // 3: retour
                     // 4: incrémentation de k
                  // 5: saut inconditionnel vers 1
   System.out.println("... ba da boum"); // 6
 } // 7: sortie du programme
```

```
public class AppelSimple3 {
                                              import java.util.Stack;
                                              public class AppelSimple3Trad {
 public static void g(int k) {
                                                static int ic = 0; int [] mem = new int[1000];
   System.out.println(k+ " éléphants ...");
                                                static Stack<BlocAvecPar> p = new

→ // 100

→ Stack<BlocAvecPar>();
                                                public static void main(String[] args) {
 } //101: sortie du q, saut incondionnel
  uers l'adresse de retour
                                                 while(true) {
                                                  switch(ic) {
 public static void main(String[] args) {
                                                  case 0: mem[0] = 1; ic++; break; // init. de k
   int k = 1: // 0: init mem [0]
                                                  case 1:
   while (k <= 10 ) { // 1: saut
    → incondionnel vers 6
                                                  case 2:
     g(k++); // 2: appel à q
                      1/ 3: retour
                                                  case 3:
                      // 4: incrémentation
                                                  case 4:
                      \rightarrow de k
                                                  case 5:
                     // 5: saut
                                                  case 6:
    → inconditionnel vers 1
                                                  case 7:
   System.out.println("... ba da boum"); //
                                                  case 100:

→ 6

                                                  case 101:
 } // 7: sortie du programme
                                               1 1 1 1
```

```
public class AppelSimple3 {
                                              import java.util.Stack;
                                              public class AppelSimple3Trad {
 public static void g(int k) {
                                                static int ic = 0; int [] mem = new int[1000];
   System.out.println(k+ " éléphants ...");
                                                static Stack<BlocAvecPar> p = new

→ // 100

→ Stack<BlocAvecPar>();
 } //101: sortie du q, saut incondionnel
                                                public static void main(String[] args) {
  uers l'adresse de retour
                                                 while(true) {
                                                  switch(ic) {
 public static void main(String[] args) {
                                                  case 0: mem[0] = 1; ic++; break; // init. de k
   int k = 1: // 0: init mem [0]
                                                  case 1: if(mem[0]>10) {ic+= 5:} else {ic++:}:
   while (k <= 10 ) { // 1: saut

→ break:

    → incondionnel vers 6
                                                  case 2:
     g(k++); // 2: appel à q
                                                  case 3:
                      1/ 3: retour
                                                  case 4:
                      // 4: incrémentation
                                                  case 5:
                      \rightarrow de k
                                                  case 6:
                     // 5: saut
                                                  case 7:
    → inconditionnel vers 1
                                                  case 100:
   System.out.println("... ba da boum"); //
                                                  case 101:
    → 6
                                               1 1 1 1
 } // 7: sortie du programme
```

```
public class AppelSimple3Trad {
public static void g(int k) {
                                               static int ic = 0; int [] mem = new int[1000];
  System.out.println(k+ " éléphants ...");
                                               static Stack<BlocAvecPar> p = new
  → // 100

→ Stack<BlocAvecPar>();
} //101: sortie du q, saut incondionnel
                                              public static void main(String[] args) {
uers l'adresse de retour
                                               while(true) {
                                                 switch(ic) {
public static void main(String[] args) {
                                                 case 0: mem[0] = 1; ic++; break; // init. de k
  int k = 1: // 0: init mem [0]
                                                 case 1: if(mem[0]>10) {ic+= 5;} else {ic++;};
  while (k \le 10) \{ // 1 : saut \}
                                                        break:
  → incondionnel vers 6
                                                 case 2: p.push(new BlocAvecPar(ic+1, mem[0])); //
   g(k++); // 2: appel \hat{a} q
                                                 → on transmets l'@ de retour et la val. du
                    // 3: retour
                                                 → paramètre,
                    // 4: incrémentation
                                                        ic = 100; break; // puis saut
                     \rightarrow de k
                                                         inconditionnel
                   // 5: saut
                                                 case 3:
  → inconditionnel vers 1
                                                 case 4:
  System.out.println("... ba da boum"); //
                                                 case 5:
  → 6
                                                 case 6:
} // 7: sortie du programme
                                                 case 7:
                                                 case 100:
```

import java.util.Stack;

public class AppelSimple3 {

```
public class AppelSimple3 {
                                              import java.util.Stack;
                                              public class AppelSimple3Trad {
 public static void g(int k) {
                                                static int ic = 0; int [] mem = new int[1000];
   System.out.println(k+ " éléphants ...");
                                                static Stack<BlocAvecPar> p = new

→ // 100

→ Stack<BlocAvecPar>();
 } //101: sortie du q, saut incondionnel
                                                public static void main(String[] args) {
  uers l'adresse de retour
                                                 while(true) {
                                                  switch(ic) {
 public static void main(String[] args) {
                                                  case 0: mem[0] = 1; ic++; break; // init. de k
   int k = 1: // 0: init mem [0]
                                                  case 1: if(mem[0]>10) {ic+= 5;} else {ic++;};
   while (k <= 10 ) { // 1: saut
                                                          break:
    → incondionnel vers 6
                                                  case 2: p.push(new BlocAvecPar(ic+1, mem[0]));
     g(k++); // 2: appel à q
                                                          ic = 100; break;
                      1/ 3: retour
                                                  case 3: p.pop(); ic++; break;
                      // 4: incrémentation
                                                  case 4:
                      \rightarrow de k
                                                  case 5:
                     // 5: saut
                                                  case 6:
    → inconditionnel vers 1
                                                  case 7:
   System.out.println("... ba da boum"); //
                                                  case 100:
    → 6
                                                  case 101:
                                               } } } }
 } // 7: sortie du programme
```

```
public class AppelSimple3 {
                                              import java.util.Stack;
                                              public class AppelSimple3Trad {
 public static void g(int k) {
                                                static int ic = 0; int [] mem = new int[1000];
   System.out.println(k+ " éléphants ...");
                                                static Stack<BlocAvecPar> p = new

→ // 100

→ Stack<BlocAvecPar>();
 } //101: sortie du q, saut incondionnel
                                                public static void main(String[] args) {
  uers l'adresse de retour
                                                 while(true) {
                                                  switch(ic) {
 public static void main(String[] args) {
                                                  case 0: mem[0] = 1; ic++; break; // init. de k
   int k = 1: // 0: init mem [0]
                                                  case 1: if(mem[0]>10) {ic+= 5:} else {ic++:}:
   while (k <= 10 ) { // 1: saut
                                                          break:
    → incondionnel vers 6
                                                  case 2: p.push(new BlocAvecPar(ic+1, mem[0]));
     g(k++); // 2: appel à q
                                                          ic = 100; break;
                      1/ 3: retour
                                                  case 3: p.pop(); ic++; break;
                      // 4: incrémentation
                                                  case 4: mem[0]++: ic++: break:
                      \rightarrow de k
                                                  case 5.
                     // 5: saut
                                                  case 6:
    → inconditionnel vers 1
                                                  case 7:
   System.out.println("... ba da boum"); //
                                                  case 100:
    → 6
                                                  case 101:
 } // 7: sortie du programme
                                               1 1 1 1
```

```
public class AppelSimple3 {
                                              import java.util.Stack;
                                              public class AppelSimple3Trad {
 public static void g(int k) {
                                                static int ic = 0; int [] mem = new int[1000];
   System.out.println(k+ " éléphants ...");
                                                static Stack<BlocAvecPar> p = new

→ // 100

→ Stack<BlocAvecPar>();
 } //101: sortie du q, saut incondionnel
                                                public static void main(String[] args) {
  uers l'adresse de retour
                                                 while(true) {
                                                  switch(ic) {
 public static void main(String[] args) {
                                                  case 0: mem[0] = 1; ic++; break; // init. de k
   int k = 1;  // O: init mem[0]
                                                  case 1: if(mem[0]>10) {ic+= 5:} else {ic++:}:
   while (k <= 10 ) { // 1: saut
                                                          break:
    → incondionnel vers 6
                                                  case 2: p.push(new BlocAvecPar(ic+1, mem[0]));
     g(k++); // 2: appel à q
                                                          ic = 100; break;
                      1/ 3: retour
                                                  case 3: p.pop(); ic++; break;
                      // 4: incrémentation
                                                  case 4: mem[0]++; ic++; break;
                      \rightarrow de k
                                                  case 5: ic -= 4: break:
                     // 5: saut
                                                  case 6:
    → inconditionnel vers 1
                                                  case 7:
   System.out.println("... ba da boum"); //
                                                  case 100:
    → 6
                                                  case 101:
 } // 7: sortie du programme
                                               1 1 1 1
```

```
public class AppelSimple3 {
                                              import java.util.Stack;
                                              public class AppelSimple3Trad {
 public static void g(int k) {
                                                static int ic = 0; int [] mem = new int[1000];
   System.out.println(k+ " éléphants ...");
                                                static Stack<BlocAvecPar> p = new

→ // 100

→ Stack<BlocAvecPar>();
 } //101: sortie du q, saut incondionnel
                                                public static void main(String[] args) {
  uers l'adresse de retour
                                                 while(true) {
                                                  switch(ic) {
 public static void main(String[] args) {
                                                  case 0: mem[0] = 1; ic++; break; // init. de k
   int k = 1: // 0: init mem [0]
                                                  case 1: if(mem[0]>10) {ic+= 5:} else {ic++:}:
   while (k <= 10 ) { // 1: saut
                                                          break:
    → incondionnel vers 6
                                                  case 2: p.push(new BlocAvecPar(ic+1, mem[0]));
     g(k++); // 2: appel à q
                                                          ic = 100; break;
                      1/ 3: retour
                                                  case 3: p.pop(); ic++; break;
                      // 4: incrémentation
                                                  case 4: mem[0]++; ic++; break;
                      \rightarrow de k
                                                  case 5: ic -= 4: break:
                     // 5: saut
                                                  case 6: System.out.println("... ba da boum");
    → inconditionnel vers 1

    ic++: break:

   System.out.println("... ba da boum"); //
                                                  case 7: System.exit(0);
    → 6
                                                  case 100:
 } // 7: sortie du programme
                                                  case 101:
                                               1 1 1 1
```

```
public class AppelSimple3 {
                                               import java.util.Stack;
                                               public class AppelSimple3Trad {
 public static void g(int k) {
                                                 static int ic = 0; int [] mem = new int[1000];
   System.out.println(k+ " éléphants ...");
                                                 static Stack<BlocAvecPar> p = new
    → // 100

→ Stack<BlocAvecPar>();
 } //101: sortie du q, saut incondionnel
                                                 public static void main(String[] args) {
  uers l'adresse de retour
                                                 while(true) {
                                                   switch(ic) {
 public static void main(String[] args) {
                                                   case 0: mem[0] = 1; ic++; break; // init. de k
   int k = 1: // 0: init mem [0]
                                                   case 1: if(mem[0]>10) {ic+= 5;} else {ic++;};
   while (k \le 10) \{ // 1 : saut \}
                                                           break:
    → incondionnel vers 6
                                                   case 2: p.push(new BlocAvecPar(ic+1, mem[0]));
     g(k++); // 2: appel \hat{a} q
                                                          ic = 100; break;
                      1/ 3: retour
                                                   case 3: p.pop(); ic++; break;
                      // 4: incrémentation
                                                   case 4: mem[0]++; ic++; break;
                       \rightarrow de k
                                                   case 5: ic -= 4: break:
                     // 5: saut
                                                   case 6: System.out.println("... ba da boum");
    → inconditionnel vers 1

    ic++: break:

   System.out.println("... ba da boum"); //
                                                   case 7: System.exit(0);
    → 6
                                                   case 100: System.out.println(p.peek().getParam()+"
 } // 7: sortie du programme

→ éléphants ... "); ic++;break;
                                                   case 101:
```

```
public class AppelSimple3Trad {
public static void g(int k) {
                                               static int ic = 0; int [] mem = new int[1000];
  System.out.println(k+ " éléphants ...");
                                               static Stack<BlocAvecPar> p = new
  → // 100

→ Stack<BlocAvecPar>();
} //101: sortie du q, saut incondionnel
                                               public static void main(String[] args) {
uers l'adresse de retour
                                                while(true) {
                                                 switch(ic) {
public static void main(String[] args) {
                                                 case 0: mem[0] = 1; ic++; break; // init. de k
  int k = 1: // 0: init mem [0]
                                                 case 1: if(mem[0]>10) {ic+= 5;} else {ic++;};
  while (k \le 10) \{ // 1 : saut \}
                                                         break:
  → incondionnel vers 6
                                                 case 2: p.push(new BlocAvecPar(ic+1, mem[0]));
   g(k++); // 2: appel \hat{a} q
                                                        ic = 100; break;
                     // 3: retour
                                                 case 3: p.pop(); ic++; break;
                     // 4: incrémentation
                                                 case 4: mem[0]++; ic++; break;
                     \rightarrow de k
                                                 case 5: ic -= 4: break:
                    // 5: saut
                                                 case 6: System.out.println("... ba da boum");
  → inconditionnel vers 1

    ic++: break:

  System.out.println("... ba da boum"); //
                                                 case 7: System.exit(0);
  → 6
                                                 case 100: System.out.println(p.peek().getParam()+"
} // 7: sortie du programme

→ éléphants ... "); ic++;break;
                                                 case 101: ic = p.peek().getAdresse(); break;
```

import java.util.Stack;

public class AppelSimple3 {