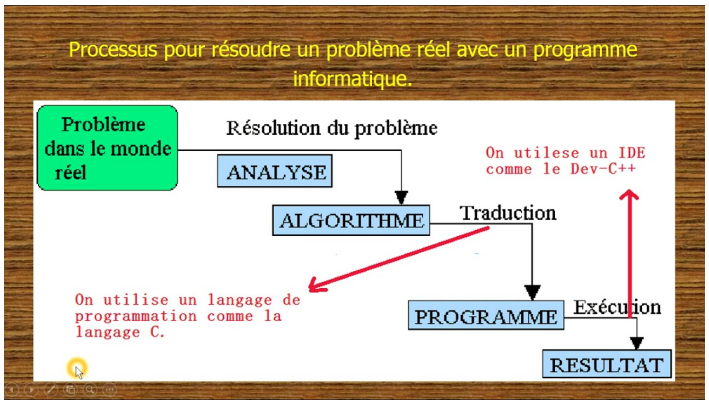
Les Structures Algorithmiques

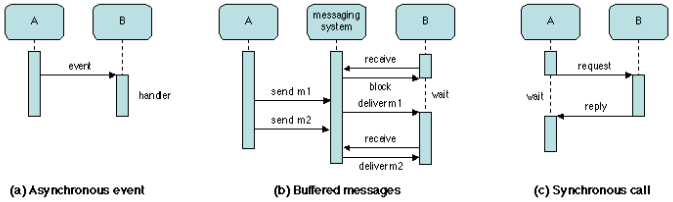


Les structures algorithmiques en Java sont des modèles de contrôle qui définissent la manière dont les instructions et les opérations sont exécutées dans un programme. Ces structures sont essentielles pour structurer les algorithmes de manière claire et efficace. Les principales structures algorithmiques comprennent les structures séquentielles, les structures conditionnelles, et les structures itératives.

- **Les structures de contrôles permettent d'arrêter l'exécution linéaire des instructions (de bas en haut et de gauche à droite)**
- **Elles permettent d'exécuter conditionnellement une instruction, ou de réaliser une boucle**

Type d'instruction	Mots clés utilisés
Décision	if() else – switch() case
Boucle	for(; ;) – while () – do while()
Traitement d'exceptions	try catch finally – throw
Branchement	label : -- break – continue -- return

1. La Structure Séquentielle



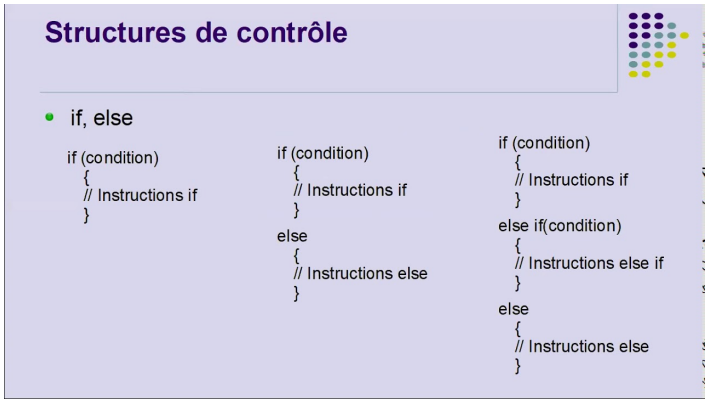
(a) Asynchronous event (b) Buffered messages (c) Synchronous call

La structure séquentielle est la plus simple des structures algorithmiques. Les instructions sont exécutées dans l'ordre où elles apparaissent dans le code, une après l'autre.

```
int a = 5;
int b = 10;
int somme = a + b;
System.out.println("La somme est : " + somme);
```

Dans cet exemple, les instructions sont exécutées séquentiellement. La valeur de `a` est ajoutée à `b`, et le résultat est stocké dans `somme`, qui est ensuite affiché.

2. La Structure Conditionnelle



Les structures conditionnelles permettent d'exécuter des blocs de code en fonction de conditions spécifiques. Les principales structures conditionnelles en Java sont `if`, `else if`, `else`, et `switch`.

a) `if`, `else if`, `else`

```
int x = 10;

if (x > 0) {
    System.out.println("x est positif");
} else if (x < 0) {
    System.out.println("x est négatif");
} else {
    System.out.println("x est zéro");
}
```

Dans cet exemple : - Le bloc `if` est exécuté si la condition `x > 0` est vraie. - Si cette condition est fausse, le programme vérifie la condition `else if`. - Si aucune condition n'est vraie, le bloc `else` est exécuté.

b) `switch`

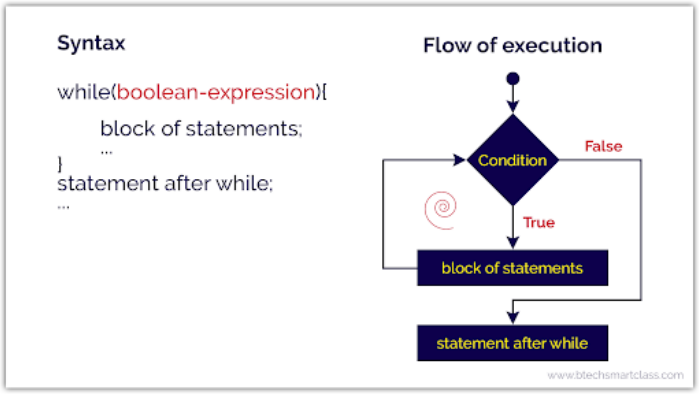
Le `switch` permet de choisir parmi plusieurs options en fonction de la valeur d'une variable.

```
int jour = 3;
String jourNom;

switch (jour) {
    case 1:
        jourNom = "Lundi";
        break;
    case 2:
        jourNom = "Mardi";
        break;
    case 3:
        jourNom = "Mercredi";
        break;
    // Autres cas...
    default:
        jourNom = "Jour invalide";
        break;
}

System.out.println("Le jour est : " + jourNom);
```

3. La Structure Itérative



Les structures itératives permettent de répéter l'exécution d'un bloc de code tant qu'une condition est remplie. Java propose trois types de boucles : `for`, `while`, et `do-while`.

a) Boucle `for`

```
for (int i = 0; i < 5; i++) {
    System.out.println("i vaut : " + i);
}
```

La boucle `for` est idéale lorsque le nombre d'itérations est connu. Dans cet exemple, la boucle s'exécute 5 fois.

b) Boucle `while`

```
int i = 0;
while (i < 5) {
    System.out.println("i vaut : " + i);
    i++;
}
```

La boucle `while` s'exécute tant que la condition `i < 5` est vraie. Elle est souvent utilisée lorsque le nombre d'itérations n'est pas connu à l'avance.

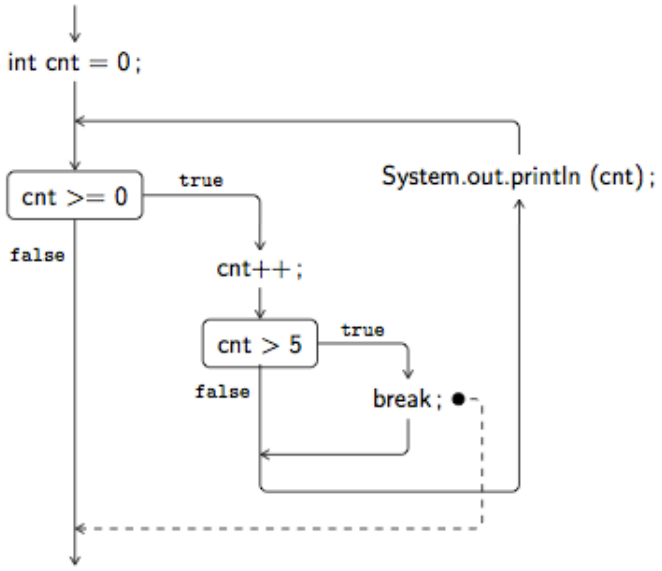
c) Boucle `do-while`

```
int i = 0;
do {
    System.out.println("i vaut : " + i);
```

```
    i++;
} while (i < 5);
```

La boucle `do-while` est similaire à `while`, mais elle garantit que le bloc de code s'exécute au moins une fois, car la condition est vérifiée après l'exécution du bloc.

4. Les Structures de Saut



Les structures de saut permettent de contrôler le flux d'exécution en interrompant ou en sautant certaines parties du code. En Java, on utilise principalement `break`, `continue`, et `return`.

a) `break`

Le `break` est utilisé pour sortir prématurément d'une boucle ou d'un `switch`.

```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        break;
    }
    System.out.println("i vaut : " + i);
}
```

b) `continue`

Le `continue` saute à l'itération suivante de la boucle sans terminer l'itération actuelle.

```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        continue;
    }
    System.out.println("i vaut : " + i);
}
```

c) `return`

Le `return` est utilisé pour sortir d'une méthode et retourner une valeur.

```
public int somme(int a, int b) {
    return a + b;
}
```

Conclusion

Les structures algorithmiques en Java sont essentielles pour contrôler l'exécution des programmes. En comprenant comment utiliser les structures séquentielles, conditionnelles, itératives et de saut, vous pouvez écrire des algorithmes efficaces et bien structurés pour résoudre une grande variété de problèmes.