Interface = implémentation

INTERFACE -> 1 seule méthode

1 classe anonyme

1Lambda

```
Public interface MonInterface{
    public void maMethode(){

    }
}
```

MAIN (Appelant)

```
MonInterface monContrat=new MonInterface(

        public void maMethode(){

        }
));
```

MAIN (Appelant)

```
MonInterface monContrat=(x) ->{ //code de la maMéthode;}
```

```
Public interface MonInterface{

    public void maMethode(){

     }

    public int getAge(){

   }  //Supplier


   public void printAge(int age){
        Sysout.out.println(age)

    } //Consumer

   public boolean test(int age){

     return true|false;

}
```

Méthode 1 type de retour

Méthode sans type de retour avec paramètre

Méthode avec type parametreEntre

Methode sansParametre

Methode qui fait un test

```
List<int> ages=new ArrayList();

ages.add(1); //i0
Ages.add(24) //i1
Ages.add(15); //i2

Ages.stream().filter( (i) ->{ isMajeur(i)   } );

Public boolean isMajeur(int age){
      return age=>18;
}
```

```
Public interface MonInterface{

    public void maMethode(){

     }

    public int getAge(){

  }  //Supplier


  public void printAge(int age){
        Sysout.out.println(age)

   } //Consumer

  public boolean test(int age){

    return true|false;

}
```

```
List<int> ages=new ArrayList();

ages.add(1); //i0
ages.add(24) //i1
ages.add(15); //i2

ages.stream().filter( (i) ->{ isMajeur(i)   } );

ages.foreach( (i) -> displayAge(i) );

ages.generate( () -> monSupplier() );


Public String monSupplier(){
    return « hello »;
}
Public void displayAge(int age){

    System.out.println(age);
}

Public boolean isMajeur(int age){
     return age=>18;
}
```