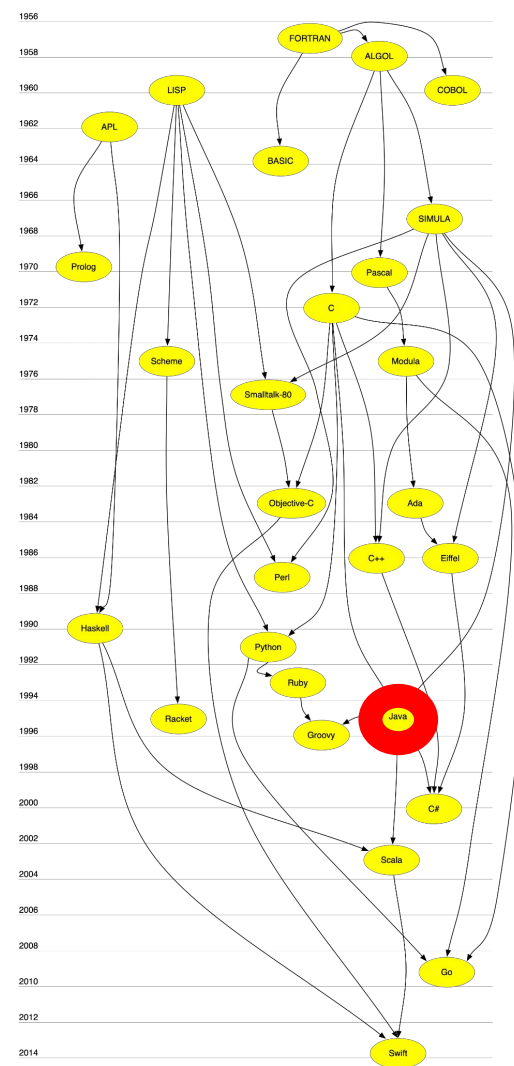


PRÈVIA

Programació Funcional

ANTECEDENTS DE JAVA



ANTECEDENTS DE JAVA - LLENGUATGES IMPERATIUS

Llenguatge d'assemblador(47)

SpeedCoding(53) Fortran(57) Algol(58)

CPL/BCPL/B (63/67/69)

C(72)

C++ (85)

Java (95)

Kotlin (2011)

El programador treballa a baix nivell amb operacions molt bàsiques

```
mult.asm      Wed Feb 02 15:42:09 2000      1
; Implements a simple 32 bit integer multiply by successive addition
; R. H. Klenke, Sun Jan 31 10:45:14 EST 1999
;

Multp: .equ    32          ; multiplicand
Multd: .equ    64          ; multiplier
      .org      4096        ; 0x1000
      lar      r30, Done    ; Load address of Done for branch
      lar      r31, Loop    ; Load address of Loop for branch
      la       r1, Multd    ; Load multiplier
      la       r2, Multp    ; Load multiplicand
      andi     r3, r3, #0    ; clear r3
      andi     r4, r4, #0    ; clear r4
      addi     r5, r3, #1    ; place 1 in r5
      neg      r3, r5        ; place -1 in r3
Loop:  add     r4, r4, r2     ; add multiplicand to running sum
      add     r1, r1, r3     ; start loop, decrement multiplier
      brzr    r30, r1        ; jump to Done if multiplier = 0
      br      r31            ; jump back to Loop
Done:  st      r4, Result    ; store result
      stop
      .org      8192        ; 0x2000
Result: .dw    1            ; storage for result
```

ANTECEDENTS DE JAVA - LLENGUATGES IMPERATIUS

Llenguatge d'assemblador(47)

SpeedCoding(53) Fortran(57) Algol(58)

CPL/BCPL/B (63/67/69)

C(72)

C++ (85)

Java (95)

Kotlin (2011)

Utilitzats per al càlcul numèric.

Poder fer operacions amb una única instrucció.

Estructures bàsiques a partir dels anys 70. Més tard afegiran arrays o programació modular.

```
1      FORMAT(5F10.2)
      DO 10 J = 1, 11
        I = 11 - J
        Y = FUN(A(I+1))
        IF (400.0-Y) 4, 8, 8
4       WRITE (6,5) I
5       FORMAT(I10, 10H TOO LARGE)
        GO TO 10
8       WRITE (6,9) I, Y
        FORMAT(I10, F12.6)
10     CONTINUE
      STOP
      END
```

ANTECEDENTS DE JAVA - LLENGUATGES IMPERATIUS

Llenguatge d'assemblador(47)

SpeedCoding(53) Fortran(57) Algol(58)

CPL/BCPL/B (63/67/69)

C(72)

C++ (85)

Java (95)

Kotlin (2011)

CPL era un llenguatge gran i complexa que va donar lloc a BCPL i després a B en un intent de simplificar-lo.

Llenguatge per a programació de sistemes, particularment per escriure compiladors.

B introdueix per primera vegada les claus per a delimitar blocs de codi així com les referències.

ANTECEDENTS DE JAVA - LLENGUATGES IMPERATIUS

Llenguatge d'assemblador(47)

SpeedCoding(53) Fortran(57) Algol(58)

CPL/BCPL/B (63/67/69)

C(72)

C++ (85)

Java (95)

Kotlin (2011)

És un llenguatge orientat a la implementació de sistemes operatius, concretament Unix.

És apreciat per l'eficiència del codi que produeix i és el llenguatge de programació més popular per crear programes de sistemes i aplicacions.

```
/* Suma de n números */  
#include <stdio.h>  
int main() {  
    int num=0,suma=0;  
  
    do {  
        suma=suma+num;  
        printf("un número: ");  
        scanf("%d",&num);  
    } while(num>=0);  
    printf("suma es: %d",suma);  
    return 0;  
}
```

ANTECEDENTS DE JAVA - LLENGUATGES IMPERATIUS

Llenguatge d'assemblador(47)

SpeedCoding(53) Fortran(57) Algol(58)

CPL/BCPL/B (63/67/69)

C(72)


C++ (83)

Java (95)

Kotlin (2011)

Naix per a afegir la programació orientada a objectes a C.

```
class Base {  
    public:  
    void print() {  
        // code  
    }  
};  
  
class Derived : public Base {  
    public:  
    void print() {  
        // code  
    }  
};  
  
int main() {  
    Derived derived1;  
    derived1.print();  
    return 0;  
}
```



ANTECEDENTS DE JAVA - LLENGUATGES IMPERATIUS

Llenguatge d'assemblador(47)

SpeedCoding(53) Fortran(57) Algol(58)

CPL/BCPL/B (63/67/69)

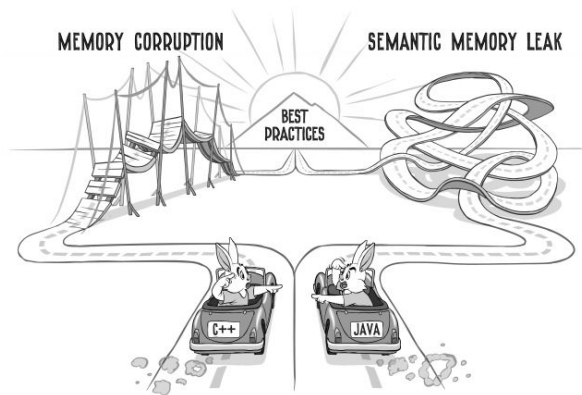
C(72)

C++ (83)

Java (95)

Kotlin (2011)

Construir programes multiplataforma i eliminar funcions de baix nivell així com tractar automàticament els punters i recollector de brossa.



ANTECEDENTS DE JAVA - LLENGUATGES IMPERATIUS

Llenguatge d'assemblador(47)

SpeedCoding(53) Fortran(57) Algol(58)

CPL/BCPL/B (63/67/69)

C(72)

C++ (83)

Java (95)

Kotlin (2011)

Vistes les mancances de Java decideixen crear un llenguatge que treballa en JVM i que tinga més funcionalitats, o que una mateixa tasca es puga fer més ràpid i amb menys línies de codi.

KOTLIN - EVOLUCIÓ NATURAL DE JAVA?

Java

```
final int x = // value;
final String xResult;

switch (x) {
    case 0;
    case 11;
        xResult = "0 or 11";
        break;
    case 1;
    case 2;
        //...
    case 10;
        xResult = "from 1 to 10";
        break;
    default;
        if (x < 12 && x > 14) {
            xResult = "not from 12 to 14"
            break;
        }

        if (isOdd(x)) {
            xResult = "is odd";
            break;
        }
    }

    xResult = "otherwise"
}
```

```
final int y = //value;
final String yResult;

if(isNegative(y)) {
    yResult = "is zero";
} else if(isOdd(y)) {
    yResult = "is odd";
} else {
    yResult = "otherwise";
}
```

Kotlin

```
val x = //value
val xResult = when (x) {
    0, 11 -> "0 or 11"
    in 1..10 -> "from 1 to 10"
    !in 12..14 -> "not true from 12 to 14"
    else -> if (isOdd(x)) { "is odd" } else { "otherwise" }
}

val y = //value
val yResult = when {
    isNegative(y) -> "is negative"
    isZero(y) -> "is zero"
    isOdd(y) -> "is odd"
    else -> "otherwise"
}
```

Java

```
public class User {
    private final String firstName;
    private final String lastName;
    private final int age;

    public User(String firstName, String lastName, int age) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public int getAge() {
        return age;
    }

    public String toString() {
        return firstName + " " + lastName + ", age " + age;
    }
}

class Main {
    public static void main(String[] args) {
        System.out.println(new User("John", "Doe", 30));
    }
}
```

Kotlin

```
public class User(val firstName: String,
                  val lastName: String,
                  val age: Int) {

    fun toString() = "$firstName $lastName, age $age"
}

fun main(args : Array<String>) {
    println(User("John", "Doe", 30))
}
```

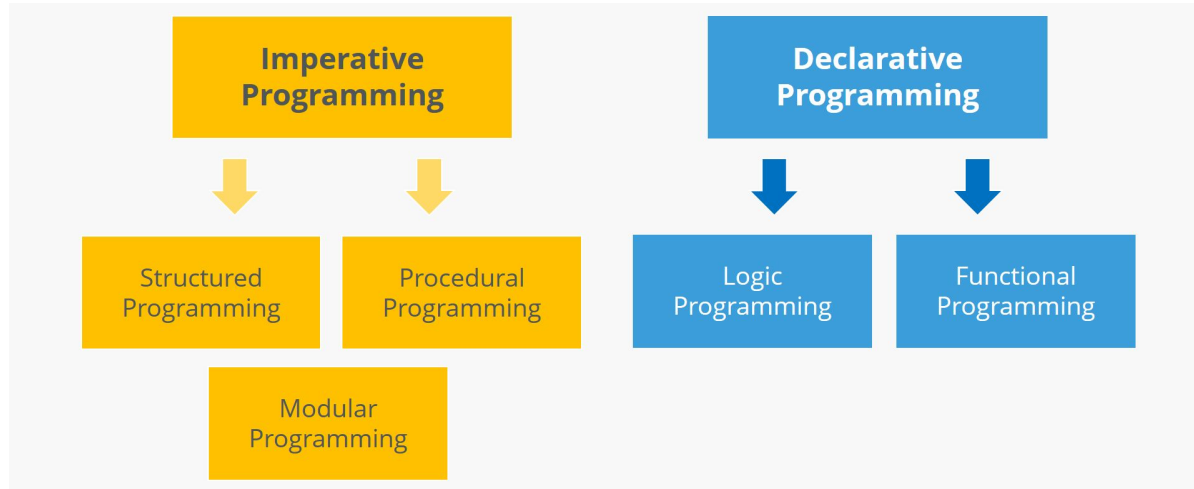
Java Null Checks

```
private ActorsData getActorsData(MediaData data){
    if (data != null){
        if (data.getmovieData() != null){
            if (data.getmovieData().getActorsData() != null){
                return data.getmovieData().getActorsData();
            }
        }
    }
    return null;
}
```

Kotlin Null Checks

```
fun getActorsData(data : MediaData) : ActorsData? =
    data?.movieData?.actorsData
```

PARADIGMES DE PROGRAMACIÓ



- Els llenguatges **declaratius** no són nous. Com a exemple de paradigma de programació lògica tenim Prolog de 1972 i en el paradigma de programació funcional tenim Lisp de 1958.

EXEMPLES D'ALTRES LLINGUATGES FUNCIONALS



Haskell



Clojure



Purescript

DUBTES...

¿Es pot considerar Java com un llenguatge funcional?
¿I JavaScript?

COMENCEM...

