```
void func (in value1, out value2, ref value3) ⇒ C#
```

```
void func (char... characters){
        char value = characters[0]; }    ⇒ Java
```

```
void func (int parametre_sayisi, ...){
    va_list valist;
    va_start (valist, parametre_sayisi);    ⇒ C (stdarg.h)
    int parametre1 = va_arg (valist, int);
    char parametre1 = va_arg (valist, char);
    va_end (valist); }
```

```
implicit declaration function ⇒ C
```

```
int i;
for (i=0; i<3; ++i)    } ⇒ C
```

```
for (int i : array) ⇒ Java
```

```
foreach (int i in array) ⇒ C#
```

```
typedef enum { true, false } bool ⇒ C
```

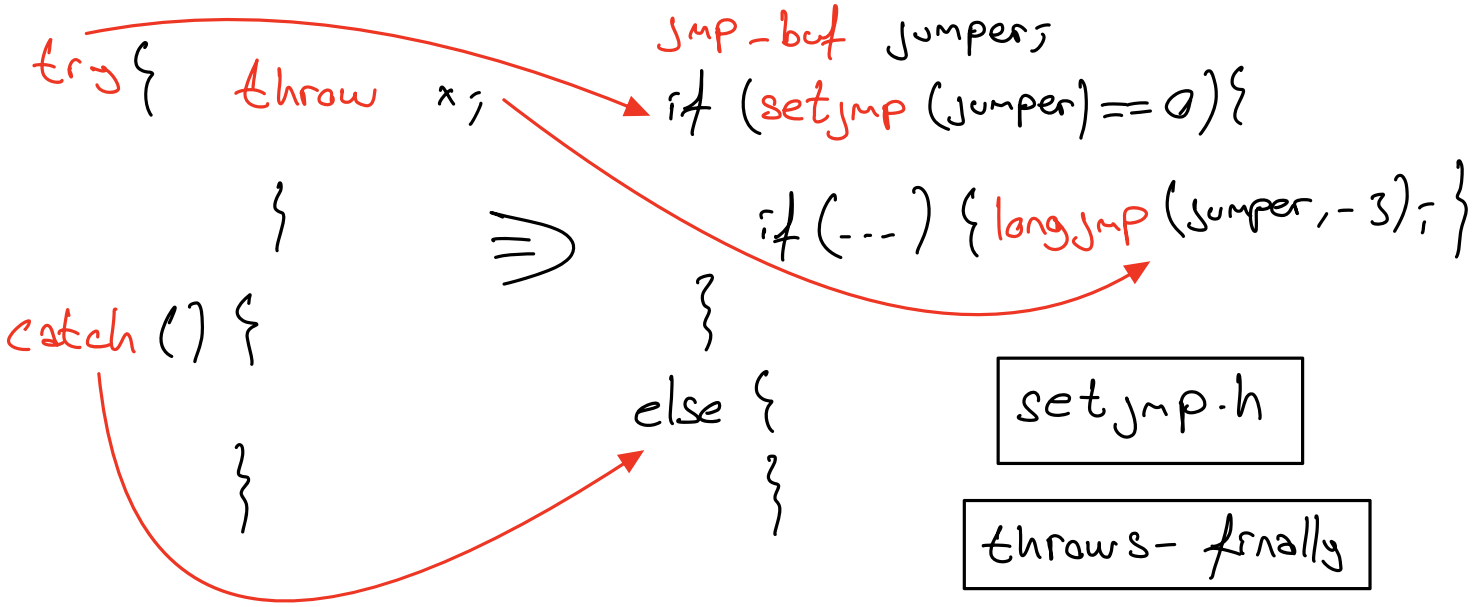
```
loop1:
for (...) {
        loop2:
        for (...) {                    ⇒ Java
                if (...) break loop2;
                if (...) break loop1; }
    }
```

Finalize ⇒ Java Dispose ⇒ C#

public class sinif implementation arayüz ⇒ Java
extends baskaSinif

super ⇒ Java base ⇒ C#

Çoklu Kalıtım ⇒ Diamond Problemi ⇒ Virtual



Paralel çalışabilir ⇒ Read(S1) ∩ write(S2) = { }
write(S1) ∩ Read (S2) = { }
write(S1) ∩ write(S2) = { }

Fork - Join (count) , Parbegin - Parend (begin -end), öncelik grafı, işlem grafı

Kritik Bölge ⇒ Mutual Exclusion + Progress + Bounded Waiting

```
public class Sinif implements Runnable{
 private final Lock bolge=new ReentrantLock();
@Override
public void run() { bolge.lock();
                    ... bolge.unlock();}

Runnable sinif = new Sinif();
Thread thread = new Thread(sinif);
thread.start();
```

→ veya →

```
ExecutorService havuz =
new Executors.newFixedThreadPool(3);
havuz.execute(new Sinif());
havuz.shutdown();
while (!havuz.isTerminated()) {}
```

```
void* run() {...}
int main(){ pthread_t th;
    int hata= pthread_create (&th, NULL, run, parametre);
    pthread_join (th, NULL);
    return 0; }
```

pthread.h

Running, Blocked, Ready, Deadlock

Semafor ⇒ P , V

```
while(S==0)      S=S+1
S=S-1;
```

Turn = j ⇒ Algoritma hangi işlemin kritik bölgesine girmesine izin verdiğini hatırlar.

Flag[i] = False ⇒ Ayrıca, işlemin hangi aşamada olduğunu hatırlar.

Her ikisi ⇒ Birden fazla threadin kritik bölgeye girmesine izin yoktur.

```
(Defun FonksiyonAdi(parametreler)
(setq x 5 y (read))
(setq z (+ x y))
(print "Sonuc:")
(print z)
(print (fonksiyon))
5)
```

```
(incf x)
(decf x)
(rotatef x y)
(setq x #b100)
(setq x #c(1 2))
; Yorum
```

```
(if(or(and(< x y)(> y z))(/=22))
(dogru) ⇝ (progn ...)
(yanlis))
```

Sayı döndü 1

```
(loop for i from 1 to 10 do
 (print (mod 5 2))
 (print (concatenate 'string isim "123"))
 ⋮
)
```

```
(loop until (< x 5) do
   (...)
   (...)
   ⋮
)
(loop while (< x 5) do
   (...)
   (...)
   ⋮
)
```

```
(Defun fonk(x &optional y z)
(list x y z))
⟶ (fonk 1 3)

(Defun fonk(x &key y z)
 ...)
⟶ (fonk 2 :z 5 :y 1)
```

```
(defparameter *A* (list 1 2 3))
(first *A*)
(rest *A*)
(append *A* *B*)
(reverse *A*)
(find 3 *A*)
(position 3 *A*)
(delete 3 *A*)
(eq *A* *B*)
(cons 2 *A*)
```

```
(car *A*)
(cdr *A*)
(nthcdr 2 *A*)
(push x *A*)
(subseq *A* index)
```

(return-from fonk deger)

Üretim Belleği ⇒ Kuralların tutulduğu yer (Kurallar)
Çalışan Bellek ⇒ Geçici depolama (olaylar)