

# Boxing vs Unboxing

ال **Boxing** = حط ال Value Type في صندوق (Object)  
ال **Unboxing** = اطلع ال Value Type من الصندوق

## Boxing

تحويل Value Type (int, float, bool) لـ Reference Type (object)

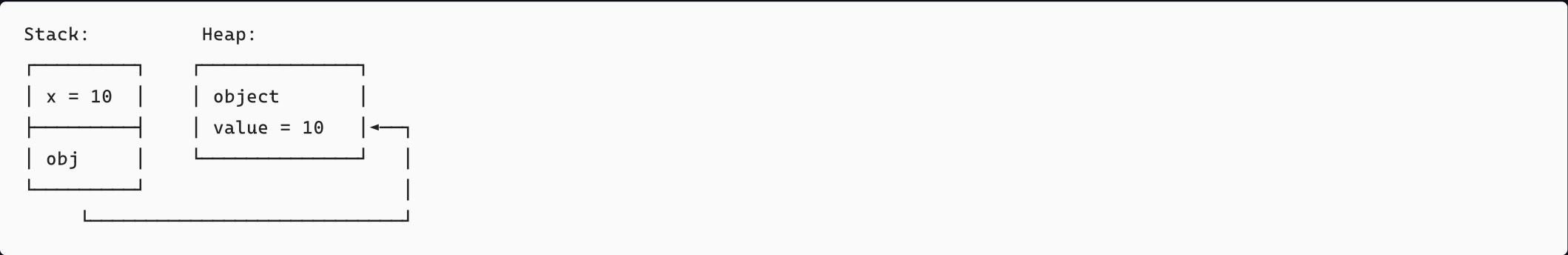
بيحصل إزاي؟

```
int x = 10;           // Value Type ال في Stack
object obj = x;       // Boxing! ال Heap اتنسخ لـ
```

إيه اللي بيحصل جوه؟

- 1. البرنامج يعمل object جديد في ال Heap
- 2. ينسخ قيمة (10) x جوه ال object

ده object يشاور على ال obj.



## Unboxing

تحويل Reference Type (object) لـ Value Type مرة ثانية

بيحصل إزاي؟

```
object obj = 10;      // أولاً Boxing
int y = (int)obj;     // Unboxing! استخرج القيمة
```

إيه اللي بيحصل جوه؟

- 1. البرنامج يتأكد إن ال object فيه int فعلاً
- 2. يستخرج القيمة من ال Heap
- 3. ينسخها في متغير جديد في ال Stack

# Upcasting vs Downcasting

## Upcasting:

تحويل object من child class لـ parent class

```
// Example
class Animal { public void Eat() { } }
class Dog : Animal { public void Bark() { } }


Dog myDog = new Dog();
Animal myAnimal = myDog; // Upcasting - آمن دائماً
```

⚠️ آمن تماماً - مش محتاج casting صريح


## Downcasting:

تحويل object من parent class لـ child class

```
Animal myAnimal = new Dog(); // Upcasting الأول
Dog myDog = (Dog)myAnimal;    // Downcasting - صريح casting محتاج
myDog.Bark(); // Bark() دلوطني تقدر تستعمل
```

 **خطر** - ممكن يرمي InvalidCastException

 لازم casting صريح

 محتاج تتأكد الأول بـ **is** أو **as**

## الطريقة الآمنة للـ Downcasting:

```
// الطريقة الأولى - as operator
Animal myAnimal = new Dog();
Dog myDog = myAnimal as Dog;
if (myDog != null) {
    myDog.Bark();
}

// الطريقة الثانية - is operator
if (myAnimal is Dog specificDog) {
    specificDog.Bark();
}
```