UNDERSTANDING BOXING AND UNBOXING IN C#

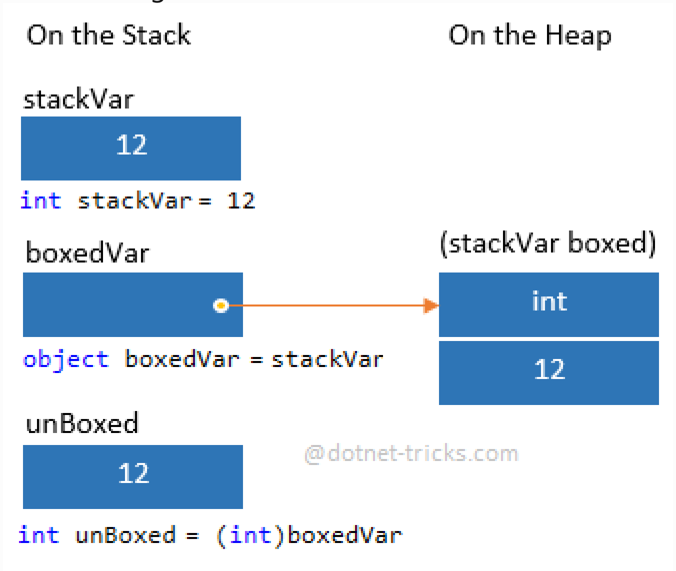
Boxing and unboxing are the most important concepts you always get asked in your interviews. Actually, it's really easy to understand, and simply refers to the allocation of a value type (e.g. int, char, etc.) on the heap rather than the stack.

Boxing

Implicit conversion of a value type (int, char etc.) to a reference type (object), is known as Boxing. In boxing process, a value type is being allocated on the heap rather than the stack.

Unboxing

Explicit conversion of same reference type (which is being created by boxing process); back to a value type is known as unboxing. In unboxing process, boxed value type is unboxed from the heap and assigned to a value type which is being allocated on the stack.



Boxing and unboxing

For Example

```
// int (value type) is created on the Stack
int stackVar = 12;

```

```
// Boxing = int is created on the Heap (reference type)
object boxedVar = stackVar;

// Unboxing = boxed int is unboxed from the heap and assigned to an int stack variable
int unBoxed = (int)boxedVar;
```

Real Life Example

```
int i = 10;
ArrayList arrlst = new ArrayList();

//ArrayList contains object type value
//So, int i is being created on heap
arrlst.Add(i); // Boxing occurs automatically

int j = (int)arrlst[0]; // Unboxing occurs
```

Note

1. Sometimes boxing is necessary, but you should avoided it if possible, since it will slow down the performance and increase memory requirements.
   For example, when a value type is boxed, a new reference type is created and the value is copied from the value type to the newly created reference type. This process takes time and required extra memory (around twice the memory of the original value type).
2. Attempting to unbox a null causes a NullReferenceException.
   ```
   int? stackVar = null;
   // Boxing= Integer is created on the Heap
   object boxedVar = stackVar;

   // NullReferenceException
   int unBoxed = (int)boxedVar; //Object reference not set to an instance of an object.
   ```
3. Attempting to unbox a reference to an incompatible value type causes an InvalidCastException.
   ```
   int stackVar = 12;
   // Boxing= Integer is created on the Heap
   ```

```
object boxedVar = stackVar;

// InvalidCastException
float unBoxed = (float)boxedVar; //Specified cast is not valid.
```

Source : http://www.dotnet-tricks.com/Tutorial/csharp/HL1W121013-Understanding-Boxing-and-Unboxing-in-C#.html