Prototype

Karoly Nyisztor
DEVELOPER
@knyisztor www.leakka.com

Overview

Motivation

AddressBook

Motivation

Avoid Expensive Initialization

- Create new objects by cloning existing ones
- Clones are independent objects

Value Types vs. Reference Types

- Value Types are copied upon assignment
- Classes must implement NSCopying

Prototype

The Prototype pattern creates new objects by copying a prototype object. Use this pattern if initializing an object is expensive.

Demo

AddressBook

- Prototype using value type
- Copy-on-write optimizations
- Prototype using reference type
- Deep copy vs. shallow copy
- AddressBook demo

Summary

Prototype

- Use it when initializing new instances of a type is expensive
- Create clones by copying all the properties of the prototype instance

Value Types vs. Reference Types

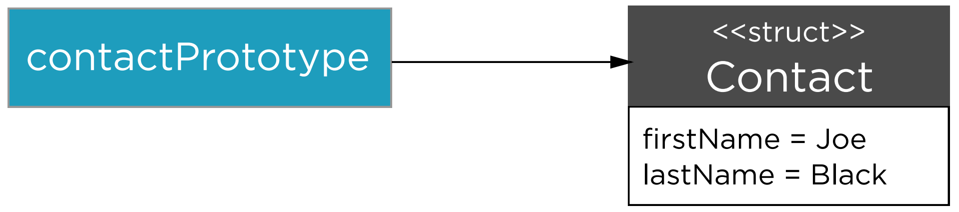
- Value types are automatically cloned
- Adopt NSCopying for classes

Shallow Copy vs. Deep Copy

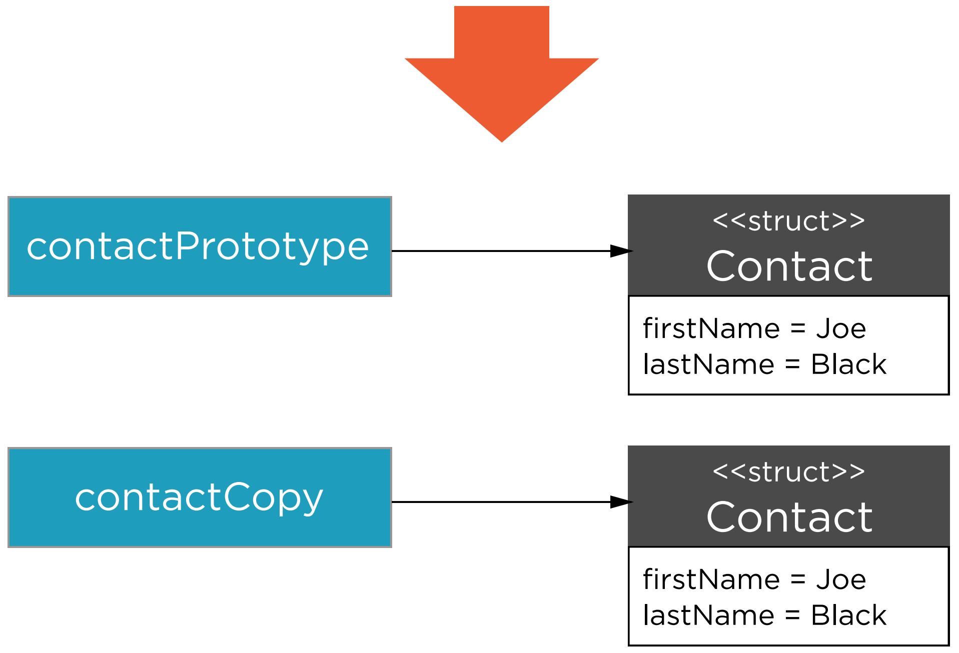
- Understand the difference
- Check the entire object hierarchy when cloning

Assigning Value Types

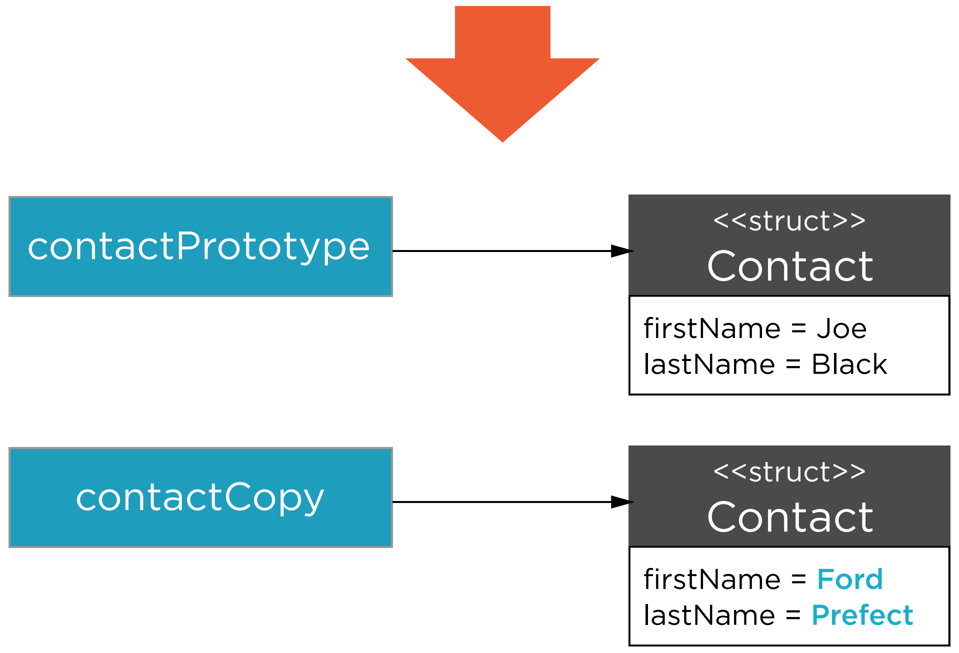
contactPrototype = Contact(firstName: "Joe", lastName: "Black")



contactCopy = contactPrototype

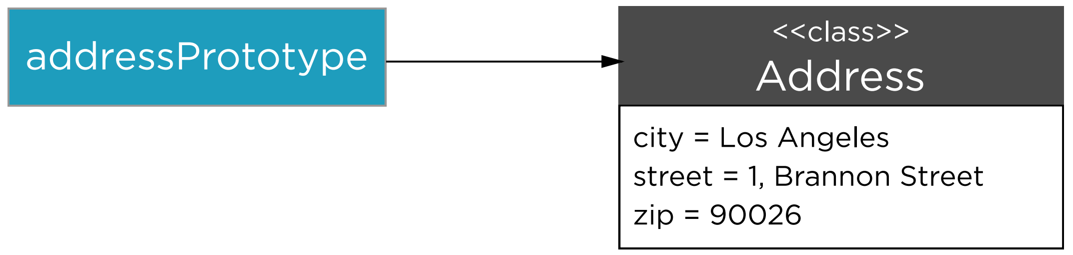


contactCopy.firstName = “Ford”
contactCopy.lastName = “Prefect”

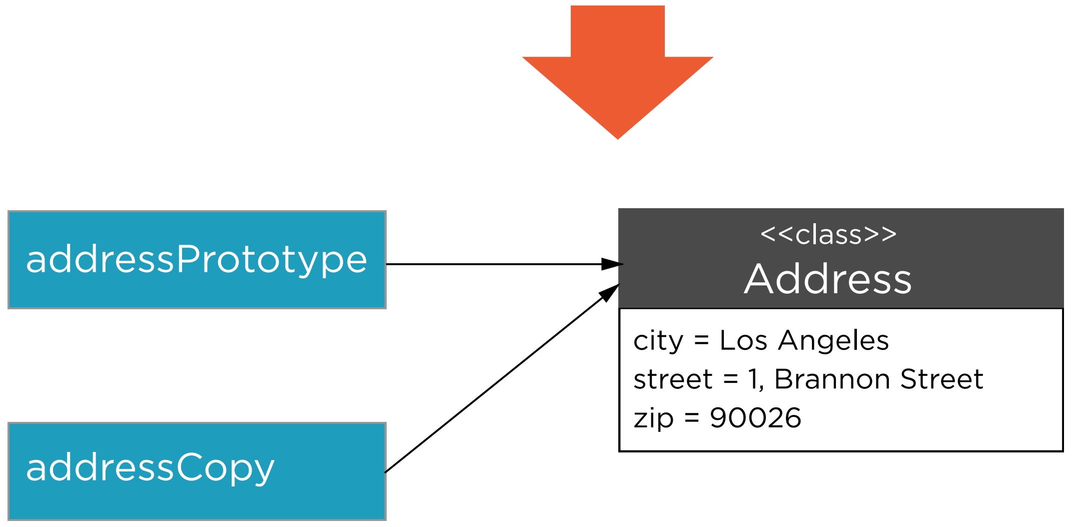


Assigning Reference Types

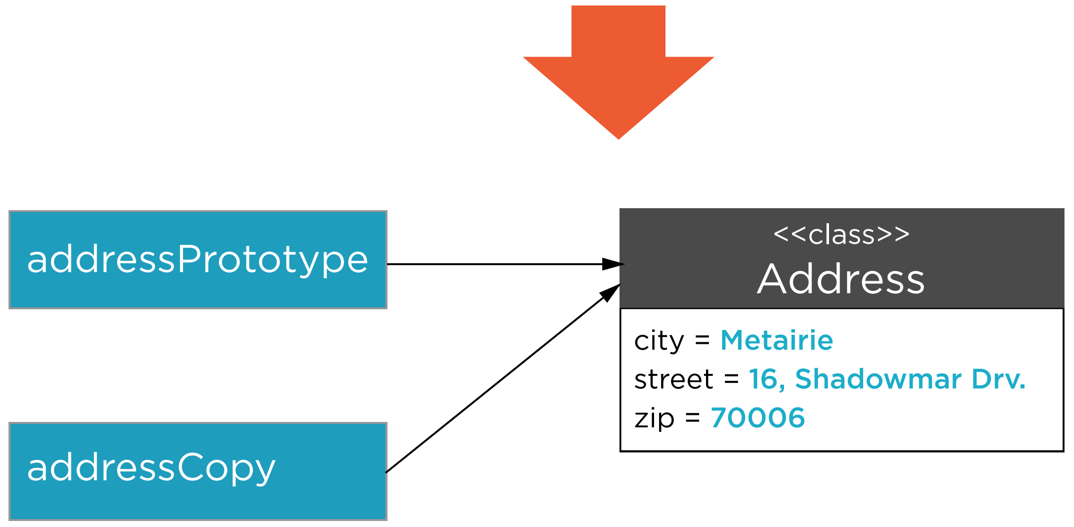
```
addressPrototype = Address(street: "1 Brannon Street", city: "Los Angeles", zip: "90026")
```



```
addressCopy = addressPrototype
```

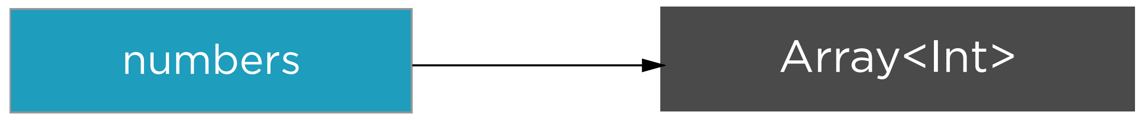


```
addressCopy.city = “Metairie”
contactCopy.street = “16, Shadowmar Drive”
contactCopy.zip = “70006”
```

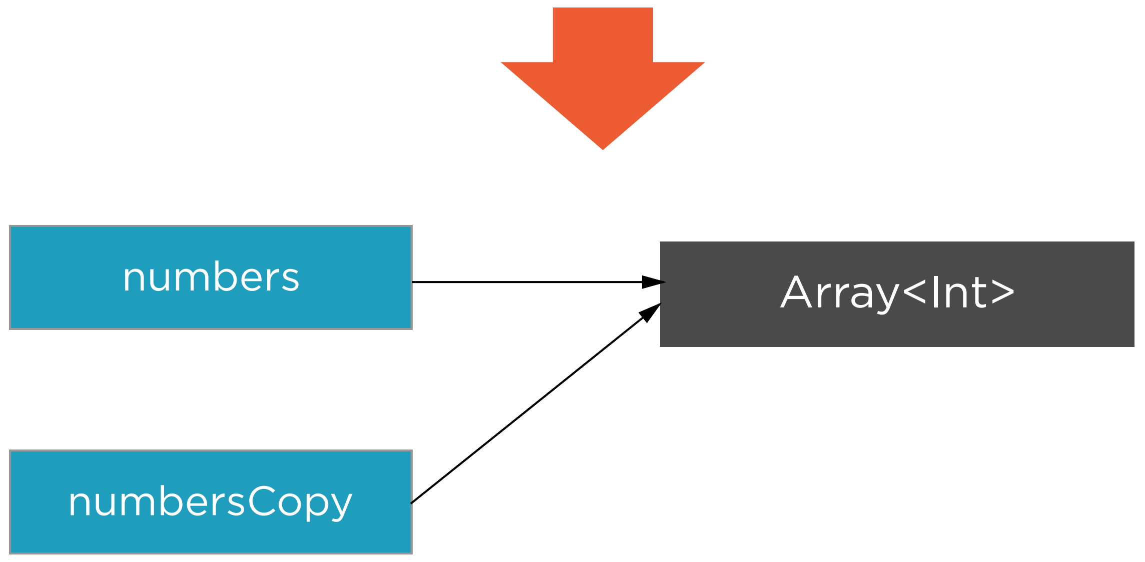


Copy-on-Write Optimization

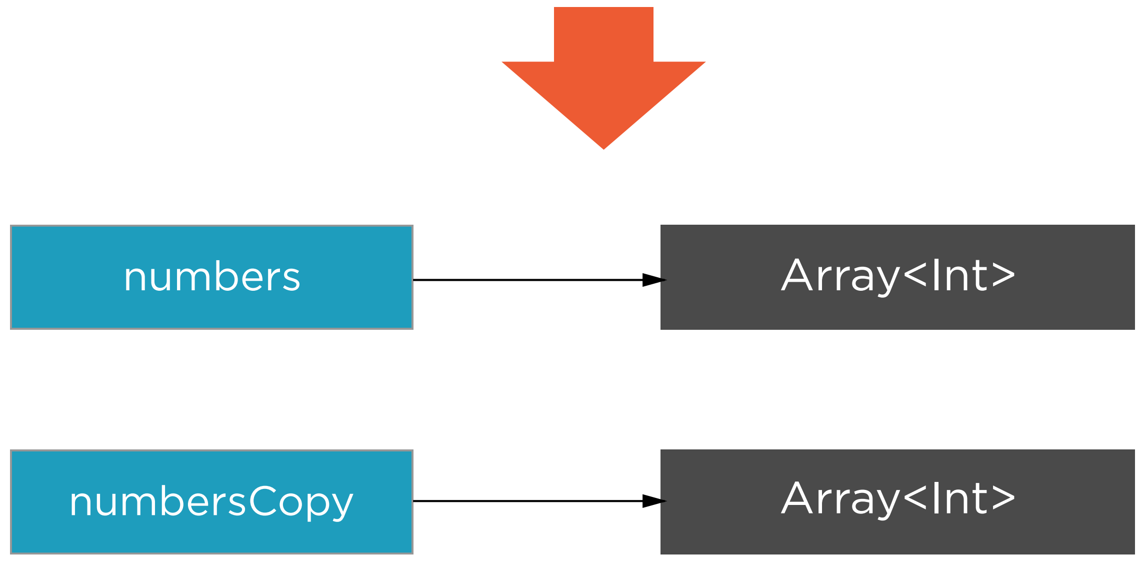
```
var numbers: Array<Int> = [1,2,3]
```



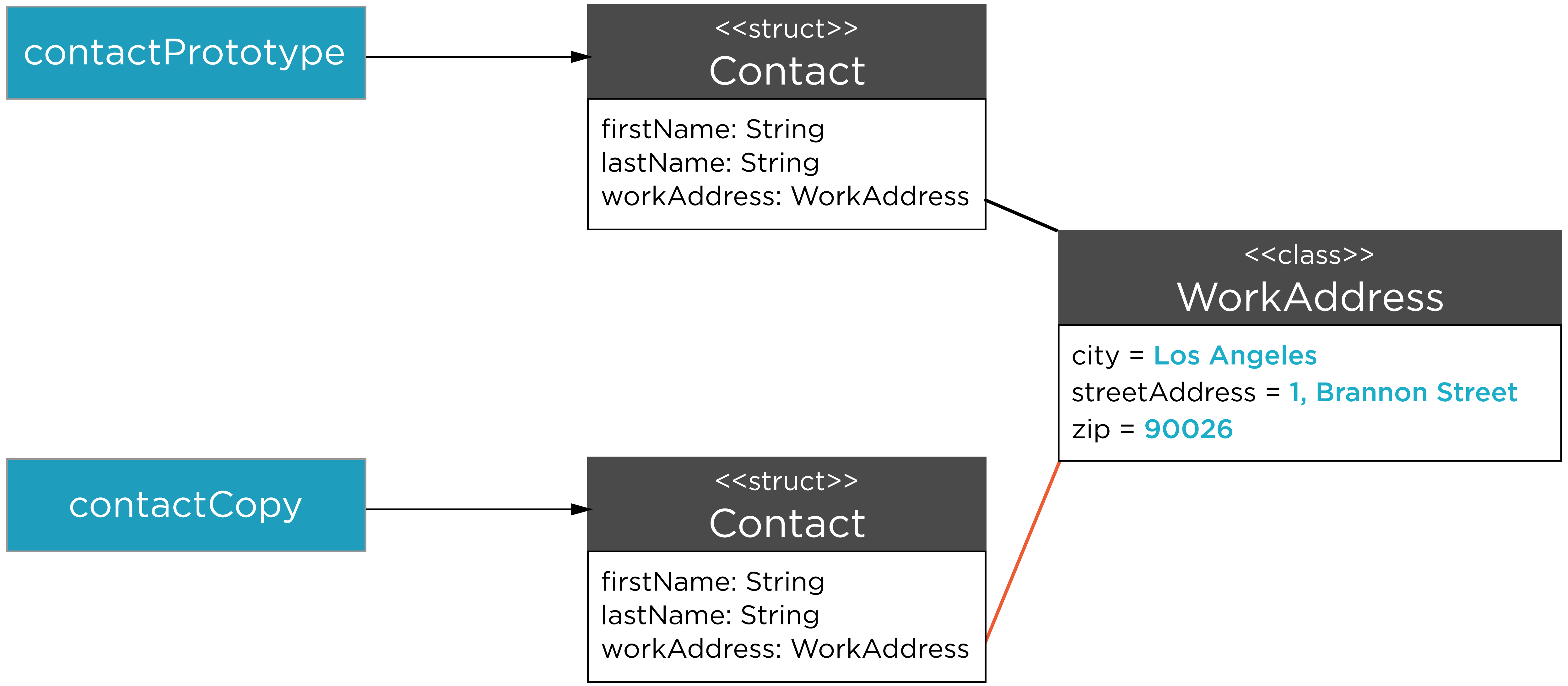
```
numbersCopy = numbers
```



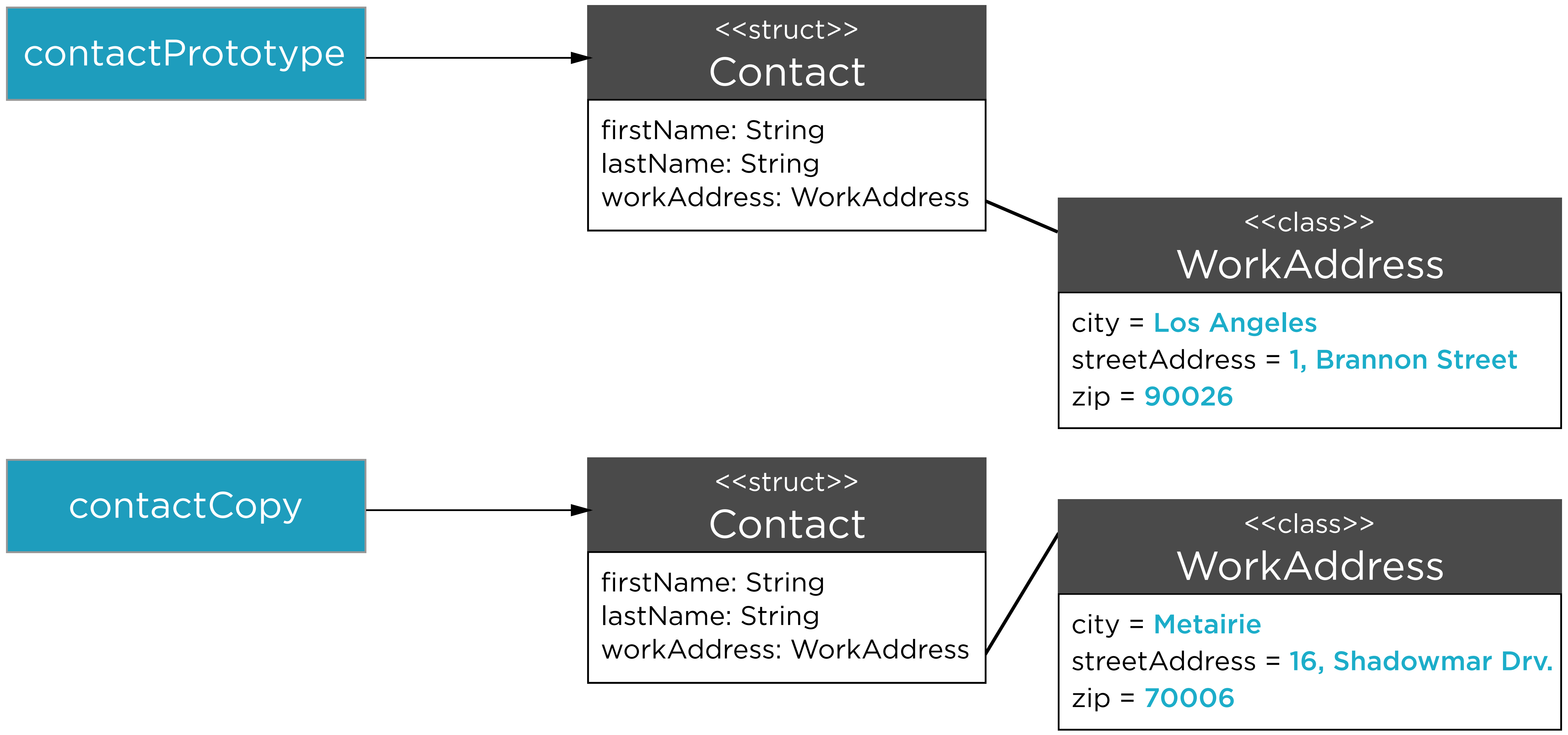
```
numbersCopy.append(4)
```



Shallow Copy



Deep Copy


contactPrototype
<<struct>>
Contact
firstName: String
lastName: String
workAddress: WorkAddress
<<class>>
WorkAddress
city = Los Angeles
streetAddress = 1, Brannon Street
zip = 90026
contactCopy
<<struct>>
Contact
firstName: String
lastName: String
workAddress: WorkAddress
<<class>>
WorkAddress
city = Metairie
streetAddress = 16, Shadowmar Drv.
zip = 70006