structs and classes

structs and classes

Way to design a reusable data model

Create the model once and generate copies of it

Define a set of characteristics and actions

Define a set of **variables** and **functions**

class

struct

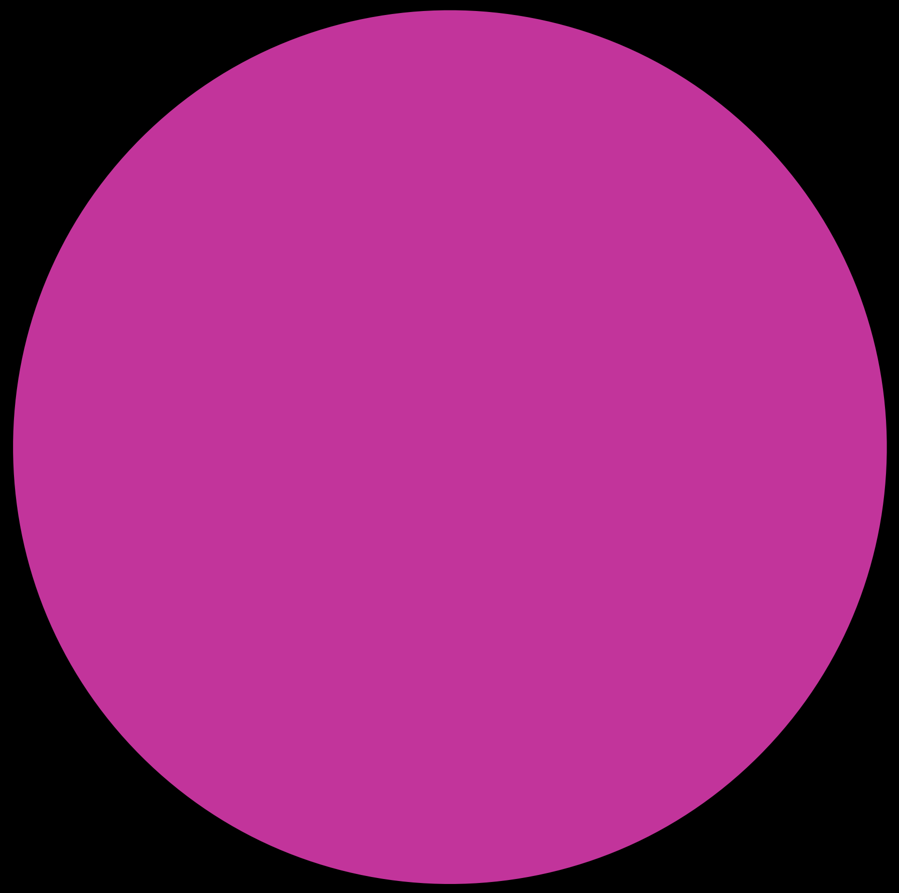
Model and instance

Model > define characteristics and actions

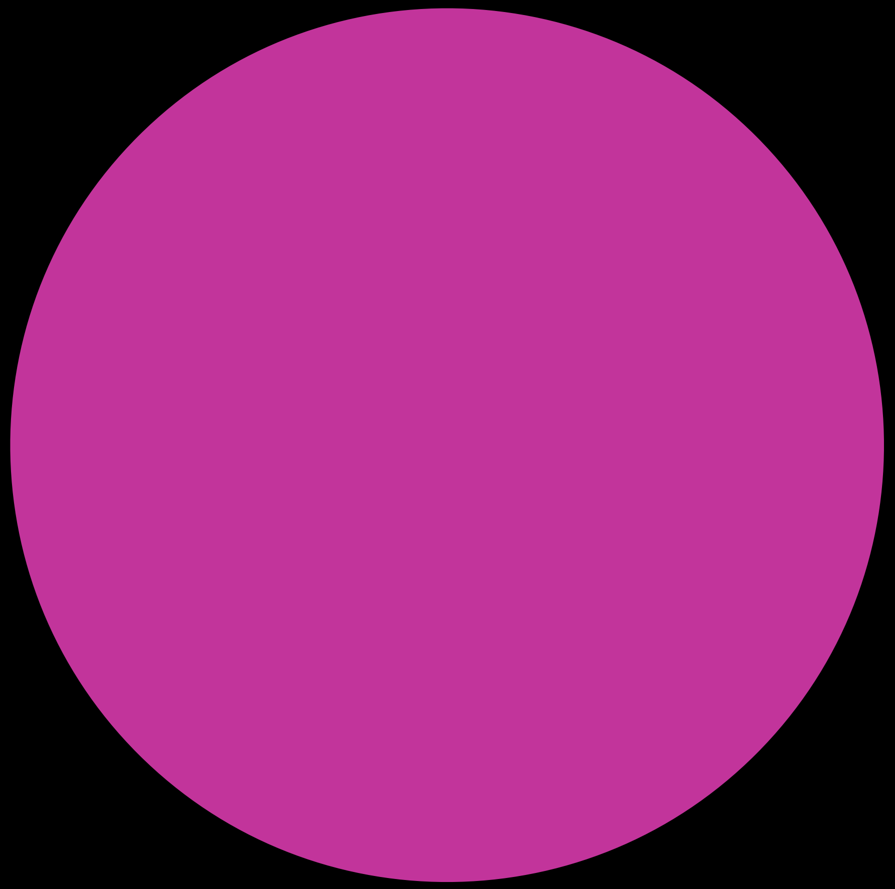
Instance > copy what was defined by the model

Any update on the Model affect its instances

Instances represent elements from the same "domain"

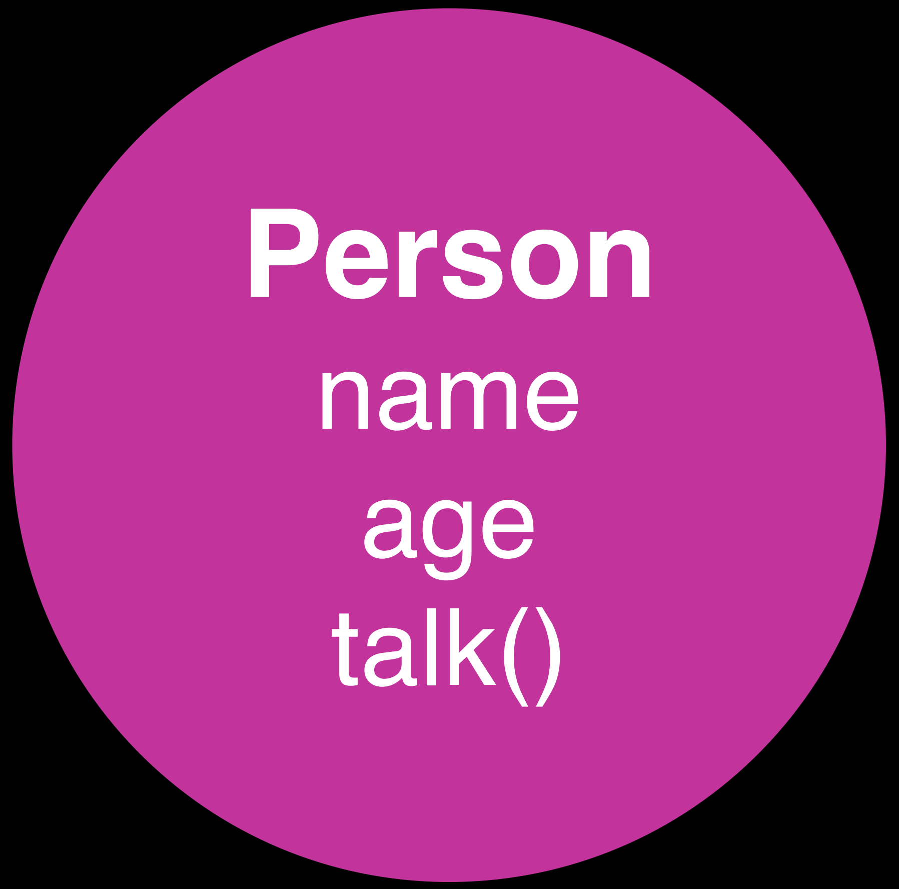


Model and instance

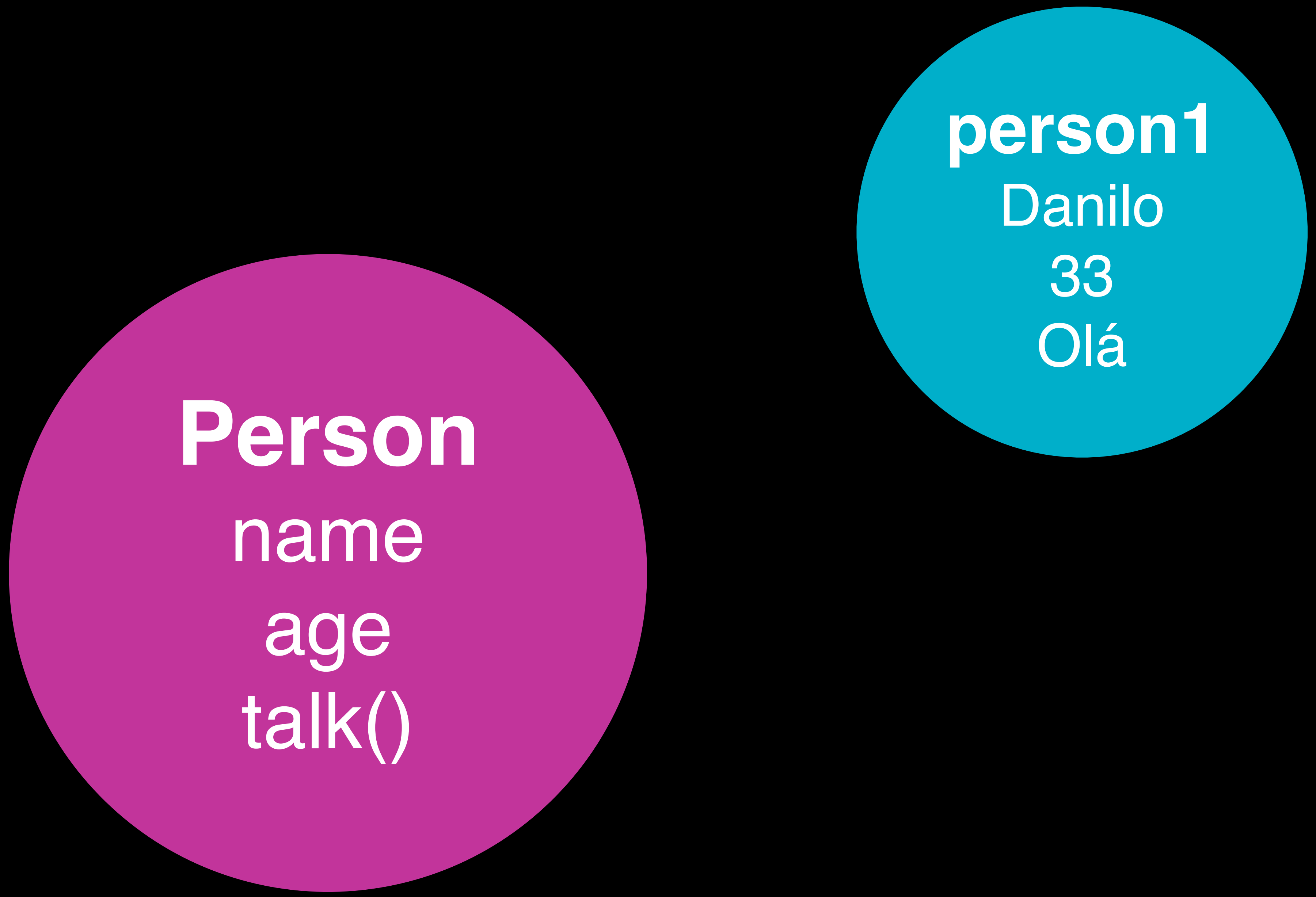


Model and instance

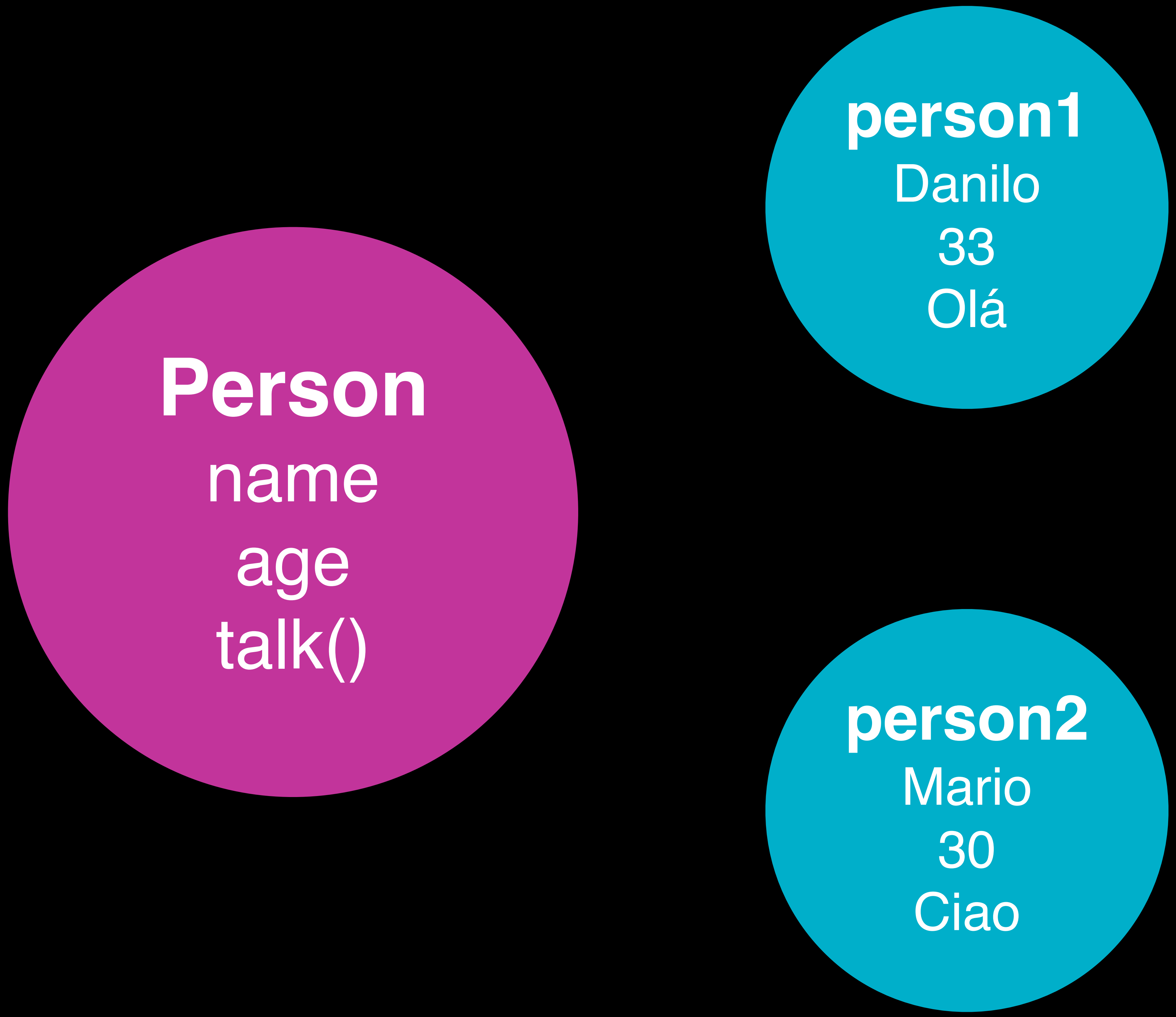
Model and instance



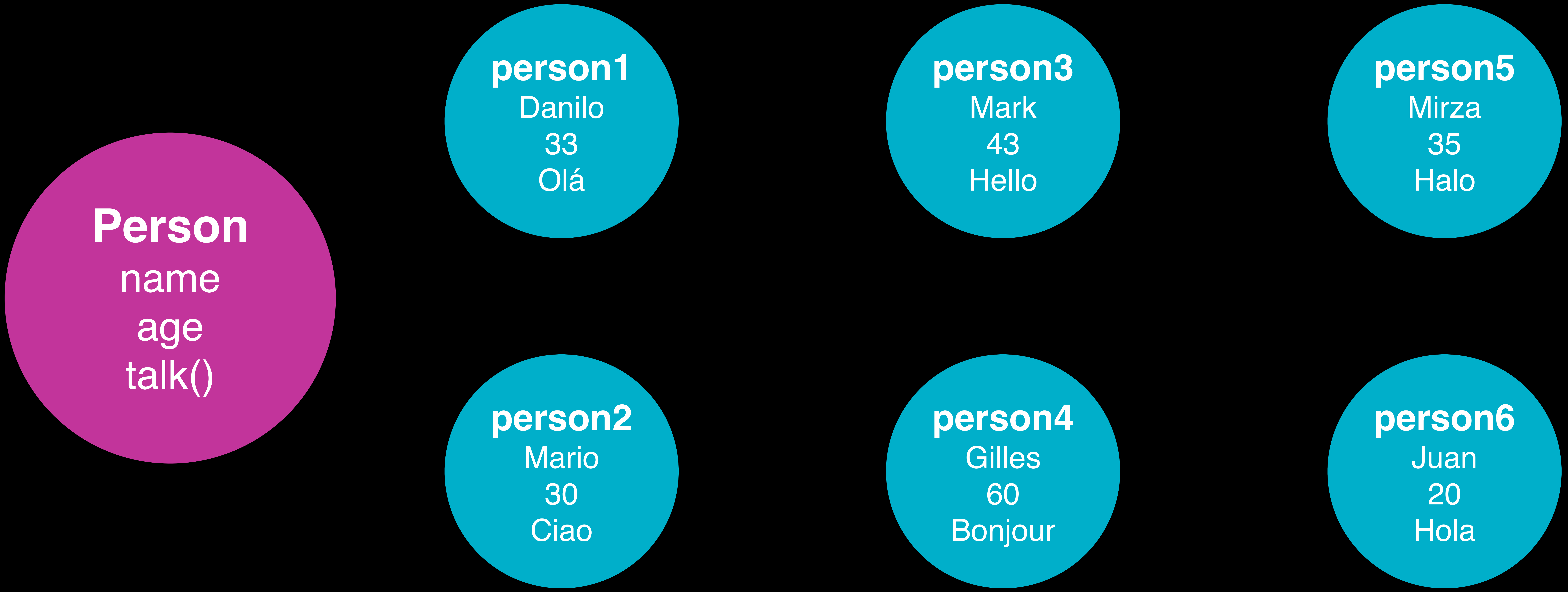
Model and instance



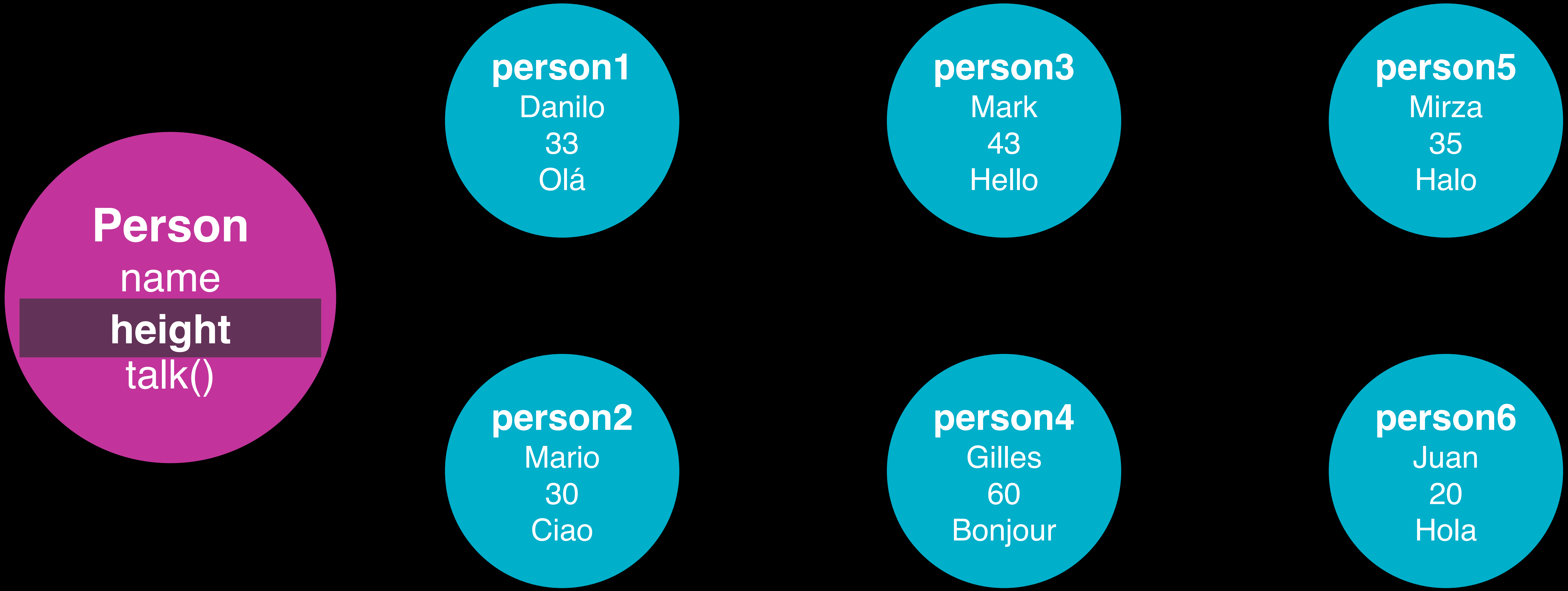
Model and instance



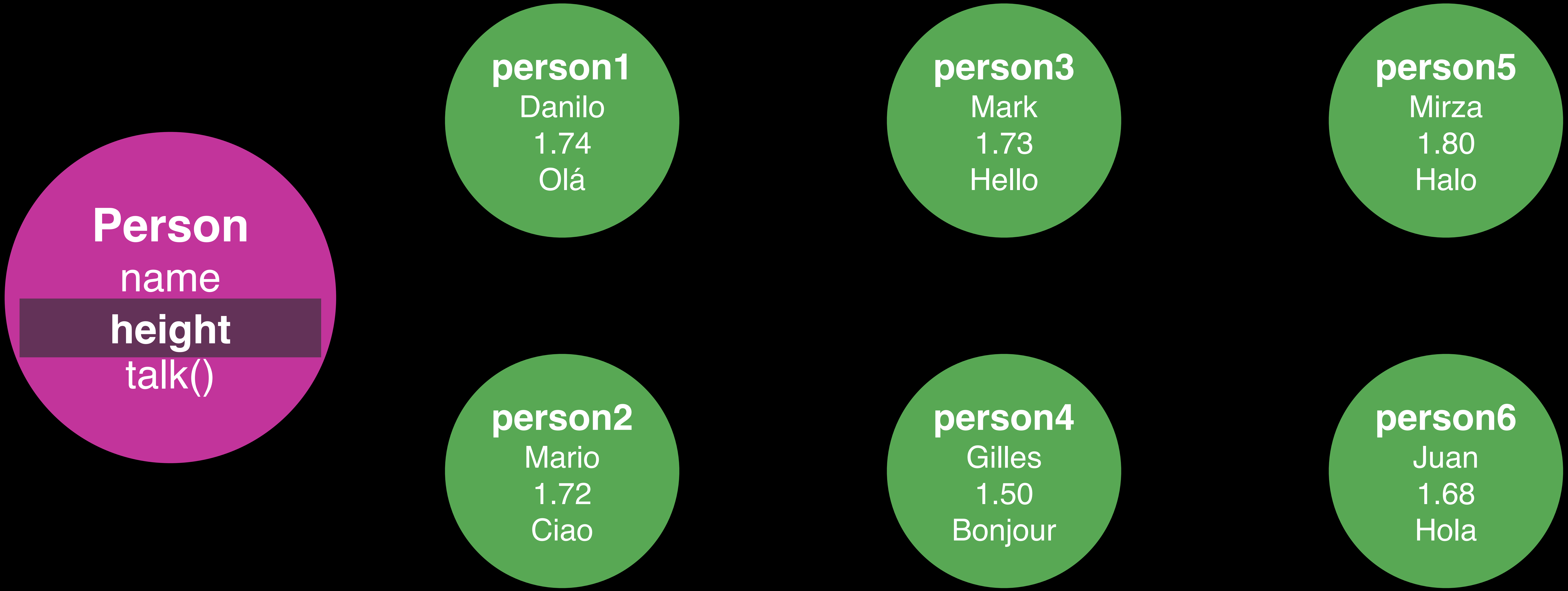
Model and instance



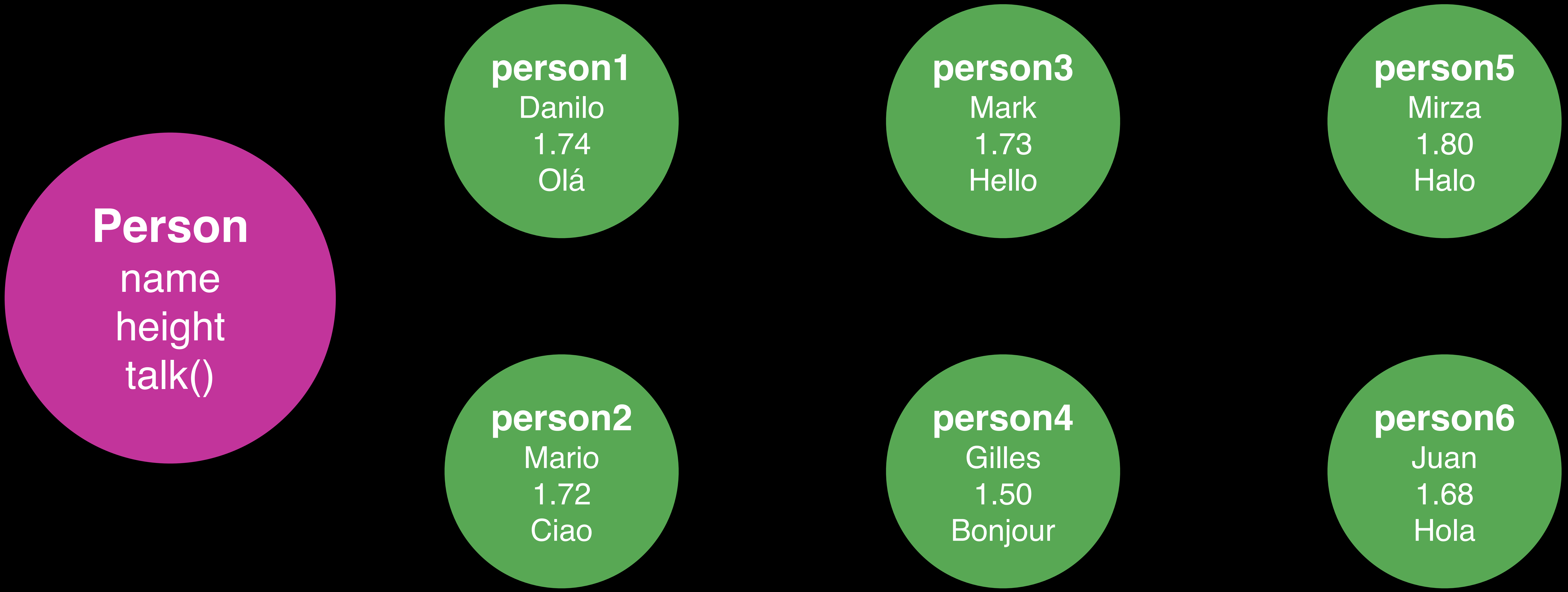
Model and instance



Model and instance



Model and instanc



struct

```swift
// Declaring the data model
struct Person {











}
```

```swift
// Declaring the data model
struct Person {










}
```

```swift
// Declaring the data model
struct Person {
    var name: String
    var talkText: String









}
```

```swift
// Declaring the data model
struct Person {
    var name: String
    var talkText: String









}
```

```swift
// Declaring the data model
struct Person {
    var name: String
    var talkText: String

    func talk(){
        print(talkText)
    }






}
```

```swift
// Declaring the data model
struct Person {
    var name: String
    var talkText: String

    func talk(){
        print(talkText)
    }




}
```

```swift
// Declaring the data model
struct Person {
    var name: String
    var talkText: String

    func talk(){
        print(talkText)
    }
    // Declaring the initializer
    init(name: String, talkText: String = "Olá") {
        self.name = name
        self.talkText = talkText
    }
}
```

```swift
// Declaring the data model
struct Person {
    var name: String
    var talkText: String

    func talk(){
        print(talkText)
    }
    // Declaring the initializer
    init(name: String, talkText: String = "Olá") {
        self.name = name
        self.talkText = talkText
    }
}
```

```swift
// Declaring the data model
struct Person {
    var name: String
    var talkText: String

    func talk(){
        print(talkText)
    }
    // Declaring the initializer
    init(name: String, talkText: String = "Olá") {
        self.name = name
        self.talkText = talkText
    }
}
```

```swift
// Declaring the data model
struct Person {
    var name: String
    var talkText: String

    func talk(){
        print(talkText)
    }
    // Declaring the initializer
    init(name: String, talkText: String = "Olá") {
        self.name = name
        self.talkText = talkText
    }
}
// Creating an instance of the data model
var eu: Person = Person(name: "Danilo")
```

```swift
// Declaring the data model
struct Person {
    var name: String
    var talkText: String

    func talk(){
        print(talkText)
    }
    // Declaring the initializer
    init(name: String, talkText: String = "Olá") {
        self.name = name
        self.talkText = talkText
    }
}
// Creating an instance of the data model
var eu: Person = Person(name: "Danilo")
```

```swift
// Declaring the data model
struct Person {
    var name: String
    var talkText: String

    func talk(){
        print(talkText)
    }
    // Declaring the initializer
    init(name: String, talkText: String = "Olá") {
        self.name = name
        self.talkText = talkText
    }
}
// Creating an instance of the data model
var eu: Person = Person(name: "Danilo")
```

```swift
// Declaring the data model
struct Person {
    var name: String
    var talkText: String

    func talk(){
        print(talkText)
    }
    // Declaring the initializer
    init(name: String, talkText: String = "Olá") {
        self.name = name
        self.talkText = talkText
    }
}
// Creating an instance of the data model
var eu: Person = Person(name: "Danilo")
```

```swift
// Declaring the data model
struct Person {
    var name: String
    var talkText: String

    func talk(){
        print(talkText)
    }
    // Declaring the initializer
    init(name: String, talkText: String = "Olá") {
        self.name = name
        self.talkText = talkText
    }
}
// Creating an instance of the data model
var eu: Person = Person(name: "Danilo")
```

```
struct Person {
    var name: String
    var talkText: String
    func talk(){  print(talkText)    }

    init(name: String, talkText: String = "Olá") {
        self.name = name
        self.talkText = talkText
    }
}

var eu: Person = Person(name: "Danilo")
eu.talk()

var tu: Person = Person(name: "Mark", talkText: "Hello")
tu.talk()
```

```swift
struct Person {
    var name: String
    var talkText: String
    func talk(){  print(talkText)    }

    init(name: String, talkText: String = "Olá") {
        self.name = name
        self.talkText = talkText
    }
}

var eu: Person = Person(name: "Danilo")
eu.talk()

var tu: Person = Person(name: "Mark", talkText: "Hello")
tu.talk()
```

```swift
struct Person {
    var name: String
    var talkText: String
    func talk(){  print(talkText)    }

    init(name: String, talkText: String = "Olá") {
        self.name = name
        self.talkText = talkText
    }
}

var eu: Person = Person(name: "Danilo")
eu.talk()                                           Olá

var tu: Person = Person(name: "Mark", talkText: "Hello")
tu.talk()
```

```
struct Person {
    var name: String
    var talkText: String
    func talk(){  print(talkText)    }

    init(name: String, talkText: String = "Olá") {
        self.name = name
        self.talkText = talkText
    }
}

var eu: Person = Person(name: "Danilo")
eu.talk()                                                Olá

var tu: Person = Person(name: "Mark", talkText: "Hello")
tu.talk()
```

```swift
struct Person {
    var name: String
    var talkText: String
    func talk(){  print(talkText)    }

    init(name: String, talkText: String = "Olá") {
        self.name = name
        self.talkText = talkText
    }
}

var eu: Person = Person(name: "Danilo")
eu.talk()                                             Olá

var tu: Person = Person(name: "Mark", talkText: "Hello")
tu.talk()                                             Hello
```

Hands on

class

Differentiating classes from structs

Classes have inheritance

Classes allow type casting

Classes can have deinitializers

Classes are **reference** type **structs** are **value** type

class

reference type

struct

value type

```swift
struct Dog {
    var name: String
    var isNice: Bool
}
```

```swift
struct Dog {
    var name: String
    var isNice: Bool
}
```

```swift
struct Dog {
    var name: String
    var isNice: Bool
}

var toto: Dog = Dog(name: "Sete", isNice: true)
var rex:  Dog = toto
```

```swift
struct Dog {
    var name: String
    var isNice: Bool
}

var toto: Dog = Dog(name: "Sete", isNice: true)
var rex:  Dog = toto
```

```swift
struct Dog {
    var name: String
    var isNice: Bool
}

var toto: Dog = Dog(name: "Sete", isNice: true)
var rex:  Dog = toto
```

```swift
struct Dog {
    var name: String
    var isNice: Bool
}

var toto: Dog = Dog(name: "Sete", isNice: true)
var rex:  Dog = toto

print(toto.name)
print(rex.name)
```

```swift
struct Dog {
    var name: String
    var isNice: Bool
}

var toto: Dog = Dog(name: "Sete", isNice: true)
var rex:  Dog = toto

print(toto.name)
print(rex.name)
```

```swift
struct Dog {
    var name: String
    var isNice: Bool
}

var toto: Dog = Dog(name: "Sete", isNice: true)
var rex:  Dog = toto

print(toto.name)                                         Sete
print(rex.name)                                          Sete
```

```
struct Dog {
    var name: String
    var isNice: Bool
}

var toto: Dog = Dog(name: "Sete", isNice: true)
var rex:  Dog = toto

print(toto.name)                                     Sete
print(rex.name)                                      Sete

toto.name = "Xuxa"

print(toto)
print(rex)
```

```swift
struct Dog {
    var name: String
    var isNice: Bool
}

var toto: Dog = Dog(name: "Sete", isNice: true)
var rex:  Dog = toto

print(toto.name)                                        Sete
print(rex.name)                                         Sete

toto.name = "Xuxa"

print(toto)
print(rex)
```

```swift
struct Dog {
    var name: String
    var isNice: Bool
}

var toto: Dog = Dog(name: "Sete", isNice: true)
var rex:  Dog = toto

print(toto.name)                                          Sete
print(rex.name)                                           Sete

toto.name = "Xuxa"

print(toto)
print(rex)
```

```
struct Dog {
    var name: String
    var isNice: Bool
}

var toto: Dog = Dog(name: "Sete", isNice: true)
var rex:  Dog = toto

print(toto.name)                                        Sete
print(rex.name)                                         Sete

toto.name = "Xuxa"

print(toto)                                             Xuxa
print(rex)                                              Sete
```

VALUE TYPE

```
struct Dog {
    var name: String
    var isNice: Bool
}

var toto: Dog = Dog(name: "Sete", isNice: true)
var rex:  Dog = toto

print(toto.name)                                       Sete
print(rex.name)                                        Sete

toto.name = "Xuxa"

print(toto)                                            Xuxa
print(rex)                                             Sete
```

```
class Dog {
    var name: String
    var isNice: Bool
}

var toto: Dog = Dog(name: "Sete", isNice: true)
var rex:  Dog = toto

print(toto.name)                                         Sete
print(rex.name)                                          Sete

toto.name = "Xuxa"

print(toto)                                              Xuxa
print(rex)                                               Xuxa
```

```
class Dog {
    var name: String
    var isNice: Bool
}

var toto: Dog = Dog(name: "Sete", isNice: true)
var rex:  Dog = toto

print(toto.name)                                        Sete
print(rex.name)                                         Sete

toto.name = "Xuxa"

print(toto)                                             Xuxa
print(rex)                                              Xuxa
```

```
class Dog {
    var name: String
    var isNice: Bool
}

var toto: Dog = Dog(name: "Sete", isNice: true)
var rex:  Dog = toto

print(toto.name)                                       Sete
print(rex.name)                                        Sete

toto.name = "Xuxa"

print(toto)                                            Xuxa
print(rex)                                             Xuxa
```

REFERENCE TYPE

```
class Dog {
    var name: String
    var isNice: Bool
}

var toto: Dog = Dog(name: "Sete", isNice: true)
var rex:  Dog = toto

print(toto.name)                                        Sete
print(rex.name)                                         Sete

toto.name = "Xuxa"

print(toto)                                             Xuxa
print(rex)                                              Xuxa
```

VALUE TYPE

REFERENCE TYPE

VALUE TYPE

REFERENCE TYPE

Instance 1

```
var toto = "Sete"
```

VALUE TYPE

REFERENCE TYPE

Instance 1

```
var toto = "Sete"
```

Instance 2

VALUE TYPE

REFERENCE TYPE

Instance 1 `var toto = "Sete"`

Instance 2 `var rex = toto`

Create

a copy

VALUE TYPE

Instance 1

```
var toto = "Sete"
```

Instance 2

```
var rex = toto
```

Create

a copy

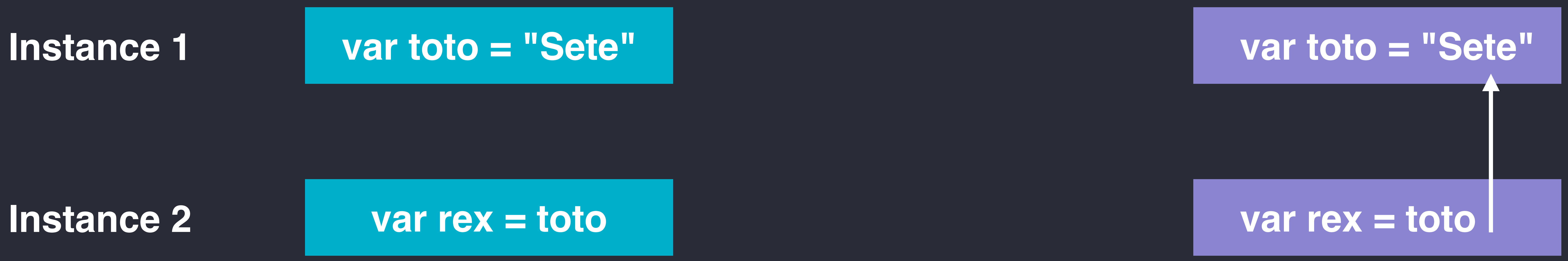
REFERENCE TYPE

```
var toto = "Sete"
```

	VALUE TYPE	REFERENCE TYPE
Instance 1	var toto = "Sete"	var toto = "Sete"
Instance 2	var rex = toto	var rex = toto
	Create a copy	

VALUE TYPE

REFERENCE TYPE

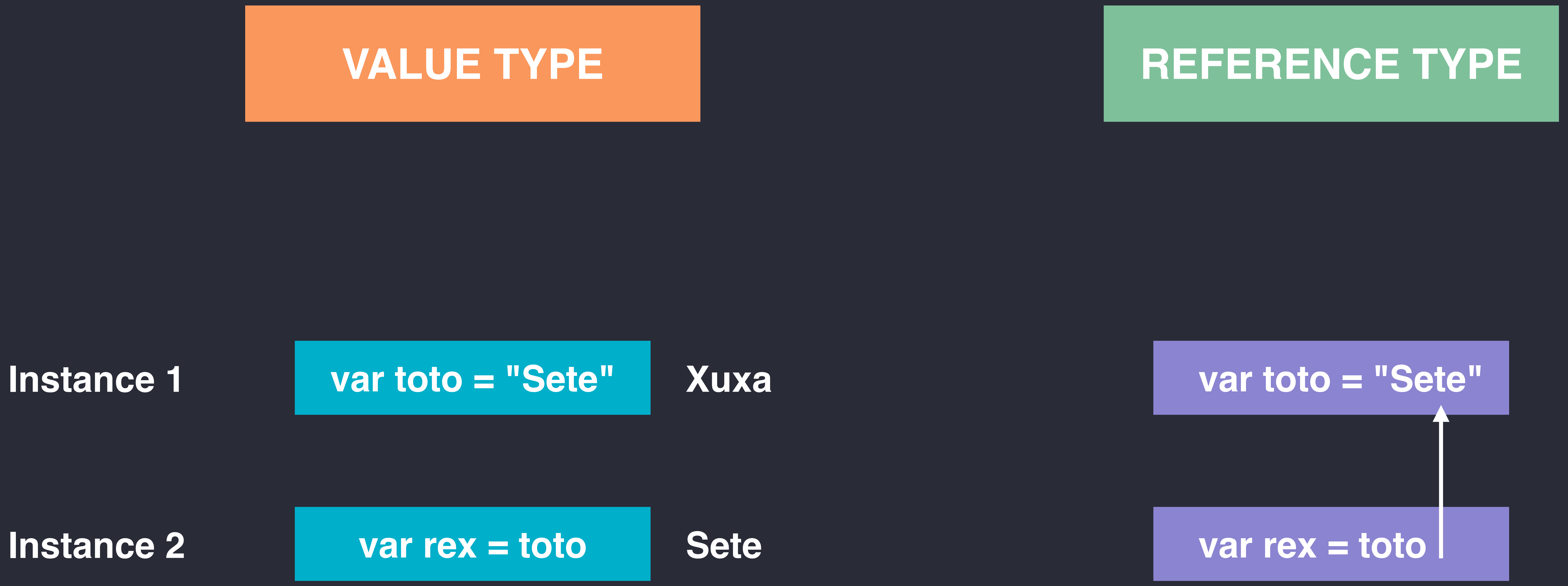


Create

a copy

Point to

the original



Create

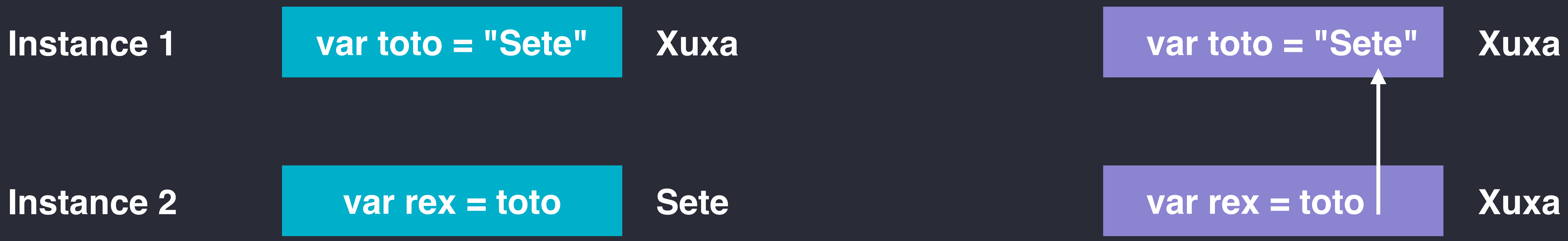
a copy

Point to

the original

VALUE TYPE

REFERENCE TYPE



Create

a copy

Point to

the original

Hands on