# Week 2 : Session 2

## OOP's Classes and Object

# Exercise - 5

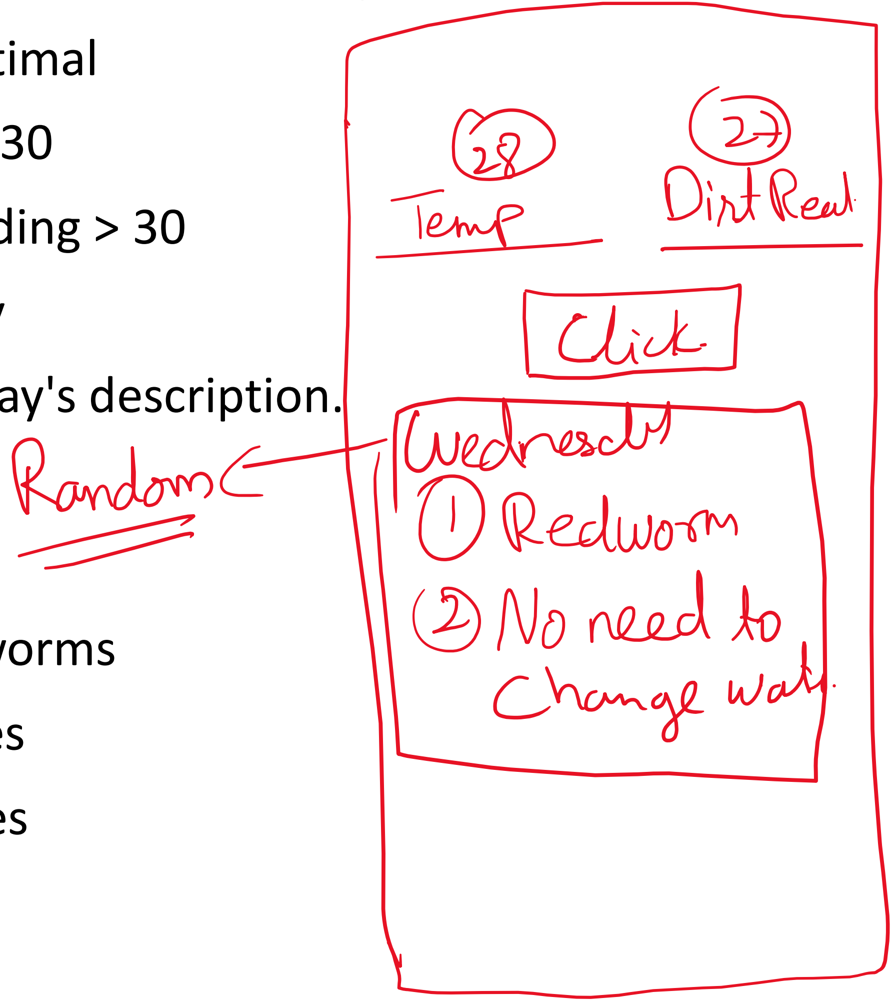
Example : Create a program in kotlin to help Bob, what food should he feed to fishes in the aquarium on particular day and does he need to change the water

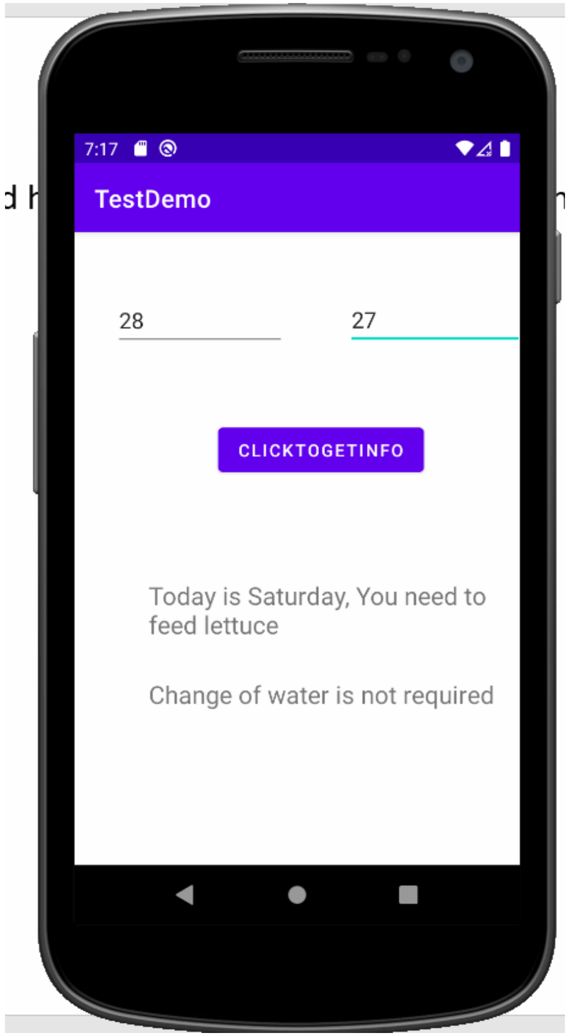
To change the water optimal

- temperature > 30
- dirt sensor reading > 30
- if day is Sunday

Display random single day's description.

- Monday -> flakes
- Tuesday -> pellets
- Wednesday -> redworms
- Thursday -> granules
- Friday -> mosquitoes
- Saturday -> lettuce
- Sunday -> plankton





Today is Saturday, You need to feed lettuce

Change of water is not required

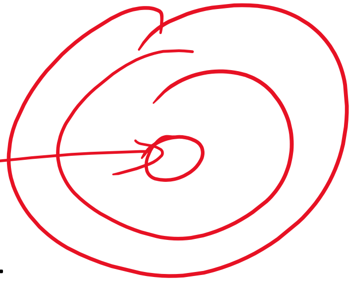
# Agenda

- Object-oriented Programming (OOP)
- Kotlin Class
  - Structure of a class.
  - Data members and function/method
    - Getter and setter methods
    - Method/function Overloading
  - Constructors
    - Primary Constructor
    - Parametrized Constructor
  - Initializers
- Exercise 6

# Object-oriented Programming (OOP)

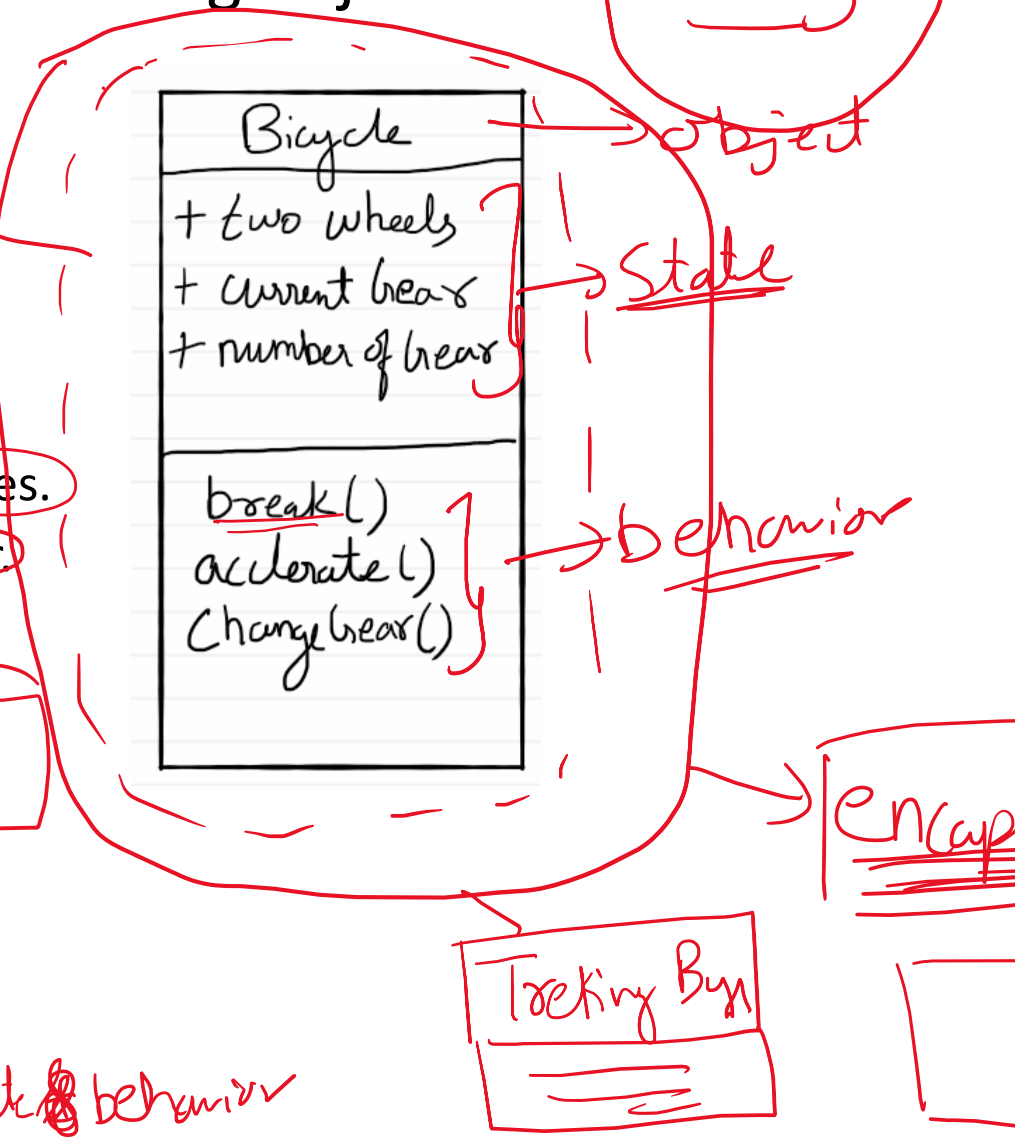
Real time

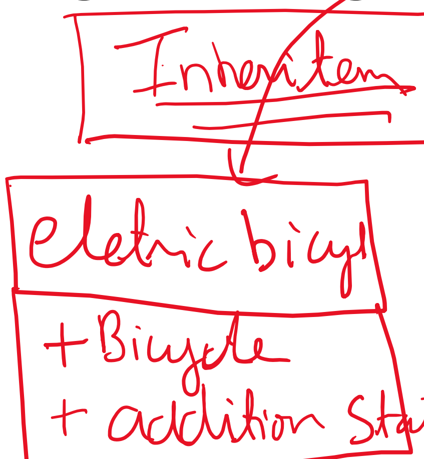
- Divide a complex problem into smaller sets by creating objects.
- These objects share two characteristics:
  - state
  - behavior
  - E.g
    - Bicycle is an object
    - It has current gear, two wheels, number of gear etc. states.
    - It has braking, accelerating, changing gears etc. behavior.
- Features of an object-oriented programming
  - *Data encapsulation* → security
  - *Inheritance* → Reusability
  - *Polymorphism* → many form implem



Object

Pen → write on pa
→ pointe
→ play as darts

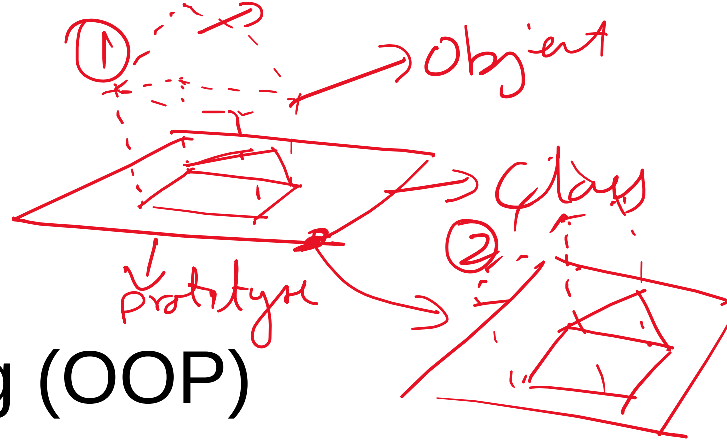




# Kotlin Class and Objects

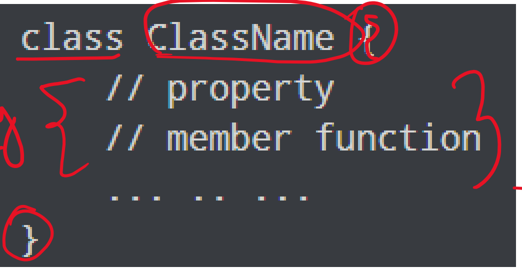
Same
- State
- Behavior

Differ → id



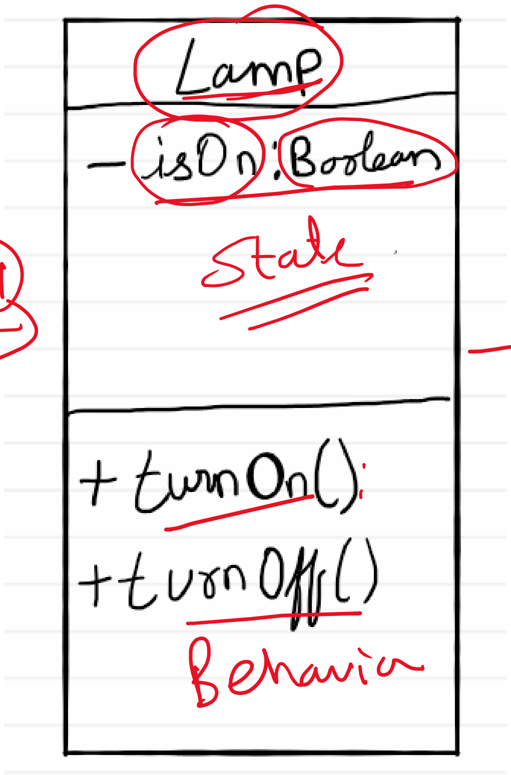
- A class is a blueprint for the object(sketch (prototype)).
- Basics component of Object-Oriented Programming (OOP)
- Kotlin also allows to create several objects of a class and call its members based on visibility.

Syntax



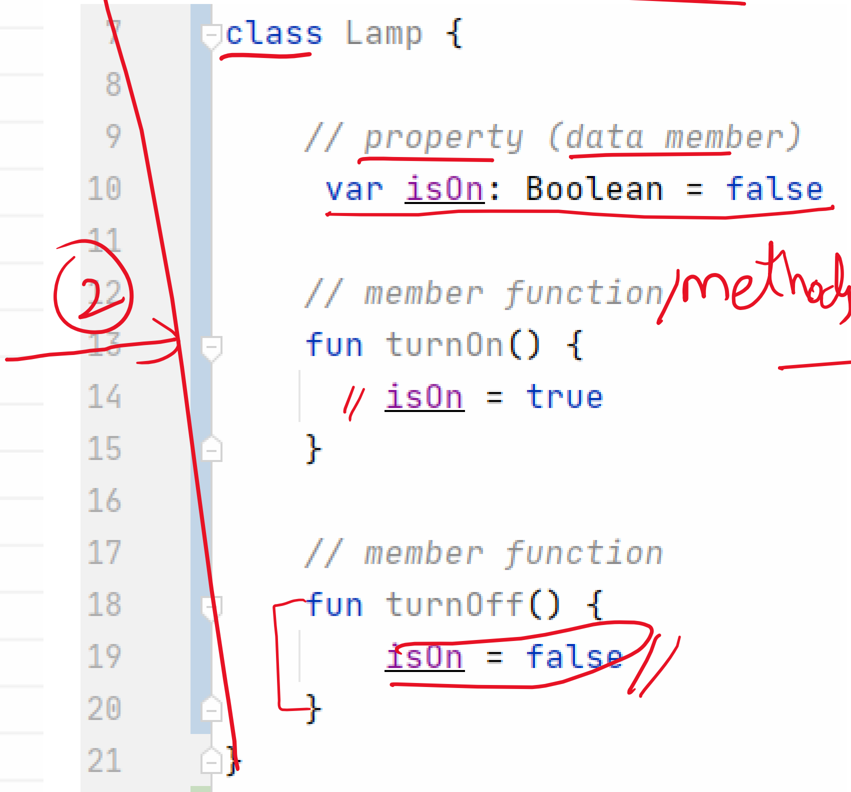
```
class ClassName {
    // property
    // member function
    ... .. ...
}
```

1



2

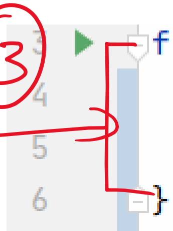
Blueprint



```
class Lamp {

    // property (data member)
    var isOn: Boolean = false

    // member function
    fun turnOn() {
        isOn = true
    }

    // member function
    fun turnOff() {
        isOn = false
    }
}
```

method

3



```
fun main(){
    val lamp = Lamp()
    print(lamp.isOn)
}
```

val lamp1 = Lamp()
var lamp2 = Lamp()

lamp.turnOff

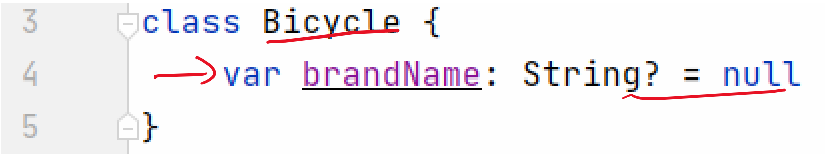
Object

Creating object
- instance
- object

# Kotlin Getters and Setters

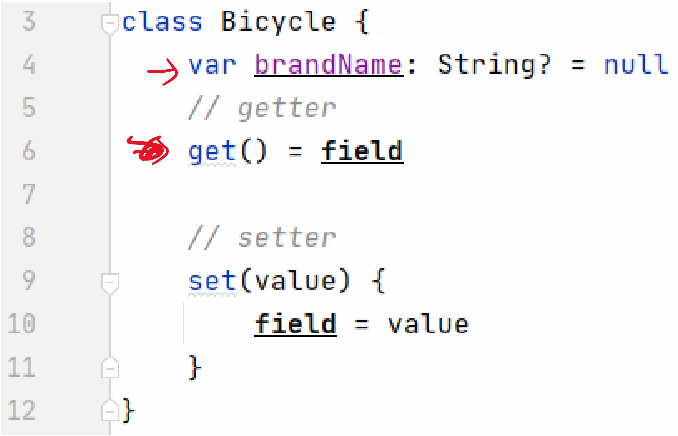
- Getters are used for getting value of the property
- Setters are used for setting value of the property

code in Kotlin



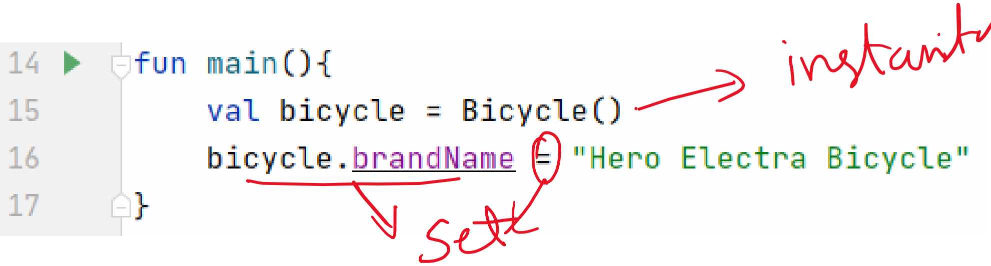
```
class Bicycle {
    var brandName: String? = null
}
```

equivalent code created in background



```
class Bicycle {
    var brandName: String? = null
    // getter
    get() = field

    // setter
    set(value) {
        field = value
    }
}
```

When we instantiate object of the Bicycle class and initialize the brandName property, it is passed to the setters parameter value and sets field to value.

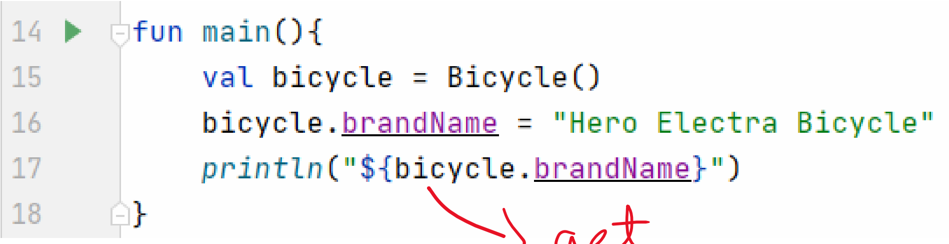


```
fun main(){
    val bicycle = Bicycle()
    bicycle.brandName = "Hero Electra Bicycle"
}
```

instantiate

Set

When you access name property of the object, you will get field because of the code get() = field.

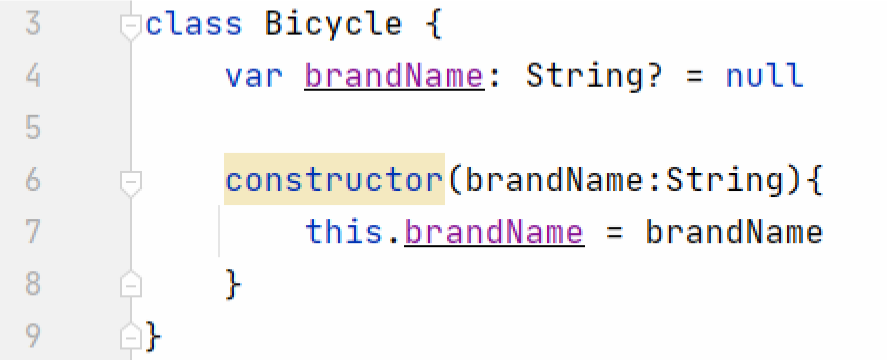


```
fun main(){
    val bicycle = Bicycle()
    bicycle.brandName = "Hero Electra Bicycle"
    println("${bicycle.brandName}")
}
```

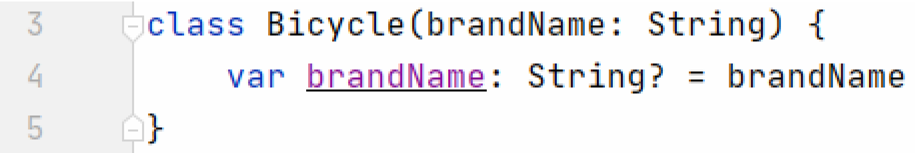
get

# Kotlin Constructor

- A constructor is a concise way to initialize class properties.
- It is a special member function that is called when an object is instantiated ( or created)
- Syntax

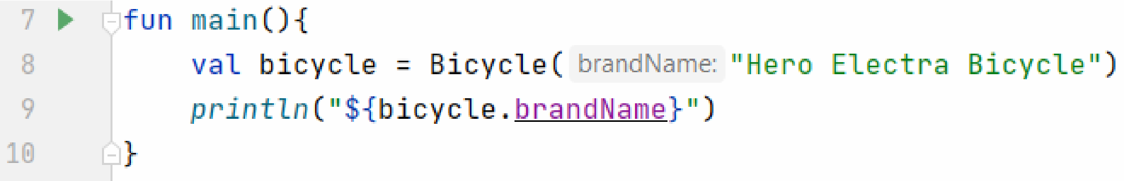


```kotlin
class Bicycle {
    var brandName: String? = null

    constructor(brandName:String){
        this.brandName = brandName
    }
}
```



```kotlin
class Bicycle(brandName: String) {
    var brandName: String? = brandName
}
```

- There are two types of constructors:
  - Primary constructor - concise way to initialize a class
    - The primary constructor is part of the class header.
    - The primary constructor has a constrained syntax, and cannot contain any code.

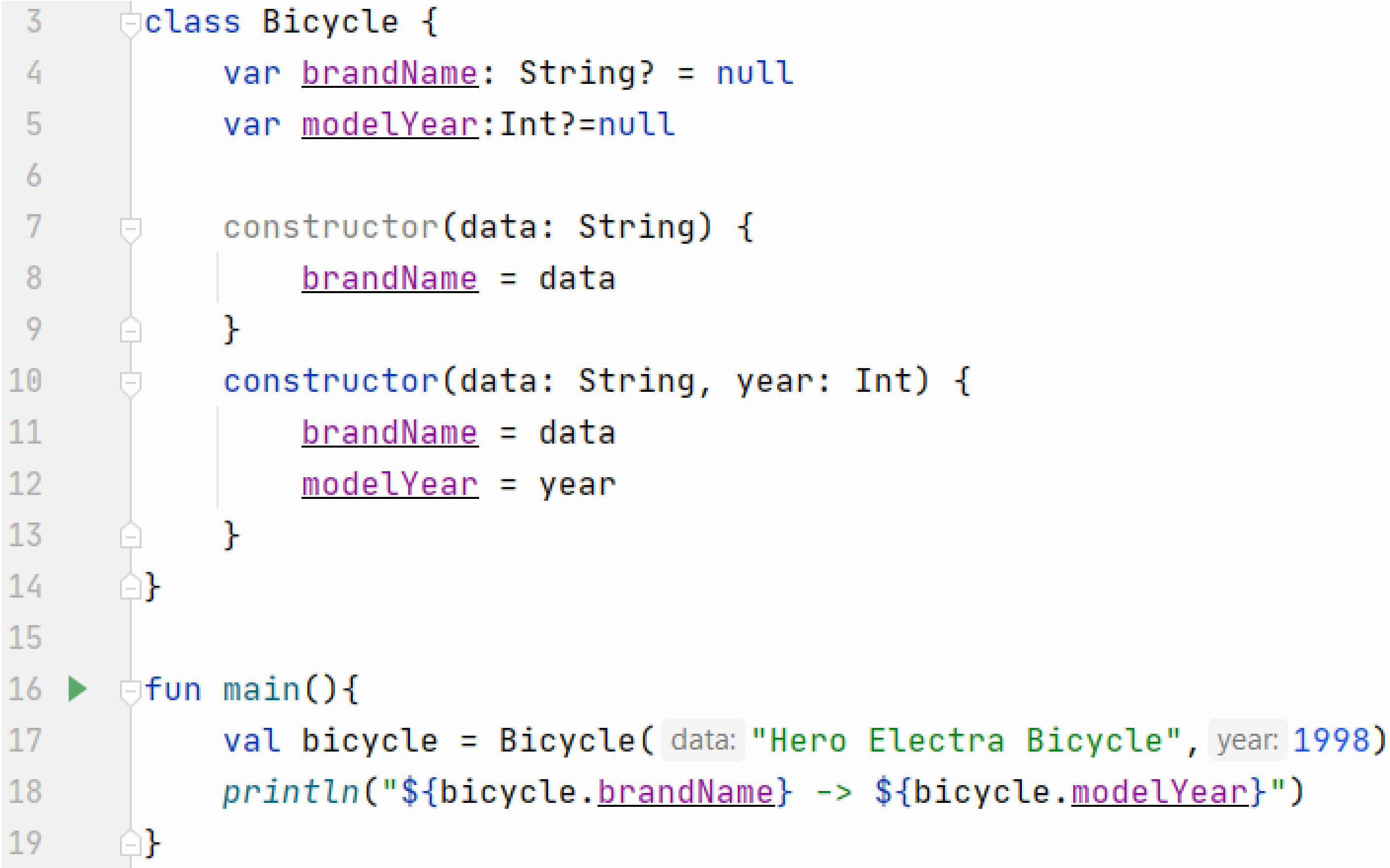


```kotlin
fun main(){
    val bicycle = Bicycle( brandName: "Hero Electra Bicycle")
    println("${bicycle.brandName}")
}
```

  - Secondary constructor - allows you to put additional initialization logic
    - These extend a class that provides multiple constructors that initialize the class in different ways.

# Kotlin Constructor

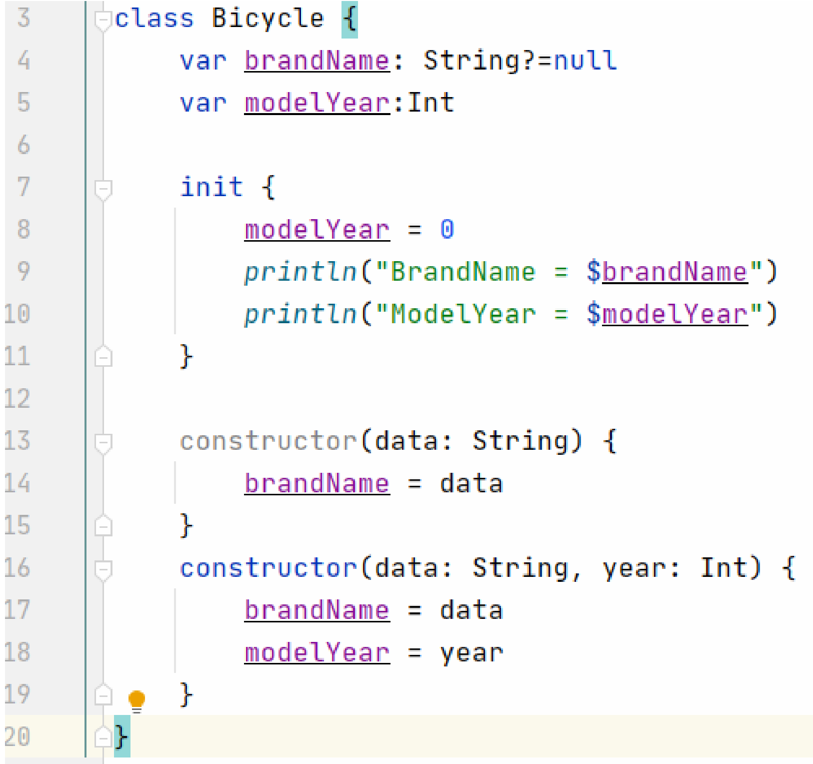
- Secondary constructor - allows you to put additional initialization logic
  - These extend a class that provides multiple constructors that initialize the class in different ways.



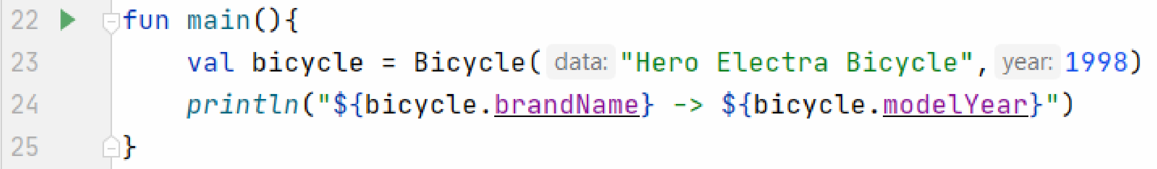
```kotlin
class Bicycle {
    var brandName: String? = null
    var modelYear:Int?=null

    constructor(data: String) {
        brandName = data
    }
    constructor(data: String, year: Int) {
        brandName = data
        modelYear = year
    }
}

fun main(){
    val bicycle = Bicycle( data: "Hero Electra Bicycle", year: 1998)
    println("${bicycle.brandName} -> ${bicycle.modelYear}")
}
```

# Kotlin initializer block

- The Initilization of data member can also be done using initializer block.
- It is prefixed with init keyword.



```kotlin
class Bicycle {
    var brandName: String?=null
    var modelYear:Int

    init {
        modelYear = 0
        println("BrandName = $brandName")
        println("ModelYear = $modelYear")
    }

    constructor(data: String) {
        brandName = data
    }
    constructor(data: String, year: Int) {
        brandName = data
        modelYear = year
    }
}
```



```kotlin
fun main(){
    val bicycle = Bicycle( data: "Hero Electra Bicycle", year: 1998)
    println("${bicycle.brandName} -> ${bicycle.modelYear}")
}
```

- When bicycle object is created, code inside initializer block is executed.
- The initializer block not only initializes its properties but also prints them.

# Kotlin Getters and Setters Demo

displa adialog

- Create an app in kotlin to validate person information and throw exception if person is minor.

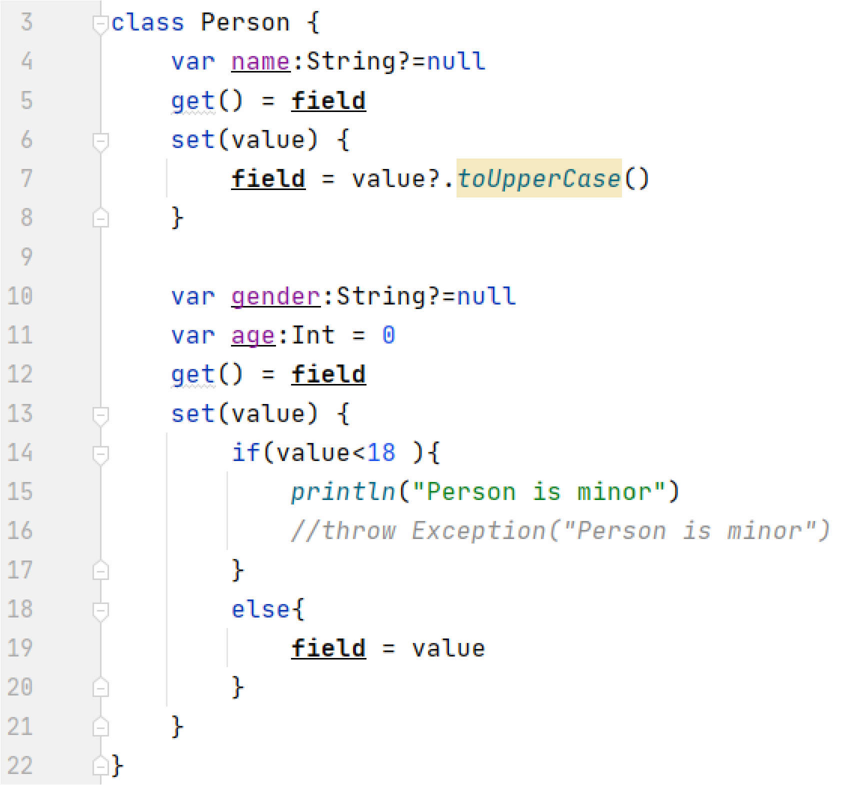
18

- Person information
  - Name
  - Gender
  - Age
- Validate Age and display dialog with
  - If valid -> Name in Upper cases
  - If invalid -> Person is Minor

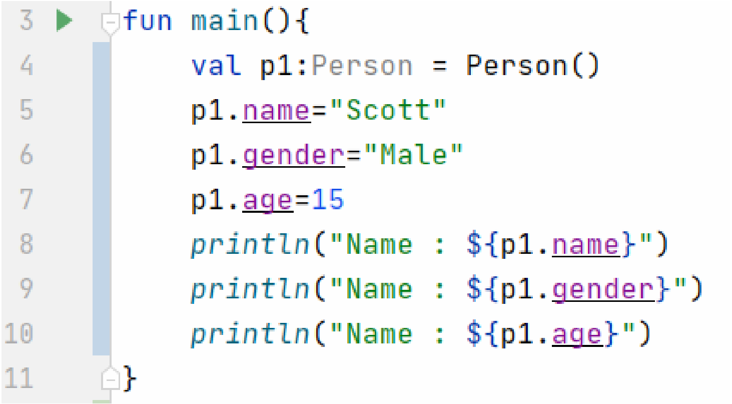
# Kotlin Getters and Setters

- Create an app in kotlin to validate person information and throw exception if person is minor.

  - Person information
    - Name
    - Gender
    - Age
  - Validate Age and display dialog with
    - If valid -> Name in Upper cases
    - If invalid -> Person is Minor



```kotlin
class Person {
    var name:String?=null
    get() = field
    set(value) {
        field = value?.toUpperCase()
    }

    var gender:String?=null
    var age:Int = 0
    get() = field
    set(value) {
        if(value<18 ){
            println("Person is minor")
            //throw Exception("Person is minor")
        }
        else{
            field = value
        }
    }
}
```



```kotlin
fun main(){
    val p1:Person = Person()
    p1.name="Scott"
    p1.gender="Male"
    p1.age=15
    println("Name : ${p1.name}")
    println("Name : ${p1.gender}")
    println("Name : ${p1.age}")
}
```