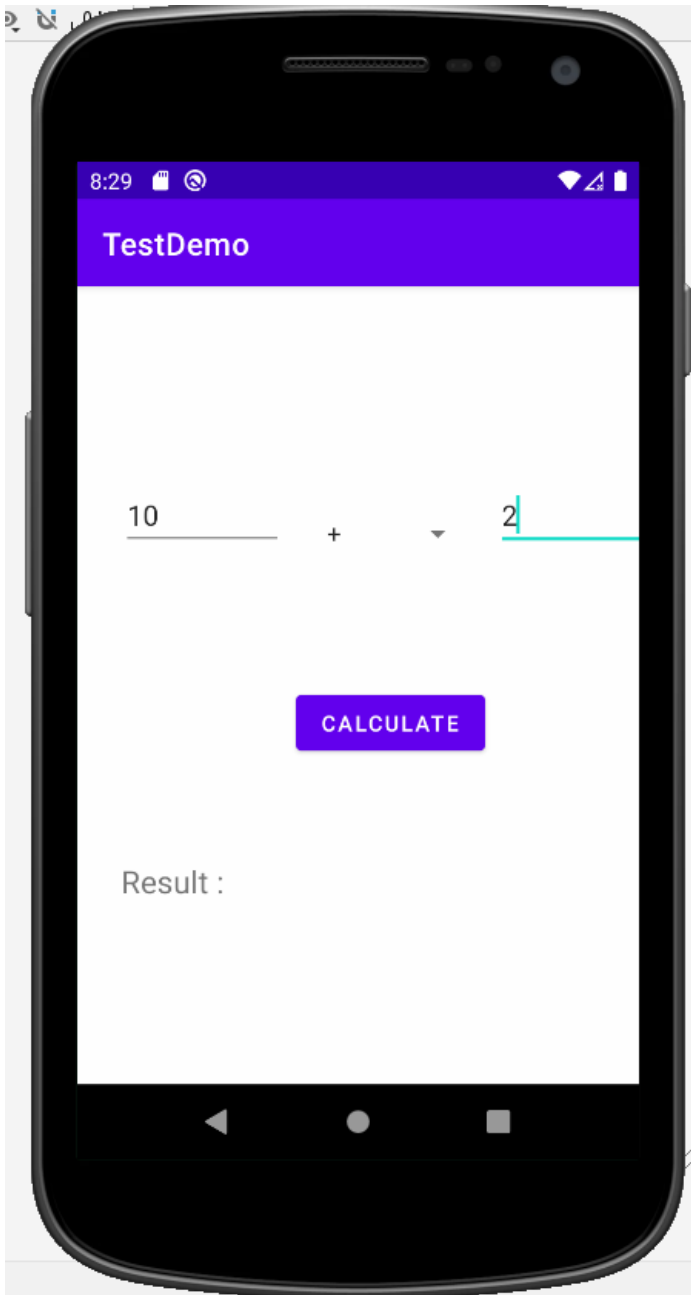




Week 2 : Session 1

More about Function, Classes and Object

Exercise



```
11 class MainActivity4 : AppCompatActivity() {
12     override fun onCreate(savedInstanceState: Bundle?) {
13         super.onCreate(savedInstanceState)
14         setContentView(R.layout.activity_main4)
15         var btn=findViewById<Button>(R.id.btn_4)
16         var fnum=findViewById<EditText>(R.id.et1_4)
17         var snum=findViewById<EditText>(R.id.et2_4)
18         var res=findViewById<TextView>(R.id.tv_4)
19         var sp=findViewById<Spinner>(R.id.spinner_4)
20         btn.setOnClickListener(View.OnClickListener { it: View!
21             var a:Float=fnum.text.toString().toFloat()
22             var b:Float=snum.text.toString().toFloat()
23             var s=sp.selectedItem.toString()
24             when(s){
25                 "+"-> res.setText("result: "+ add(a,b))
26                 "-"-> res.setText("result: "+ sub(a,b))
27                 "*"-> res.setText("result: "+ mul(a,b))
28                 "/"-> res.setText("result: "+ div(a,b))
29                 "%"-> res.setText("result: "+ mod(a,b))
30             }
31         })
32     }
33     fun add(a:Float,b:Float):Float = a+b
34     fun sub(a:Float,b:Float):Float = a-b
35     fun mul(a:Float,b:Float):Float= a*b
36     fun div(a:Float,b:Float):Float = a/b
37     fun mod(a:Float,b:Float):Float = a%b
38 }
```

Agenda

- More about Functions
 - Learn why (almost) everything has a value
 - Learn more about functions
 - Explore default values and compact functions
- Object-oriented Programming (OOP)
- Class and Objects
- Exercise 5

Exercise

More about Functions

```
3 fun main(){
4     val temperature = 10
5     val message = "The water temperature is ${ if (temperature > 50) "too warm" else "OK" }."
6     println(message)
7 }
```

Handwritten notes:
- A red circle around the `if` expression in line 5, with the word "what" written above it.
- A red bracket on the right side of the code block, labeled "class" at the top and "functia a" (sic) in the middle, with a red underline under "functia a".

- In Kotlin, almost everything is an *expression* and has a value.

```
8 fun main(){
9     val isUnit = println("This is an expression")
10    println(isUnit)
11 }
```

Handwritten notes:
- A red circle around the `println("This is an expression")` expression in line 9, with an arrow pointing to the word "Unit" written below it.
- A red arrow points from the `println(isUnit)` call in line 10 to the `Unit` type mentioned in the previous note.

- Compact functions:

- also called single-expression functions
- When a function returns the results of a single expression, you can specify the body of the function after an `=` symbol, omit the curly braces `{}`, and omit the return

```
8 fun main(){
9     print(isTooHot( temperature: 32))
10 }
11
12 fun isTooHot(temperature : Int ):Boolean = temperature > 30
```

Handwritten notes:
- A large red curly brace on the left side of the code block, spanning from line 8 to line 12.

Handwritten notes:
- A red bracket on the right side of the slide, labeled "class" at the top, "function" in the middle, and "println" at the bottom, with a red underline under "function".
- The text "Kotlin 2 Unit" is written in red below the bracket.

More about Functions

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

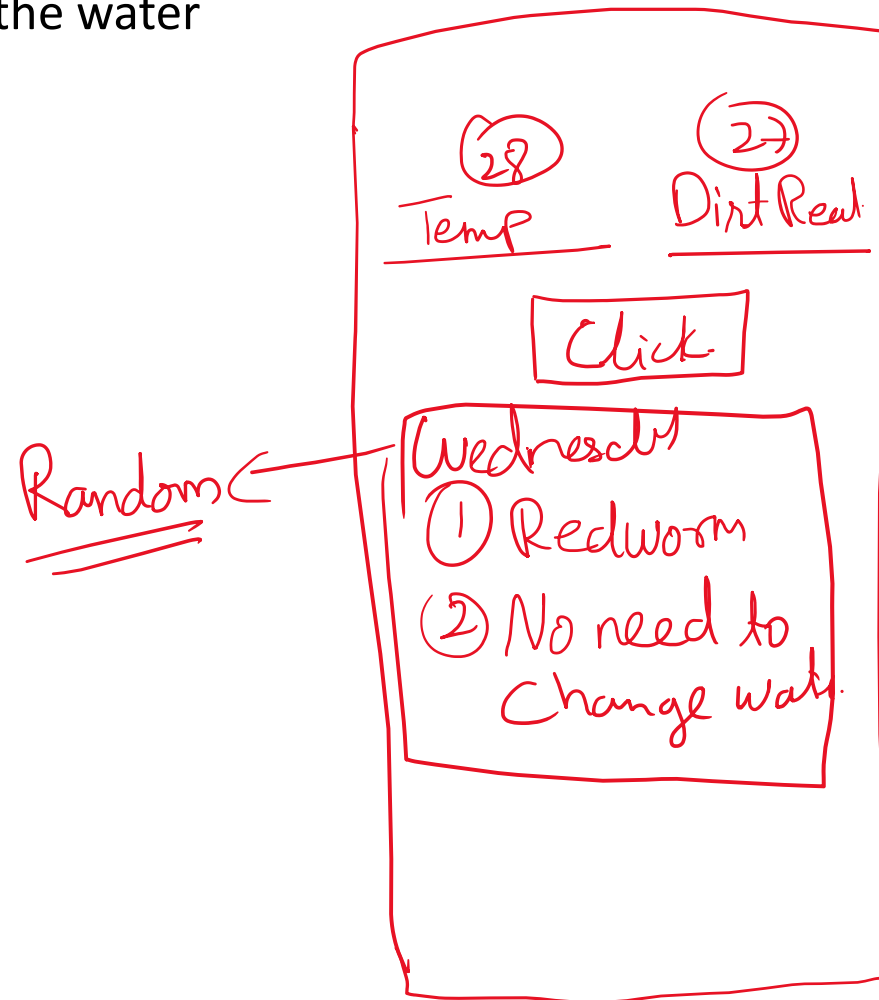
Wednesday -> redworms

Thursday -> granules

Friday -> mosquitoes

Saturday -> lettuce

Sunday -> plankton



Solution

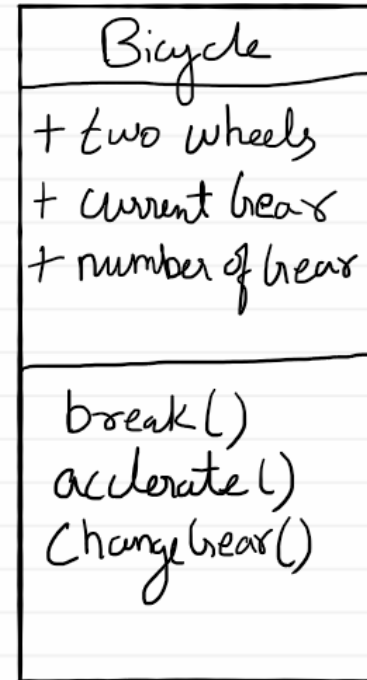
```
6 import java.util.*
7
8 fun main(){
9     feedTheFish()
10 }
11
12 fun feedTheFish() {
13     val day = randomDay()
14     val food = fishFood(day)
15     println("Today is $day and the fish eat $food")
16     println("Change water: ${shouldChangeWater(day)}")
17 }
18
19 fun fishFood (day : String) : String {
20     var food = ""
21     when (day) {
22         "Monday" -> food = "flakes"
23         "Tuesday" -> food = "pellets"
24         "Wednesday" -> food = "redworms"
25         "Thursday" -> food = "granules"
26         "Friday" -> food = "mosquitoes"
27         "Saturday" -> food = "lettuce"
28         "Sunday" -> food = "plankton"
29     }
30     return food
31 }
32
33 fun randomDay() : String {
34     val week = arrayOf("Monday", "Tuesday", "Wednesday", "Thursday",
35         "Friday", "Saturday", "Sunday")
36     return week[Random().nextInt(week.size)]
37 }
```

```
39 fun isTooHot(temperature: Int) = temperature > 30
40
41 fun isDirty(dirty: Int) = dirty > 30
42
43 fun isSunday(day: String) = day == "Sunday"
44
45 fun shouldChangeWater (day: String, temperature: Int = 22, dirty: Int = 20): Boolean {
46     return when {
47         isTooHot(temperature) -> true
48         isDirty(dirty) -> true
49         isSunday(day) -> true
50         else -> false
51     }
52 }
53
54
```



Object-oriented Programming (OOP)

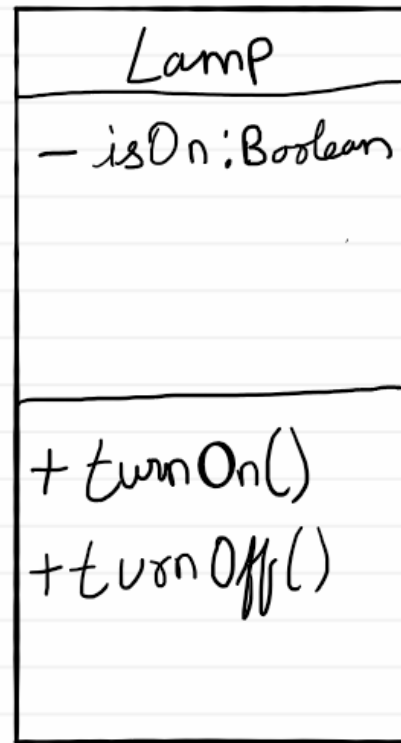
- Divide a complex problem into smaller sets by creating objects.
- These objects share two characteristics:
 - state
 - behavior
 - E.g.
 - Bicycle is an object
 - It has current gear, two wheels, number of gear etc. states.
 - It has braking, accelerating, changing gears etc. behavior.
- Features of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



Kotlin Class

- A class is a blueprint for the object(sketch (prototype)).

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



```
7  class Lamp {  
8  
9      // property (data member)  
10     var isOn: Boolean = false  
11  
12     // member function  
13     fun turnOn() {  
14         isOn = true  
15     }  
16  
17     // member function  
18     fun turnOff() {  
19         isOn = false  
20     }  
21 }
```

```
3  fun main(){  
4      val lamp = Lamp()  
5      print(lamp.isOn)  
6  }
```