Języki programowania i GUI

Lista 5 - 2022

0. Zainstaluj na swoim komputerze język `kotlin`. Zapoznaj się i wypróbuj na swoim komputerze przykłady ze strony: `https://kotlinlang.org/docs/basic-syntax.html`

1. ```
   class Osoba(var imie:String, var nazwisko:String){
       override fun toString()="$imie $nazwisko"
   }
   var osoby=listOf(
      Osoba("Jan","Kowalski"),
      Osoba("Ewa","Nowak"),
      Osoba("Artur","Kowalski"),
      Osoba("Adam","Nowak")
   )
   ```

   Napisz linię kodu sortującą i wypisującą na ekranie listę osób w kolejności alfabetycznej: (a) wg imienia używając `sortedBy`, (b) wg imienia używając `sortedWith`. (c) wg nazwiska i imienia używając `sortedWith`, (d) wg nazwiska i imienia używając `sortedBy`.

2. Napisz klasę `Prostokąt` o wymiarach typu `Double`. Długości boków `a` i `b` powinny być argumentami konstruktora, własności `pole` i `obwód` powinny posiadać gettery zawierające znane z geometrii wzory oraz settery zamieniające prostokąt w kwadrat o żądanej wartości `pola`/`obwodu`. Dodatkowo zdefiniuj własność tylko do odczytu `val przekątna` z getterem korzystającym z tw. Pitagorasa oraz metodę `toString()` pokazującą wymiary prostokąta. W funkcji `main` utwórz listę co najmniej 4 prostokątów o różnych wymiarach. Wypisz je (a) używając pętli `for`, (b) używając `forEach`. Następnie pięciokrotnie wypisz listę jako całość `println(lista.sorderBy{...})`, za każdym razem posortowaną wg innej ze zdefiniowanych własności prostokąta.

3. Z klasy prostokąt wyprowadź przez dziedziczenie klasę `Plakat`, której konstruktor ma 4 parametry: `a`, `b`, `kolor` i `tekst`. Zadbaj o to, by każdy argument miał domyślną wartość (np wymiary A4, kolor `"#FFF"`, tekst `"Witaj!"`). Przed `a` i `b` nie należy umieszczać słowa kluczowego `var`. Dodaj metodę `toString()` na przykład taką:
   `override fun toString()= "'$tekst' (plakat $a x $b w kolorze $kolor)"`
   Korzystając z parametrów kluczowych (np. `Plakat(tekst="Kotlin rulez!")`) stwórz jeszcze dwa plakaty w różnych kolorach, dodaj je do listy prostokątów z poprzedniego zadania i zobacz jak działa teraz sortowanie i wypisywanie listy.

   Uwaga: aby dziedziczenie było możliwe definicję klasy bazowej trzeba poprzedzić słowem kluczowym `open`.

4. Dany jest interfejs: `interface Masywny {val masa:Double}`
   Napisz proste klasy: `Osoba`, `Zwierzę`, `Bagaż`, `Auto` implementujące ten interfejs, oraz metodę `toString()`, utwórz listę `var cargo=..` złożoną z obiektów tych klas, a następnie (a) wypisz elementy listy malejąco wg masy, (b) użyj metody `fold`, by wyznaczyć łączną oraz średnią masę obiektów z listy `cargo`, (c) użyj metod `filter` oraz `forEach` by wypisać elementy o wadze powyżej średniej.

5. * (2pkt) Wyważanie promu. Napisz funkcję która dla dowolnej listy obiektów z interfejsu `Massive` znajdzie podzbiór o masie maksymalnie zbliżonej do (ale nie przekraczającej) połowy łącznej masy całego ładunku. Wskazówka: można rozważyć wszystkie podzbiory.

6. Rozkład liczby na czynniki pierwsze można wykonać za pomocą iteratora:
   ```

```
class Rozkład(var n:Int=1){
   var i=2
   operator fun iterator()=this
   operator fun hasNext()=n>1
   operator fun next():Int { while(n%i!=0) i++; n/=i; return i}
}
fun main(){
   print("120=")
   for(x in Rozkład(120))
      print(" $x")
}
```

Napisz iterator `Dzielniki` generujący w podobny sposób wszystkie dzielniki liczby n.

7. Jeśli ostatnim argumentem funkcji jest lambda (funkcja anonimowa), to można ją wywoływać umieszczając lambdę poza listą argumentów. Na przykład:

```
fun rozkład(n:Int, operations:(Int)->Unit)
{...}
fun main(){
   print("120=")
   rozkład(120){ print(" $it") }
}
```

Uzupełnij ciało funkcji `rozkład`, tak by działanie kodu było identyczne, jak w poprzednim zadaniu. W podobny sposób napisz i przetestuj funkcję `dzielniki`. Następnie, używając funkcji `dzielniki` z odpowiednio dobraną lambdą, oblicz sumę dzielników liczby 144.

8. Kotlin pozwala "dodefiniować metodę" do istniejącej klasy. Na przykład:
`fun String.długość()=this.length`
Można również "dodefiniowywać" operatory:

```
operator fun Int.times(text:String):String{
  var res=""; repeat(this){res+=text}; return res
}
println(3 * "Witaj! ") // "Witaj! Witaj! Witaj! "
```

A nawet mnożyć liczby przez funkcje anonimowe:
`operator fun Int.times(action:(Int)->Unit){repeat(this,action)}`
by potem pisać np. `3*{println("Cześć!")}`

Zdefiniuj odpowiednią funkcję tak, aby zachodziła równość:
`"Ala ma kota"*3 == "AAAlllaaa   mmmaaa   kkkooottttaaa"`

9. Napisz klasę funkcję `gcd()` oraz klasę `Ułamek` uzupełniając wykropkowane miejsca:

```
fun gcd(a:Int,b:Int):Int{...}

data class Ułamek(var licznik:Int=0, var mianownik:Int=1){
    init{
        val x=gcd(Math.abs(licznik),mianownik)
        licznik/=x
        mianownik/=x
    }
```

```
    override fun toString()="$licznik/$mianownik"
    operator fun times(u:Ułamek)=...
    operator fun div(u:Ułamek)=...
    operator fun plus(u:Ułamek)=...
    operator fun minus(u:Ułamek)=...
    operator fun unaryMinus()=...
    operator fun unaryPlus()=this


    operator fun times(u:Int):Ułamek=...
    operator fun div(u:Int):Ułamek=...
    operator fun plus(u:Int):Ułamek=..
    operator fun minus(u:Int):Ułamek=...
}

operator fun Int.times(u:Ułamek):Ułamek=..
operator fun Int.plus(u:Ułamek):Ułamek=...
operator fun Int.minus(u:Ułamek):Ułamek=...
operator fun Int.div(u:Ułamek):Ułamek=...
```

I przetestuj jej działanie na przykład w taki sposób:

```
fun main()
{
var a=Ułamek(2,5)
var b=Ułamek(3,10)
println("a=$a")
println("b=$b")
println("$a * $b = ${a*b}")
println("$a / $b = ${a/b}")
println("$a + $b = ${a+b}")
println("$a - $b = ${a-b}")
println("$a + 4 = ${a+4}")
println("$a - 4 = ${a-4}")
println("$a * 4 = ${a*4}")
println("$a / 4 = ${a/4}")
println("4 + $a = ${4+a}")
println("4 - $a = ${4-a}")
println("4 * $a = ${4*a}")
println("4 / $a = ${4/a}")
}
```