	Wydział Politechniczny **Katedra Informatyki**
Prowadzący	Tomasz Gądek
Kurs	Narzędzia i środowiska programistyczne
Rok / Semestr	1 / Letni
Temat	Lab05: Programowanie obiektowe.

Data ostatniej modyfikacji: **01-10-2024**

Klasa, obiekt i konstruktor

Nasz świat podlega klasyfikacji. Wszystkie przedmioty, które nas otaczają można zaprezentować w postaci klas i obiektów w języku Kotlin. Za chwilę dowiesz się w jaki sposób konstruować klasy i obiekty. Spójrz na poniższy kod klasy **Person**.

```
class Person { // 1
  var name: String // 2

  constructor(name: String) { // 3
    this.name = name
  }
}

fun main() {
  var mike = Person("Mike") // 4
  var tom = Person("Tom") // 4

  println("Hello: ${mike.name}")
  println("Hello: ${tom.name}")
}
```

Rezultat działania kodu jest następujący.

```
Hello: Mike
Hello: Tom
```

Klasa reprezentowana jest poprzez słowo kluczowe **class**. Ciało klasy obejmują nawiasy klamrowe **{** i **}**. Wewnątrz klasy znajduje się pole **name**. Będzie ono przechowywało informacje o imieniu użytkownika. Poniżej pola **name** znajduje się tzw. *konstruktor*. Jest to specjalna metoda klasy, która jest wywoływana w momencie tworzenia obiektu (komentarz nr 4). Do konstruktora przekazujemy konkretne imię, które jest przypisywane do pola klasy **this.name = name** (przy pomocy **this** wskazujemy, które **name** należy do obiektu klasy).

Wyobraź sobie, że klasa jest pewnego rodzaju szablonem osoby, a dopiero obiekt tworzy konkretną osobę na podstawie szablonu (obiekt "żyje", a definicja jest wzorem dla obiektów). Po utworzeniu obiektu możemy mieć dostęp do konkretnego pola. Do pola odwołujemy się przy pomocy operatora kropki "." (pod warunkiem, że pole **name** nie będzie prywatne, ale o modyfikatorach dostępu opowiemy sobie w kolejnym rozdziale). Po kropce podajemy konkretną nazwę pola.

W komentarzach zostały umieszczone cyfry, które odnoszą się do odpowiednich konstrukcji: 1 = klasa, 2 = pole klasy, 3 = konstruktor, 4 = obiekt klasy Person.

Pola, metody i modyfikatory dostępu

Spójrz na rozbudowany przykład klasy Person.

```
class Person {
  public var name: String
  private var age: Int

  constructor(name: String, age: Int) {
      this.name = name
      this.age = age
  }

  internal fun getName(): String {
      return this.name
  }

  fun getAge(): Int {
      return this.age
  }
}

fun main() {
  var mike = Person("Mike", 18)
  println("${mike.name}, ${mike.getName()}, ${mike.getAge()}");
}
```

Rezultat jest następujący.

```
Mike, Mike, 18
```

W naszym kodzie mamy sporo słów kluczowych. Przed polami i niektórymi metodami znajdują się następujące słowa kluczowe: **public**, **private**, **internal**. Wymienione słowa kluczowe to **modyfikatory dostępu**. Możemy nimi oznaczać pola i metody (**metoda** klasy to jest **funkcja**, która jest umieszczona wewnątrz klasy) klasy.

Oto lista dostępnych modyfikatorów dostępu.

- **public**: element jest dostępny z każdego miejsca w programie.
- **private**: element jest dostępny tylko wewnątrz klasy, w której został zdefiniowany.
- **protected**: element jest dostępny wewnątrz klasy oraz w klasach dziedziczących.
- **internal**: element jest dostępny tylko wewnątrz tego samego modułu.

Zapamiętaj sobie, że gdy nie podasz żadnego modyfikatora dostępu to domyślnie takie pole, czy metoda jest publiczna. W naszym przykładzie metoda **getAge()** jest metodą publiczną.

Pojazdy

Zanim przystąpisz do realizacji zadań przeanalizuj poniższy kod i skopiuj go do swojego projektu.

```
class Car {
    private var mark: String
    public var color: String
    private var speed: Int
    private var engineStarted: Boolean

    constructor(mark: String, color: String) {
        this.mark = mark
        this.color = color
        this.speed = 0
        this.engineStarted = false
    }

    private fun incSpeed() { // metoda prywatna
        this.speed += 10
    }

    fun run() { // metoda publiczna
        incSpeed()
    }

    fun checkSpeed(): Int { // metoda publiczna
        return speed
    }
}

fun main() {
    // tworzenie obiektu, czyli konkretnego pojazdu
    val audi = Car("Audi", "SILVER")

    audi.color = "RED" // public - ok, dostęp publiczny
    //audi.speed = 200 // private - błąd, dostęp prywatny

    // dostęp do metody publicznej: [obiekt].[metoda]
    println("Audi ma na liczniku ${audi.checkSpeed()} km/h")
    println("Audi przyspiesza...")

    audi.run() // przyspiesza...
    audi.run() // przyspiesza...
    audi.run() // nadal przyspiesza...

    println("Audi ma na liczniku ${audi.checkSpeed()} km/h")
}
```

Gdy już będziesz gotowy/a postaraj się wykonać poniższe zadania.

Zadania

Klasa **Car**

- Klasę **Car** umieść w oddzielnym pliku (**Car.kt**).
- Pole **color** w klasie **Car** powinno być prywatne (staramy się ukrywać pola przed światem zewnętrznym). Nanieś poprawki w klasie.
- Dodaj do klasy **Car** dwie publiczne metody (**engineStart()** i **engineStop()**), które ustawią w odpowiedni sposób pole **engineStarted**.
- W metodzie odpowiedzialnej za zatrzymanie silnika powinniśmy ustawić prędkość na **0** oraz pole **engineStarted** na **false**.
- W metodzie odpowiedzialnej za uruchomienie silnika powinniśmy ustawić prędkość na **0** oraz pole **engineStarted** na **true**.
- Dodaj zabezpieczenie w metodzie **incSpeed()**, które zwiększy prędkość pojazdu pod warunkiem, że silnik będzie uruchomiony.
- Utwórz jeszcze kilka obiektów / pojazdów (np. **tesla**, **audi**, **bmw**, **ferrari**).
- Umieść pojazdy na liście.
- Uruchom silnik w co drugim pojeździe (użyj pętli **for in**).

Klasa **Student**

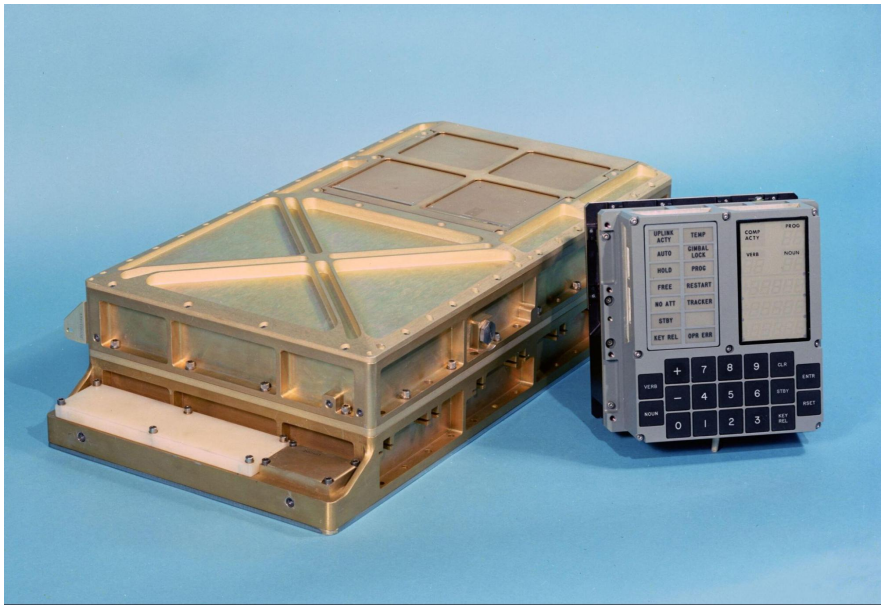
- Utwórz plik **Student.kt** oraz zaprojektuj klasę **Student**, która posiada atrybuty: **idx**, **firstName**, **lastName**, **mark**. Zaimplementuj konstruktor oraz gettery (metody, które zwracają konkretne wartości pól np: **getFirstName()**).

Klasa **University** (zarządzająca studentami)

- Utwórz plik **University.kt** oraz zaprojektuj klasę, która zarządza studentami (**University**), konstruktor klasy powinien przyjmować listę studentów (listę obiektów typu **Student**).
- W klasie **University** utwórz publiczną metodę **avg()**, która zwraca średnią arytmetyczną ocen.
- W klasie **University** utwórz metodę **failed()**, która zwraca listę studentów, którzy nie uzyskali zaliczenia (**2.0**).
- W klasie **University** utwórz metodę **add()** , która przyjmuje obiekt studenta i będzie odpowiedzialna za dodawanie nowego studenta do listy studentów.
- W klasie **University** utwórz metodę **update()**, która przyjmuje numer indeksu oraz nowy obiekt studenta, który powinien być zaktualizowany na liście.
- W klasie **University** utwórz metodę **remove()**, która przyjmuje numer indeksu studenta, który zostanie usunięty.

Czy wiesz, że...

Komputer pokładowy Apollo 11 (AGC) posiadał zaledwie 64 kB pamięci RAM, a jego jednostka obliczeniowa pracowała z częstotliwością 2 MHz. Był jednym z najbardziej imponujących osiągnięć technologicznych ludzkości.



AGC jest uznawany za jedno z najbardziej imponujących osiągnięć technologicznych ludzkości. Jego pamięć i moc obliczeniowa mogą wydawać się niewiarygodnie małe w porównaniu do dzisiejszych standardów, ale w tamtych czasach pozwoliły na realizację jednego z najważniejszych wydarzeń w historii ludzkości - lądowania człowieka na Księżycu.