Композиція. Агрегація

Композиція у програмуванні або ж **Об'єктна композиція**, також **Агрегація** та **включення** — це базові принципи об’єктно-орієнтованого програмування, згідно з якими створення нових класів може бути здійснено з використанням компонент, які є об’єктами інших класів

Порівняльна таблиця ***композиції*** і ***агрегації***

Композиція	***Агрегація***
відношення приналежності (відношення типу “частина-ціле”)	принцип включення
цільовий об’єкт створює свої об’єкти-компоненти	цільовий об’єкт отримує адресу об’єктів-компонентів
при знищенні цільового об’єктів об’єкти-компоненти не мають сенсу і повинні бути знищені (між об’єктами класів існує зв’язок за часом існування)	при знищенні цільового об’єктів об’єкти-компоненти можуть продовжувати своє незалежне існування (між об’єктами класів не існує зв’язку за часом існування)

Раніше розглядали вкладені об’єкти

Задача. Описати клієнта банку
----- Властивості ------
- ПІБ
  * прізвище
  * ім’я
- адреса
  * код
  * місто
  * вулиця
  * номер будинку
- номер рахунку
- кількість грошей

Раніше розглядали вкладені об’єкти

Задача. Описати клієнта банку
----- Властивості ------
- ПІБ
  * прізвище
  * ім’я
- адреса
  * код
  * місто
  * вулиця
  * номер будинку
- номер рахунку
- кількість грошей

Крок 0. Позначення величин

----- Властивості ------	--- позначення ---
- ПІБ	- title
* прізвище	* secondName
* ім’я	* firstName
- адреса	- address
* код	* zipCode
* місто	* city
* вулиця	* street
* номер будинку	* num
- номер рахунку	- accountNumber
- кількість грошей	- balance

Раніше розглядали вкладені об’єкти

Задача. Описати клієнта банку
----- Властивості ------
- ПІБ
  * прізвище
  * ім’я
- адреса
  * код
  * місто
  * вулиця
  * номер будинку
- номер рахунку
- кількість грошей

Крок 0. Позначення величин

----- Властивості ------	--- позначення ---
- ПІБ	- title
* прізвище	* secondName
* ім’я	* firstName
- адреса	- address
* код	* zipCode
* місто	* city
* вулиця	* street
* номер будинку	* num
- номер рахунку	- accountNumber
- кількість грошей	- balance

```
let client = {
  title:    {
            secondName: 'Smith',
            firstame: 'John',
          },
  address: {
            zipCode: '88000',
            city: 'Uzhhorod',
            street: 'Svobody',
            num: 15,
          },
  accountNumber: 'ac2341fsr2',
  balance: 2587000,
}
```

Реалізуємо з використанням класів

Крок 0. Позначення величин

----- Властивості ------	--- позначення ---
- ПІБ	- title:
	Person
* прізвище	* secondName
* ім’я	* firstName
- адреса	- address
	Address
* код	* zipCode
* місто	* city
* вулиця	* street
* номер будинку	* num
- номер рахунку	- accountNumber
- кількість грошей	- balance

```
class Person {
  constructor({ firstName, surname, age }) {
    this.firstName = firstName
    this.surname = surname
    this.age = age
  }

  get fullName() {
    return this.firstName + ' ' + this.surname
  }
  set fullName(value) {
    var split = value.split(' ')
    this.firstName = split[0]
    this.surname = split[1]
  }

  toString() {
    return `${this.fullName} - ${this.age}`
  }
}
```

Реалізуємо з використанням класів

Крок 0. Позначення величин

----- Властивості ------	--- позначення ---
- ПІБ	- title: Person
* прізвище	* secondName
* ім’я	* firstName
- адреса	- address: Address
* код	* zipCode
* місто	* city
* вулиця	* street
* номер будинку	* num
- номер рахунку	- accountNumber
- кількість грошей	- balance

```
class Person {
  constructor({ firstName, surname, age }) {
    this.firstName = firstName
    this.surname = surname
    this.age = age
  }

  get fullName() {
    return this.firstName + ' ' + this.surname
  }
  set fullName(value) {
    var split = value.split(' ')
    this.firstName = split[0]
    this.surname = split[1]
  }

  toString() {
    return `${this.fullName} - ${this.age}`
  }
}
```

```
class Address {
  constructor(zipCode, city, street, num) {
    this.zipCode = zipCode
    this.city = city
    this.street = street
    this.num
  }
  toString() {
    return `${this.zipCode} ${this.city}
    ${this.street} ${this.num}`
  }
}
```

Реалізуємо з використанням класів

Крок 0. Позначення величин

----- Властивості ------ --- позначення ---

- ПІБ - title:
  - * прізвище
  - * ім’я
- адреса - address
  - * код
  - * місто
  - * вулиця
  - * номер будинку
- номер рахунку - accountNumber
- кількість грошей - balance

Person
* secondName
* firstName

Address
* zipCode
* city
* street
* num

```
class Address {
  constructor(zipCode, city, street, num) {
    this.zipCode = zipCode
    this.city = city
    this.street = street
    this.num
  }
  toString() {
    return `${this.zipCode} ${this.city}
    ${this.street} ${this.num}`
  }
}
```

```
class Person {
  constructor({ firstName, surname, age }) {
    this.firstName = firstName
    this.surname = surname
    this.age = age
  }

  get fullName() {
    return this.firstName + ' ' + this.surname
  }
  set fullName(value) {
    var split = value.split(' ')
    this.firstName = split[0]
    this.surname = split[1]
  }

  toString() {
    return `${this.fullName} - ${this.age}`
  }
}
```

```
class Client {
  #balance
  constructor(initData) {

    this.title = new Person(initData)

    this.Address = new Address(initData)

    this.accountNumber = initData.accountNumber
    this.balance = initData.balance
  }

  . . . . .
}
```

```
class Person {
  constructor({ firstName, surname, age }) {
    this.firstName = firstName
    this.surname = surname
    this.age = age
  }

  get fullName() {
    return this.firstName + ' ' + this.surname
  }
  set fullName(value) {
    var split = value.split(' ')
    this.firstName = split[0]
    this.surname = split[1]
  }

  toString() {
    return `${this.fullName} - ${this.age}`
  }
}
```

```
class Client {
  #balance
  constructor(initData) {
    this.title = new Person(initData)

    this.Address = new Address(initData)

    this.accountNumber = initData.accountNumber
    this.balance = initData.balance
  }

  get balance() {
    return this.#balance
  }
  set balance(val) {
    if (val < 0) throw new Error('The value i incorrect')
    this.#balance = val
  }
  toString() {
    return `${this.title}\n${this.Address}\n${this.accountNumber}\n${this.balance}`
  }
}
```

```
class Address {
  constructor({ zipCode, city, street, num }) {
    this.zipCode = zipCode
    this.city = city
    this.street = street
    this.num = num
  }
  toString() {
    return `${this.zipCode} ${this.city} ${this.street} ${this.num}`
  }
}
```

```
let client1 = new Client({
  firstName: 'Ivan', surname: 'Sirko', age: 21,
  zipCode: 88000, city: 'Best sity', street: 'Holovna', num: 25,
  accountNumber: '21212234546',
  balance: 45000,
})

console.log(client1)
```

```
class Person {
  constructor({ firstName, surname, age }) {
    this.firstName = firstName
    this.surname = surname
    this.age = age
  }

  get fullName() {
    return this.firstName + ' ' + this.surname
  }
  set fullName(value) {
    var split = value.split(' ')
    this.firstName = split[0]
    this.surname = split[1]
  }

  toString() {
    return `${this.fullName} - ${this.age}`
  }
}
```

```
class Client {
  #balance
  constructor(initData) {
    this.title = new Person(initData)

    this.Address = new Address(initData)

    this.accountNumber = initData.accountNumber
    this.balance = initData.balance
  }

  get balance() {
    return this.#balance
  }
  set balance(val) {
    if (val < 0) throw new Error('The value i incorrect')
    this.#balance = val
  }
  toString() {
    return `${this.title}\n${this.Address}\n${this.accountNumber}\n${this.balance}`
  }
}
```

Композиція

```
class Address {
  constructor({ zipCode, city, street, num }) {
    this.zipCode = zipCode
    this.city = city
    this.street = street
    this.num = num
  }
  toString() {
    return `${this.zipCode} ${this.city} ${this.street} ${this.num}`
  }
}
```

```
let client1 = new Client({
  firstName: 'Ivan', surname: 'Sirko', age: 21,
  zipCode: 88000, city: 'Best sity', street: 'Holovna', num: 25,
  accountNumber: '21212234546',
  balance: 45000,
})

console.log(client1)
```

Задача. Дано список учнів школи.У школі проводиться декілька гуртків. Учень може брати участь у декількох гуртках. Реалізувати додавання/вірахування учнів з гуртків

===== Учень ======
- прізвище
- ім’я
- номер класу

===== Гурток ======
----- Властивості ------
- назва
- день провдення
- список учасників-учнів. Список об’єктів
  {
    id - *учасника*
    data - інформація про учасника
  }

---- Методи -----
- додвання учня
- відрахування учня
- перевірка належності учня до гуртка

```
class Student {
  constructor(initData) {
    Object.assign(this, initData)
  }
  // constructor({ firstName, surname, classNum }) {
  //   this.firstName = firstName
  //   this.surname = surname
  //   this.classNum = classNum
  // }

  toString() {
    return `${this.firstName} ${this.surname} - ${this.classNum}`
  }
}
```

Задача. Дано список учнів школи.У школі проводиться декілька гуртків. Учень може брати участь у декількох гуртках. Реалізувати додавання/вірахування учнів з гуртків

===== Учень ======

- прізвище
- ім’я
- номер класу

===== Гурток ======

----- Властивості ------

- назва
- день провдення
- список учасників-учнів. Список об’єктів

{

id - *учасника*

data - інформація про учасника

}

---- Методи -----

- додвання учня
- відрахування учня
- перевірка належності учня до гуртка

```
class Workshop {
  // constructor({title, workingDay}) {
  //   this.title=title
  //   this.workingDay=workingDay
  //   this.participants = []
  // }
  constructor(initData) {
    Object.assign(this, initData)
    this.participants = []
  }
  addStudent(newStudent) {
    this.participants.push({
      id: new Date().getTime(),
      data: newStudent,
    })
  }
  removeStudentById(studentIdToRemove) {
    this.participants = this.participants.filter(
      (item) => item.id !== studentIdToRemove
    )
  }
  removeStudent(studentToRemove) {
    this.participants = this.participants.filter(
      (item) => item !== studentToRemove
    )
  }
  isStudentVisitingWorkshop(student) {
    return this.participants.includes(student)
  }
  toString() {
    return `${this.title} : <br> ${this.participants.map(
      (item) => item.toString() + '<br>'
    )}`
  }
}
```

Агрегація

```
class WorkshopManager {
  constructor(studentsList, workshops) {
    this.studentsList = studentsList
    this.workshops = workshops
  }
  addStudentsToWorkshop() {
    let workshopsNames = Object.keys(workshops)
    for (const student of this.studentsList) {
      const selectedWorkshopNumbers = prompt(
        `Виберіть через кому номери гуртків для\n${student} \n \n1. Math\n2.History\n3. Geography`
      )
        .split(',')
        .map((item) => parseInt(item) - 1)

      for (const workshopIndex of selectedWorkshopNumbers) {
        const workshopKey = workshopsNames[workshopIndex]
        this.workshops[workshopKey].participants.push(student)
      }
    }
  }
  printData() {
    for (const workshopKey in this.workshops) {
      document.write(workshops[workshopKey])
    }
  }
}
```

```
let studentsList = [
  new Student({ firstName: 'Ivan', surname: 'Sirko', classNum: 7 }),
  new Student({ firstName: 'Petro', surname: 'Sich', classNum: 8 }),
  new Student({ firstName: 'Olga', surname: 'Kazka', classNum: 3 }),
  new Student({ firstName: 'John', surname: 'Rich', classNum: 5 }),
  new Student({ firstName: 'Mihael', surname: 'Bax', classNum: 6 }),
]

let workshops = {
  math: new Workshop({ title: 'Math', workingDay: 3 }),
  history: new Workshop({ title: 'History', workingDay: 4 }),
  geography: new Workshop({
    title: 'Geography',
    workingDay: 1,
  }),
}

let workshopManager = new WorkshopManager(studentsList, workshops)
workshopManager.addStudentsToWorkshop()
workshopManager.printData()

workshopManager.workshops['math'].removeStudent(studentsList[0])
workshopManager.printData()
```

Задача. Команда. Дано список студентів (ім"я, кількість балів). По черзі випадковим чином вибирають дві команди. Виграє той, у кого середній бал менше

Задача. Склад. База товарів, які зберігаються на складі: назва товару, одиниця виміру, кількість, фірма виробник (назва, реєстраційний номер). Організувати реєстрацію/відвантаження товарів, філтрація за назвою товару, фільтрація за назвою фірми.