Уводна лекция

Трифон Трифонов

Обектно-ориентирано програмиране,
спец. Компютърни науки, 1 поток, 2021/22 г.

23 февруари 2022 г.

Какво е обектно-ориентирано програмиране?

- Вече знаем че на C++ можем да напишем всичко, което може да се напише на компютър
- Всъщност, за това ни трябват само `int[]`, `if` и `while`
- Какво повече ни трябва?
- Искаме да пишем програми, които са
  - лесни за четене и разбиране
  - лесни за писане и промяна
  - удобни за използване от други разработчици
  - удобни за работа от много хора едновременно

Какво е обектно-ориентирано програмиране?

- Обектно-ориентираното програмиране (ООП) е **стил на програмиране**
- Дава удобен начин за представянето на реални проблеми и задачи в език за програмиране
- Въвежда дисциплина и структура в програмите, което ги прави по-разбираеми
- Задава правила, които правят разширяването и използването на програми по-лесно
- Стимулира използването на интуитивни имена и понятия

Митове за ООП

- ООП е универсалното решение
  - по-скоро едно от възможните решения
  - има предимства, но и недостатъци
- Само някои програмни езици стават за ООП
  - ООП е стил, а не език
  - някои езици наистина са по-удобни, но всеки език допуска ООП
  - всички съвременни езици имат добра поддръжка за ООП
- ООП е иновативна концепция
  - всъщност датира от 60-те години на миналия век
  - C е създаден през 1972 г., а C++ през 1979 г.

Основната идея

- Представяне на частите от решаваната задача като набор от **обекти**, които включват в себе си **данни** и **методи** за обработката на тези данни
- Еднотипни обекти се групират в **класове**
- Методите включват
  - конструктори (функции за построяване на обекти)
  - селектори (функции за достъп до компоненти на обекти)
  - мутатори (функции за промяна на компоненти на обекти)
  - и много други

Някои основни принципи

- **Абстракция** със структури от данни
  - представянето на данните е отделено от използването им
- **Отворена рекурсия**
  - методите работят със „собствените“ данни на обекта
- **Наследяване**
  - един клас от обекти може да разширява друг вече съществуващ клас като използва наготово функционалността му
- **Генеричност**
  - обработване на различни класове обекти по универсален начин
- **Полиморфизъм**
  - обработване на различни класове обекти по специфичен за тях начин
- **Динамично свързване**
  - извиканият метод се определя по време на изпълнение, в зависимост от обекта, а не от класа на който принадлежи

Да започваме!