Уводна лекция

Трифон Трифонов

Обектно-ориентирано програмиране,
спец. Компютърни науки, 1 поток,
спец. Софтуерно инженерство,
2016/17 г.

23 февруари 2017 г.

Екип

Компютърни науки, 1 поток:

- **1 група семинар:** Дафина Зотева
- **1 група практикум:** Дафина Зотева
- **2 група семинар:** Боян Киров
- **2 група практикум:** Марио Дончев, Стамен Драгоев
- **3 група семинар:** Дафина Зотева
- **3 група практикум:** Владимир Начев, Яна Георгиева
- **4 група семинар:** Димитър Узунов
- **4 група практикум:** Алекс Николов

Софтуерно инженерство:

- **4 група семинар:** Николай Стойков
- **5 група семинар:** Алекс Сърбински
- **6 група семинар:** Никола Димитров

Схема за оценяване

- Минимално ниво (Среден 3)
  - Кратък тест с въпроси върху материала
  - Провежда се по време на семестъра
  - Практическа част (програма или фрагмент код)
  - Теоретични въпроси (затворени или отворени с кратък отговор)
- Средно и високо ниво (Добър 4 и Много добър 5)
  - Изпит през сесията
  - Тест с две нива, високото ниво се отключва след преминаване на средното
  - Практическа задача с две нива на трудност
- Отлично ниво (Отличен 6)
  - Разработване на курсов проект
  - Защита на проект през сесията
- Всяко следващо ниво изисква успешно преминаване на предишното
- Текущият контрол няма да е определящ за оценката

Learn.fmi

- `https://learn.fmi.uni-sofia.bg/`
- Общ курс за всички специалности
- Учебни материали
  - Обектно-ориентирано програмиране 2016/17

ООП-практикум

- Избираема дисциплина
- Необходимо е да се запише в СУСИ в рамките на кампанията за избираеми дисциплини
- Силно препоръчително е записването
- Софтуерно инженерство могат да посещават часовете по практикум заедно с Компютърни науки както следва:
  - 4 група СИ: заедно с 6 група КН
  - 5 група СИ: заедно с 5 или 8 група КН
  - 6 група СИ: заедно с 2 или 4 група КН
- Оценяване
  - Отделна оценка от ООП
  - Контролно през семестъра (50%)
  - Защита на проект (50%)

Какво е обектно-ориентирано програмиране?

- Вече знаем че на C++ можем да напишем всичко, което може да се напише на компютър

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 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?

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

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

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

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

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**

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

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

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

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

Митове за ООП

- ООП е универсалното решение
  - по-скоро едно от възможните решения

Митове за ООП

- ООП е универсалното решение
  - по-скоро едно от възможните решения
  - има предимства, но и недостатъци

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

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

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

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

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

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

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

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**

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

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

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

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

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

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!