до онлайн навчання та навіть деяких робіт в онлайн форматі, сталося набагато швидше та легше.

*Список використаних джерел*

1. *Блог про онлайн навчання. Url: https://creately.com/blog/education/online-teaching-tools/*
2. *Стаття про Google Classroom. Url: https://edu.google.com/intl/ALL_us/workspace-for-education/classroom/*
3. *Стаття про Microsoft Teams. Url: https://desktime.com/blog/how-to-use-microsoft-teams*

**УДК 004.02**

*Підруцький Д.А., здобувач 2 курсу*
*Ніколюк П.К., професор, доктор фізико-математичних наук*
*кафедри Інформаційних технологій*

## ВИКОРИСИТАННЯ ПРИНЦИПІВ ПРОГРАМУВАННЯ ДЛЯ СПРОЩЕННЯ ТА ПОЛЕГШЕННЯ ПРОЦЕСІВ

*Донецький національний університет імені Василя Стуса*

Роздумуючи над вибором теми для майбутньої тези, яка має бути пов’язаною з моєю спеціальністю на парі з тематикою технологій програмування, я все ж таки знайшов ту тему яка могла б одночасно підійти до обох тематик на які я намагався орієнтуватись, тому все ж таки вибрав тему для тези – принципи програмування та підходи до швидшого написання коду, що може бути й корисними, якщо ви активно займаєтесь програмування чи в звичному плануванні.

Чи актуально це? На мою думку так, оскільки хоч і ці підходи, які я буду досліджувати вже давно сформувалися, одначе є ще багато нових та не дуже розробників, яким було б цікаво навчитись за допомогою цих підходів чи принципів писати код швидше ніж інші та бути більш конкурентноспроміжними, ніж інші і тому припускаю що ця тема ще має попит в своїх кругах, тому і є актуальною.

Таких чином, до основних таких підходів, я можу віднести принцип "Don't Repeat Yourself» (DRY), який був сформований Енді Хантом і Дейві Томасом у своїй книзі «Прагматичний програміст»[1], в якій вони його застосовують доволі широко, включаючи плани тестування, схеми баз даних та навіть до документації. Також віднесу принцип Kiss «Keep It Short and Simple», який також заключається в швидшому написанні коду, але шляхом його спрощення та розбиття на легші методи і який проаналізував в своїй книзі Ерік Реймонд під

назвою «Філософія Unix»[2], де був описаний досвід розробників ОС Unix, які розповідали також і про цей метод.

Написати простий код чи повноцінну програму, використовуючи мінімум класів та функцій, та й взагалі мінімум символів, для його написання та розуміння іншими користувачами, які б звісно спробували б його прочитати та зрозуміти його суть без додаткових коментарів.

Оскільки наші підходи в основному використовуються в програмуванні, тому продивимось на прямому прикладі з блогу про використання цих методів[3], де ми маємо ось такий код, який генерує псевдо випадові числа:

```
const DB = require('./helpers/db.js')

function saveUser() {
  const minValueCode = Math.ceil(Number(1000))
  const maxValueCode = Math.floor(Number(4000))
  const newCode = Math.floor(Math.random() * (maxValueCode - minValueCode)) + minValueCode

  const minValueHash = Math.ceil(Number(100000))
  const maxValueHash = Math.floor(Number(999999))
  const newHash = Math.floor(Math.random() * (maxValueHash - minValueHash)) + minValueHash

  const database = new DB()
  database
    .table('users')
    .save({
      name: 'Maxim',
      code: newCode,
      hash: newHash
    })
}

function sendCodeForCaptcha() {
  const minValue = Math.ceil(Number(1000))
  const maxValue = Math.floor(Number(4000))
  const result = Math.floor(Math.random() * (maxValue - minValue)) + minValue
  return result
}

saveUser()
sendCodeForCaptcha()
```

Рис.1 Код до змін принципами програмування.

Ми бачимо, що програма має виділені частини, які повторюються кілька разів, але який ми можемо спростити та зробити легшим для читання.

І в результаті застосуванню цих методів, ми отримаємо такий код:

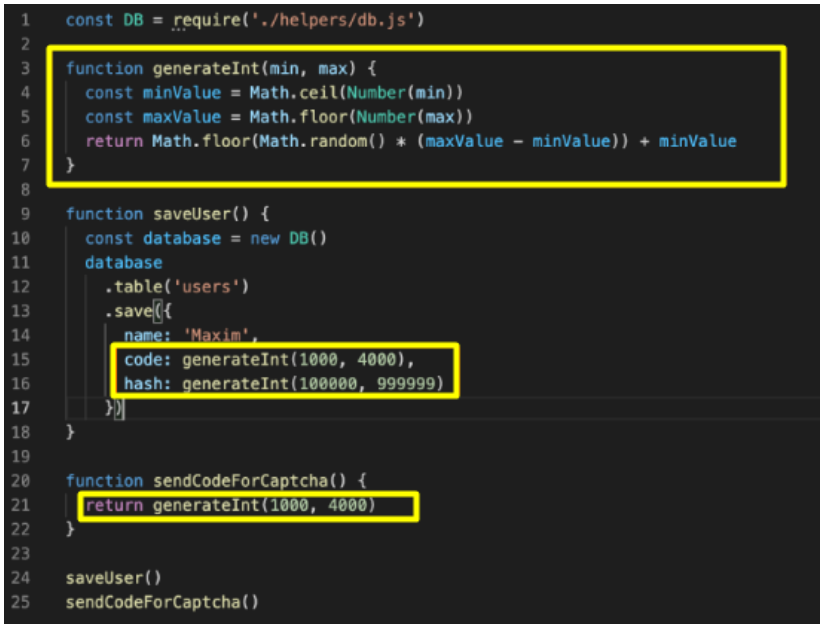


Рис.2 Код після змін

Який і зовнішньо менший, і навіть читається легше, завдяки прописуванні функції генерування чисел та її визову, а не постійного прописування 3 рядків коду для просто генерування чисел.

Схоже, що завдяки використанню таких підходів, дійсно можливо спростити та полегшити написання програм чи навіть створення простих планів, щоб зробити їх коротшими та зрозумілішими для себе та для інших.

*Список використаних джерел*

*1.Е. Хантом,Т. Дейві «Прагматичний програміст»*
*2.Е. Реймонд «Філософія Unix»*
*3.Стаття про принципи програмування. Url:https://blog.vverh.digital/2022/dont-repeat-yourself-or-dry/*

**УДК 004.6:005**

*Потапова Н.А., к.е.н.,*
*доцент, доцент кафедри*
*інформаційних технологій*

## ТЕХНОЛОГІЇ БЛОКЧЕЙН В ІНФОРМАЦІЙНІЙ ЛОГІСТИЦІ

*Донецький національний університет імені Василя Стуса, м. Вінниця*

Однією із вагомих складових інноваційних рішень є впровадження інформаційних технологій, синергетичний ефект в економіці України становить