Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Кафедра інженерії програмного забезпечення в енергеніці

Домашнє завдання № 9

з курсу: "Розробка програмного забезпечення мобільних пристроїв"

**Виконала:**  
студентка 4-го курсу,  
групи ТІ-01  
Круть Катерина Олександрівна

**Перевірив:**

Недашківський О.Л.

Київ 2023/2024

Домашнє завдання № 9

**Завдання:**

Написати клас, який описує студента НТУУ «КПІ» використовуючи наступні

елементи:

* змінні екземпляра
* змінні класу
* public і private методи
* javadoc
* toString

Та клас який використовує клас Student.

**Хід виконання:**

Під час виконання програми, було створено об’єкти таких класів: обʼєкти Студента та обʼєкт Студентської карти

Main.kt

fun main(args: Array<String>) {

// Створюємо об'єкт студента

val student1 = Student(

"Борис",

"Ткаченко",

"Іванович",

21,

4,

"TI-01",

"121",

StudyForm.*FULL\_TIME*

)

*println*(student1)

// Створюємо об'єкт картки студента

val ticketStudent1 = StudentTicket(student1)

*println*(ticketStudent1)

val student2 = Student("Іван", "Іванов", "Іванович", 20, 3, "КН-301", "131", StudyForm.*FULL\_TIME*)

val student3 = Student("Іван", "Іванов", "Іванович", 20, 3, "КН-301", "131", StudyForm.*FULL\_TIME*)

val student4 = Student("Петро", "Петров", "Петрович", 21, 2, "КН-202", "122", StudyForm.*PART\_TIME*)

*println*(student2 == student3)

*println*(student2 == student4)

}

Student.kt

import java.time.LocalDate

*/\*\**

*\* Клас для збереження типів форм навчання*

*\*/*

enum class StudyForm {

*FULL\_TIME*,

*PART\_TIME*

}

*/\*\**

*\* Клас "Студент"*

*\**

*\* Цей клас описує інформацію про студента та надає методи для роботи з цією інформацією.*

*\**

*\* Параметри:*

*\** ***@param*** *[\_id] Параметр, який зберігає айді студента.*

*\** ***@param*** *[name] Параметр, який зберігає ім'я студента.*

*\** ***@param*** *[surname] Параметр, який зберігає прізвище студента.*

*\** ***@param*** *[patronymic] Параметр, який зберігає по-батькові студента.*

*\** ***@param*** *[age] Параметр, який зберігає вік студента.*

*\** ***@param*** *[course] Параметр, який зберігає курс студента.*

*\** ***@param*** *[group] Параметр, який зберігає групу студента.*

*\** ***@param*** *[specialty] Параметр, який зберігає спеціальність студента.*

*\** ***@param*** *[formOfStudying] Параметр, який зберігає форму навчання студента.*

*\**

*\* Функції:*

*\** ***@param*** *getFullName Функція, яка повертає повне ім'я студента.*

*\** ***@param*** *calculateTicketExpiryDate Функція, яка рахує дату закінчення дії студентського квитка.*

*\** ***@param*** *checkAge Приватна функція, яка перевіряє правильність віку студента.*

*\** ***@param*** *checkCourse Приватна функція, яка перевіряє правильність курсу студента.*

*\** ***@param*** *checkName Приватна функція, яка перевіряє правильність імені студента.*

*\** ***@param*** *checkSurname Приватна функція, яка перевіряє правильність прізвища студента.*

*\** ***@param*** *checkPatronymic Приватна функція, яка перевіряє правильність по-батькові студента.*

*\** ***@param*** *checkSpecialty Приватна функція, яка перевіряє правильність спеціальності студента.*

*\** ***@param*** *getFormOfStudying Функція, яка повертає форму навчання студента.*

*\** ***@param*** *getSpecialtyName Приватна функція, яка повертає назву спеціальності за її кодом.*

*\** ***@param*** *equals Функція, яка допомагає зрівнювати об'єкти класу.*

*\** ***@param*** *hashCode Функція, яка надає об'єкту хеш-код.*

*\** ***@param*** *toString Функція, яка повертає повний опис об'єкту.*

*\*/*

class Student {

*/\*\**

*\* Параметр який зберігає айді студента*

*\* \*/*

var \_id: Long = 0

*/\*\**

*\* Параметр який зберігає ім'я студента*

*\* \*/*

private lateinit var name: String

*/\*\**

*\* Параметр який який зберігає прізвище студента*

*\* \*/*

private lateinit var surname: String

*/\*\**

*\* Параметр який який зберігає по-батькові студента*

*\* \*/*

private lateinit var patronymic: String

*/\*\**

*\* Параметр який який зберігає вік студента*

*\* \*/*

private var age: Int = 17

*/\*\**

*\* Параметр який який зберігає курс студента*

*\* \*/*

private var course: Int = 1

*/\*\**

*\* Параметр який який зберігає групу студента*

*\* \*/*

private var group: String

*/\*\**

*\* Параметр який який зберігає спеціальність студента*

*\* \*/*

private lateinit var specialty: String

private var formOfStudying: StudyForm = StudyForm.*FULL\_TIME*

// Змінні екземпляра класу

companion object {

*/\*\**

*\* Всі дсотупні для факультета ІАТЕ спеціальності*

*\*/*

val specialties: Map<String, String> = *mapOf*( // Змінено тип ключа на String

"023" *to* "ОБРАЗОТВОРЧЕ МИСТЕЦТВО, ДЕКОРАТИВНЕ МИСТЕЦТВО, РЕСТАВРАЦІЯ",

"035" *to* "ФІЛОЛОГІЯ",

"051" *to* "ЕКОНОМІКА",

"053" *to* "ПСИХОЛОГІЯ",

"054" *to* "СОЦІОЛОГІЯ",

"061" *to* "ЖУРНАЛІСТИКА",

"073" *to* "МЕНЕДЖМЕНТ",

"075" *to* "МАРКЕТИНГ",

"081" *to* "ПРАВО",

"101" *to* "ЕКОЛОГІЯ",

"104" *to* "ФІЗИКА ТА АСТРОНОМІЯ",

"105" *to* "ПРИКЛАДНА ФІЗИКА ТА НАНОМАТЕРІАЛИ",

"111" *to* "МАТЕМАТИКА",

"113" *to* "ПРИКЛАДНА МАТЕМАТИКА",

"121" *to* "ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ",

"122" *to* "КОМП’ЮТЕРНІ НАУКИ",

"123" *to* "КОМП’ЮТЕРНА ІНЖЕНЕРІЯ",

"124" *to* "СИСТЕМНИЙ АНАЛІЗ",

"125" *to* "КІБЕРБЕЗПЕКА ТА ЗАХИСТ ІНФОРМАЦІЇ",

"126" *to* "ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ",

"131" *to* "ПРИКЛАДНА МЕХАНІКА",

"132" *to* "МАТЕРІАЛОЗНАВСТВО",

"133" *to* "ГАЛУЗЕВЕ МАШИНОБУДУВАННЯ",

"134" *to* "АВІАЦІЙНА ТА РАКЕТНО-КОСМІЧНА ТЕХНІКА",

"136" *to* "МЕТАЛУРГІЯ",

"141" *to* "ЕЛЕКТРОЕНЕРГЕТИКА, ЕЛЕКТРОТЕХНІКА ТА ЕЛЕКТРОМЕХАНІКА",

"142" *to* "ЕНЕРГЕТИЧНЕ МАШИНОБУДУВАННЯ",

"143" *to* "АТОМНА ЕНЕРГЕТИКА",

"144" *to* "ТЕПЛОЕНЕРГЕТИКА",

"161" *to* "ХІМІЧНІ ТЕХНОЛОГІЇ ТА ІНЖЕНЕРІЯ",

"162" *to* "БІОТЕХНОЛОГІЇ ТА БІОІНЖЕНЕРІЯ",

"163" *to* "БІОМЕДИЧНА ІНЖЕНЕРІЯ",

"171" *to* "ЕЛЕКТРОНІКА",

"172" *to* "ЕЛЕКТРОННІ КОМУНІКАЦІЇ ТА РАДІОТЕХНІКА",

"173" *to* "АВІОНІКА",

"174" *to* "АВТОМАТИЗАЦІЯ, КОМП’ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ ТА РОБОТОТЕХНІКА",

"175" *to* "ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНІ ТЕХНОЛОГІЇ",

"176" *to* "МІКРО- ТА НАНОСИСТЕМНА ТЕХНІКА",

"184" *to* "ГІРНИЦТВО",

"186" *to* "ВИДАВНИЦТВО ТА ПОЛІГРАФІЯ",

"227" *to* "ТЕРАПІЯ ТА РЕАБІЛІТАЦІЯ",

"231" *to* "СОЦІАЛЬНА РОБОТА",

"281" *to* "ПУБЛІЧНЕ УПРАВЛІННЯ ТА АДМІНІСТРУВАННЯ"

)

}

*/\*\**

*\** ***@constructor*** *Створює об'єкт студента*

*\* Функція отримує такі параметри:*

*\** ***@param*** *[\_id] Параметр який зберігає айді студента*

*\** ***@param*** *[surname] Параметр який зберігає прізвище студента*

*\** ***@param*** *[name] Параметр який який зберігає ім'я студента*

*\** ***@param*** *[age] Параметр який який зберігає вік студента*

*\** ***@param*** *[course] Параметр який який зберігає курс студента*

*\** ***@param*** *[group] Параметр який який зберігає групу студента*

*\** ***@param*** *[specialty] Параметр який який зберігає спеціальність студента*

*\** ***@property*** *<examinationAge> Функція яка перевіряє правильність віку студента*

*\** ***@property*** *<examinationCourse> Функція яка перевіряє правильність курсу студента*

*\** ***@property*** *<hashCode> Функція яка надає об'єкту хеш-код*

*\* \*/*

constructor(

name: String,

surname: String,

patronymic: String,

age: Int,

course: Int,

group: String,

specialty: String,

formOfStudying: StudyForm

) {

checkName(name)

checkSurname(surname)

checkPatronymic(patronymic)

checkCourse(course)

checkAge(age)

this.group = group

checkSpecialty(specialty)

this.formOfStudying = formOfStudying

\_id = hashCode().toLong()

}

*/\*\**

*\** ***@constructor*** *Створює об'єкт студента*

*\* Функція отримує такі параметри:*

*\** ***@param*** *[student] Параметр який є об'єктом класу і переписує дані цього об'єкту в новий об'єкт*

*\* \*/*

constructor(student: Student) {

this.name = student.name

this.surname = student.surname

this.patronymic = student.patronymic

this.age = student.age

this.group = student.group

this.course = student.course

this.specialty = student.specialty

this.formOfStudying = student.formOfStudying

}

*/\*\**

*\* Перевіряє, чи є ім'я студента вірним.*

*\** ***@param*** *name Ім'я студента, яке потрібно перевірити.*

*\** ***@throws*** *IllegalArgumentException якщо ім'я коротше за 2 символи або довше за 100 символів.*

*\*/*

private fun checkName(name: String) {

*require*(name.length in 2..100) **{** "Ім'я повинно містити від 2 до 100 символів" **}**

this.name = name

}

*/\*\**

*\* Перевіряє, чи є прізвище студента вірним.*

*\** ***@param*** *surname Прізвище студента, яке потрібно перевірити.*

*\** ***@throws*** *IllegalArgumentException якщо прізвище коротше за 2 символи або довше за 100 символів.*

*\*/*

private fun checkSurname(surname: String) {

*require*(surname.length in 2..100) **{** "Прізвище повинно містити від 2 до 100 символів" **}**

this.surname = surname

}

*/\*\**

*\* Перевіряє, чи є по-батькові студента вірним.*

*\** ***@param*** *patronymic По-батькові студента, яке потрібно перевірити.*

*\** ***@throws*** *IllegalArgumentException якщо по-батькові коротше за 2 символи або довше за 100 символів.*

*\*/*

private fun checkPatronymic(patronymic: String) {

*require*(patronymic.length in 2..100) **{** "По-батькові повинно містити від 2 до 100 символів" **}**

this.patronymic = patronymic

}

*/\*\**

*\* Функція яка повертає повне ім'я студента*

*\** ***@return*** *імʼя призвище по-батькові*

*\* \*/*

fun getFullName(): String {

return "$surname $name $patronymic"

}

*/\*\**

*\* Функція яка повертає форму навачання студента*

*\** ***@return*** *строка з назвою форми навачання*

*\* \*/*

fun getFormOfStudying(): String = if (formOfStudying == StudyForm.*FULL\_TIME*) "денна" else "заочна"

*/\*\**

*\* Функція для підрахунку дати кінця навчання / дати, до якої дійсний студентський квиток*

*\** ***@return*** *дата кінця навчання*

*\* \*/*

fun calculateTicketExpiryDate(): String {

// val monthsLeft = ChronoUnit.MONTHS.between(today, expiryDate)

val yearsLeft = if (this.course > 3) 1 else 4 - this.course

return "${LocalDate.of(LocalDate.now().*year* + yearsLeft, 6, 30)}"

}

*/\*\**

*\* Функція яка перевіряє правильність віку студента*

*\** ***@param*** *[age] Параметр який отримує функція є вік який перевіряється*

*\** ***@return*** *вік студента*

*\** ***@throws*** *IllegalArgumentException якщо вік не відповідає можливому віку студента відповідного курсу.*

*\* \*/*

private fun checkAge(age: Int) {

*require*(age >= 16 + this.course) **{** "\nВік студента не може бути меньше ${16 + this.course} для ${this.course} курсу" **}**

this.age = age

}

*/\*\**

*\* Функція яка перевіряє правильність курсу студента*

*\** ***@param*** *[course] Параметр який отримує функція є курс який перевіряється*

*\** ***@return*** *курс студента*

*\** ***@throws*** *IllegalArgumentException якщо курс меньше 1 і більше 5*

*\* \*/*

private fun checkCourse(course: Int) {

*require*(course in 1..5) **{** "\nIncorrect Data. Course must be in [1...5]" **}**

this.course = course

}

*/\*\**

*\* Перевіряє, чи є код спеціальності вірним.*

*\** ***@param*** *specialtyCode Код спеціальності, який потрібно перевірити.*

*\** ***@throws*** *IllegalArgumentException якщо код спеціальності не існує в мапі спеціальностей.*

*\*/*

private fun checkSpecialty(specialtyCode: String) {

*require*(specialties.containsKey(specialtyCode)) **{** "Недопустимий код спеціальності: $specialtyCode" **}**

this.specialty = specialtyCode

}

*/\*\**

*\* Повертає назву спеціальності за її кодом.*

*\** ***@param*** *specialtyCode Код спеціальності, для якої потрібно отримати назву.*

*\** ***@return*** *Назва спеціальності, якщо вона існує, або "Невідома спеціальність", якщо такої спеціальності не існує.*

*\*/*

private fun getSpecialtyName(specialtyCode: String): String {

return specialties[specialtyCode] ?: "Невідома спеціальність"

}

*/\*\**

*\* Функція яка допомогає зрівнювати об'єкти класу*

*\* Використовуються усі параметри класу*

*\** ***@return*** *Функція повертає булиан (об'єкти схожі чи ні)*

*\* \*/*

override fun equals(other: Any?): Boolean {

return when {

this === other -> true

*javaClass* != other?.*javaClass* -> false

else -> {

other as Student

name == other.name &&

surname == other.surname &&

patronymic == other.patronymic &&

age == other.age &&

course == other.course &&

group == other.group &&

specialty == other.specialty &&

formOfStudying == other.formOfStudying

}

}

}

*/\*\**

*\* Функція яка надає об'єкту хеш-код*

*\* Використовуються усі параметри класу*

*\** ***@receiver*** *Функція має публічний доступ*

*\** ***@return*** *Функція повертає число (хеш-код об'єкту)*

*\* \*/*

override fun hashCode(): Int {

return *listOf*(name, surname, patronymic, age, course, group, specialty, formOfStudying).hashCode()

}

*/\*\**

*\* Функція яка повертає повний опис об'єкту*

*\* Використовуються усі параметри класу*

*\** ***@receiver*** *Функція має публічний доступ*

*\** ***@return*** *Функція яка повертає повний опис об'єкту*

*\* \*/*

override fun toString(): String {

return "Student(\_id=$\_id, name='$name', surname='$surname', patronymic='$patronymic', age=$age, " +

"course=$course, group='$group', specialty='$specialty - ${getSpecialtyName(specialty)}')"

}

}

StudentTicket.kt

*/\*\**

*\* Класс "Студентський квиток"*

*\**

*\* Цей клас описує інформацію про картку студента*

*\** ***@param*** *[\_id] Параметр який зберігає айді картки*

*\** ***@param*** *[info] Параметр який зберігає опис картки студента*

*\** ***@param*** *[student] Параметр який зберігає об'єкт студента*

*\**

*\** ***@param*** *equals Функція яка допомогає зрівнювати об'єкти класу*

*\** ***@param*** *hashCode Функція яка надає об'єкту хеш-код*

*\** ***@param*** *toString Функція яка повертає повний опис об'єкту*

*\**

*\** ***@constructor*** *використовується вбудований конструктор який отримує:*

*\** ***@param*** *[student] Параметр який зберігає об'єкт студента*

*\* \*/*

class StudentTicket(var student: Student) {

*/\*\**

*\* Параметр який зберігає айді картки*

*\** ***@receiver*** *Параметр має публічний доступ*

*\* \*/*

var \_id: Long = student.\_id

*/\*\**

*\* Параметр який зберігає назву університета студента*

*\* \*/*

var nameUniversity: String = "Київський Політехнічний Інститут ім. Ігоря Сікорського"

*/\*\**

*\* Параметр який зберігає назву факультета студента*

*\** ***@receiver*** *Параметр має публічний доступ*

*\* \*/*

var Faculty: String = "ІАТЕ"

*/\*\**

*\* Параметр який зберігає опис картки студента*

*\** ***@receiver*** *Параметр має публічний доступ*

*\* \* \*/*

var info: String =

"Студентський квиток\n${student.getFullName().*replace*(" ", "\n")}\n" +

"$nameUniversity\n${Faculty}\nДійсний до: ${student.calculateTicketExpiryDate()}\n" +

"Форма навчання: ${student.getFormOfStudying()}"

*/\*\**

*\* Функція яка повертає повний опис об'єкту*

*\* Використовуються усі параметри класу*

*\** ***@receiver*** *Функція має публічний доступ*

*\** ***@return*** *Функція яка повертає повний опис об'єкту*

*\* \*/*

override fun toString(): String {

return "studentCard(\_id=$\_id, student=$student, \ninfo='$info')"

}

*/\*\**

*\* Функція яка допомогає зрівнювати об'єкти класу*

*\* Використовуються усі параметри класу*

*\** ***@receiver*** *Функція має публічний доступ*

*\** ***@return*** *Функція повертає булиан (об'єкти схожі чи ні)*

*\* \*/*

override fun equals(other: Any?): Boolean {

if (this === other) return true

if (other !is StudentTicket) return false

return \_id == other.\_id &&

student == other.student &&

info == other.info

}

*/\*\**

*\* Функція яка надає об'єкту хеш-код*

*\* Використовуються усі параметри класу*

*\** ***@receiver*** *Функція має публічний доступ*

*\** ***@return*** *Функція повертає число (хеш-код об'єкту)*

*\* \*/*

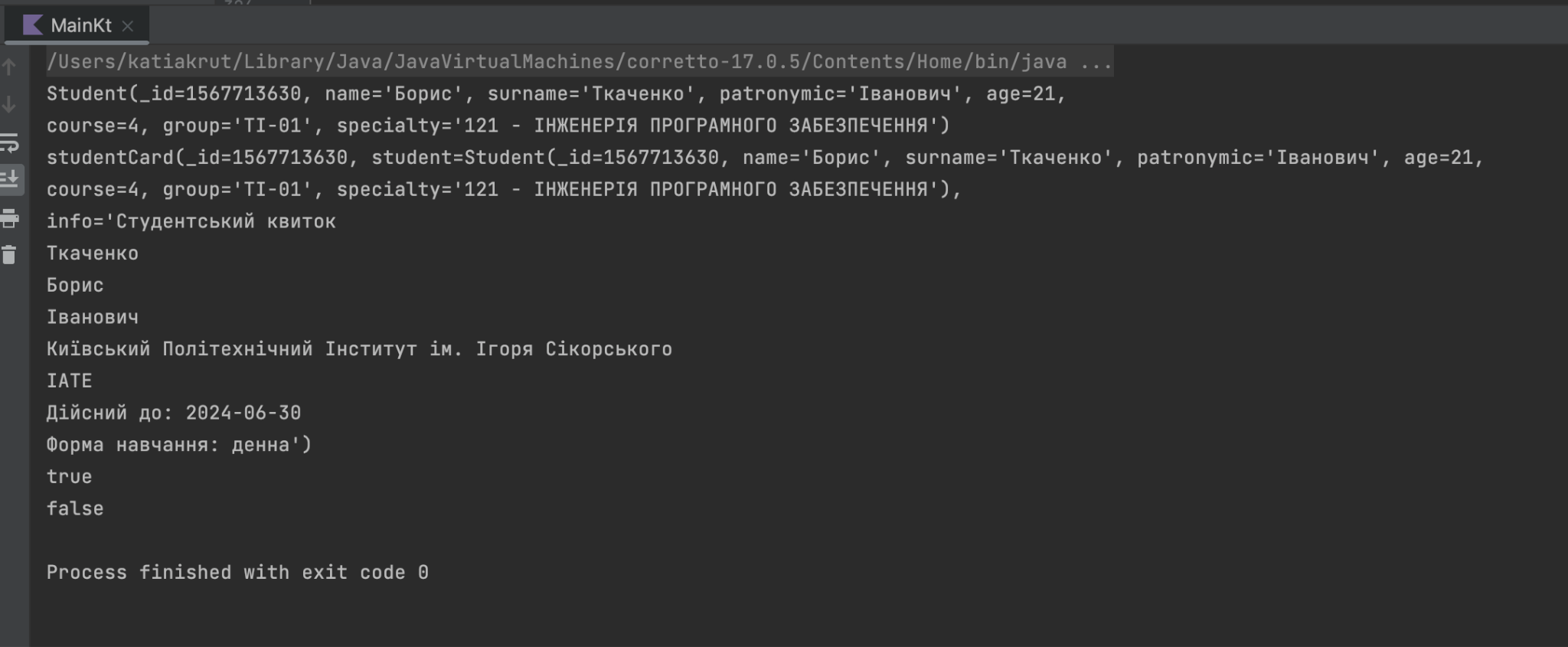
override fun hashCode(): Int {

return *listOf*(student, \_id, info).hashCode()

}

}

Виконання:



**Висновок**

В результаті виконання цього домашнього завдання було освоєно основні принципи роботи с мовою Kotlin та використано їх на практиці. Найбільшу увагу в цій роботі було приділено класам, їх параметрам та методам, створенню їх обʼєктів.