Лабораторна робота №3

**Тема:** Базові принципи ООП

**Мета:** Використовуючи теоретичне підґрунтя про базові принципи ООП та підходи використання в Java виконати дії що будуть вказано в завданні до лабораторної роботи.

Хід роботи

Теоретичні відомості та завдання до лабораторної роботи:

У цьому завданні ви формулюватимете бізнес-логіку планування спринту. Вам належить організувати планування спринту, що включає завдання, які вам потрібно реалізувати на основі базового класу *Ticket*. Врахуйте, у цьому випадку спринт приймає лише два підтипи класу *Ticket:Bug* та *UserStory*.

**Ticket**

Кожен *ticket* має свій ідентифікатор, ім'я та приблизний час для його виконання. Ці значення надаються через конструктор класу *Ticket*.

*Ticket* може вважатися завершеним або незавершеним. Щойно створений *ticket* вважається незавершеним.

* *getId*() - повертає ідентифікатор ticket'у.
* *getName*() - повертає ім'я ticket'у.
* *getEstimate*() - повертає оцінку ticket'у.
* *isCompleted*() - повертає *true*, якщо ticket завершено, інакше — *false*.
* *complete*() - переводить ticket у завершений стан.

**UserStory**

*UserStory* (історія користувача) представляє ticket, який може містити залежності. Залежності — це інші екземпляри *UserStory*, які мають бути завершені в першу чергу, щоб потім вдалося завершити і залежне від них *UserStory*. Залежності передаються через конструктор класу *UserStory*.

* *complete*() - як і метод Ticket#complete() - переводить ticket у завершений стан. Різниця лише в тому, що історія користувача може не бути завершена, якщо його залежність ще не завершена.
* *getDependencies*() - повертає копію масиву ticket'ів на спринт, що захищена від змін.
* *toString*() - повертає рядок, що представляє *UserStory*, використовуючи його ідентифікатор та ім'я. Приклад: з ідентифікатором =1, ім’я *UserStory* = "Об'єкт реєстрації користувача", отримаємо "[US 1] Об'єкт реєстрації користувача".

**Bug**

Багом вважається ticket, що відноситься до завершеної UserStory. Баги не існують самі по собі, без відповідного екземпляра UserStory.

* *createBug*(int id, String name, int estimate, UserStory userStory) - статичний метод створення екземпляра Bug. Повертає null, якщо UserStory має значення null або не завершено. Інакше повертає створений екземпляр Bug.
* toString() - повертає рядкове представлення цього бага, використовуючи ідентифікатор, ім'я та ім'я відповідної UserStory. Приклад: з ідентифікатором = 2, ім'ям бага = "Додати пароль, що повторюється" і ім'ям відповідної UserStory = "Форма реєстрації" в результаті отримаємо "[Bug 2] Форма реєстрації: Додати повторюваний пароль".

**Sprint**

Спринти мають тимчасову ємність та обмеження кількості ticket'ів, які задаються через конструктор. Спринт не може містити тікет з сумарною оцінкою часу виконання, що перевищує тимчасову ємність спринту.

Не допускається, щоб спринт містив більше тикетів, ніж визначено обмеженням кількості тикетів для цього спринту. Спринт повинен приймати ticket'и за допомогою методів *add*\*. Ці методи повертають *true*, коли вхідний ticket був прийнятий у спринт, і *false* в іншому випадку.

Спринт не повинен приймати

1. Значення null.
2. ticket'и, які вже завершені.
3. ticket'и, що мають значення оцінки часу виконання, яке, у разі додавання ticket'а, призведе до переповнення тимчасової ємності спринту.
4. Будь-який ticket, якщо досягнуто межі кількості ticket'ів у спринті.

Опис методів спринта

* *addUserStory*(UserStory userStory) - приймає userStory, якщо вона не дорівнює null, і не завершена. Повертає true, якщо історія користувача прийнята, у противному випадку false.
* addBug(Bug bugReport) - приймає bug, якщо він не дорівнює null і не завершений. Повертає true, якщо баг прийнято, інакше false.
* getTickets() - повертає захищену копію масиву ticket'ів на спринт. Переконайтеся, що ticket'и розташовані в тому ж порядку, як вони були прийняті в спринт.

getTotalEstimate() - повертає суму оцінок часу виконання всіх ticket'ів, прийнятих на спринт.

Код програми

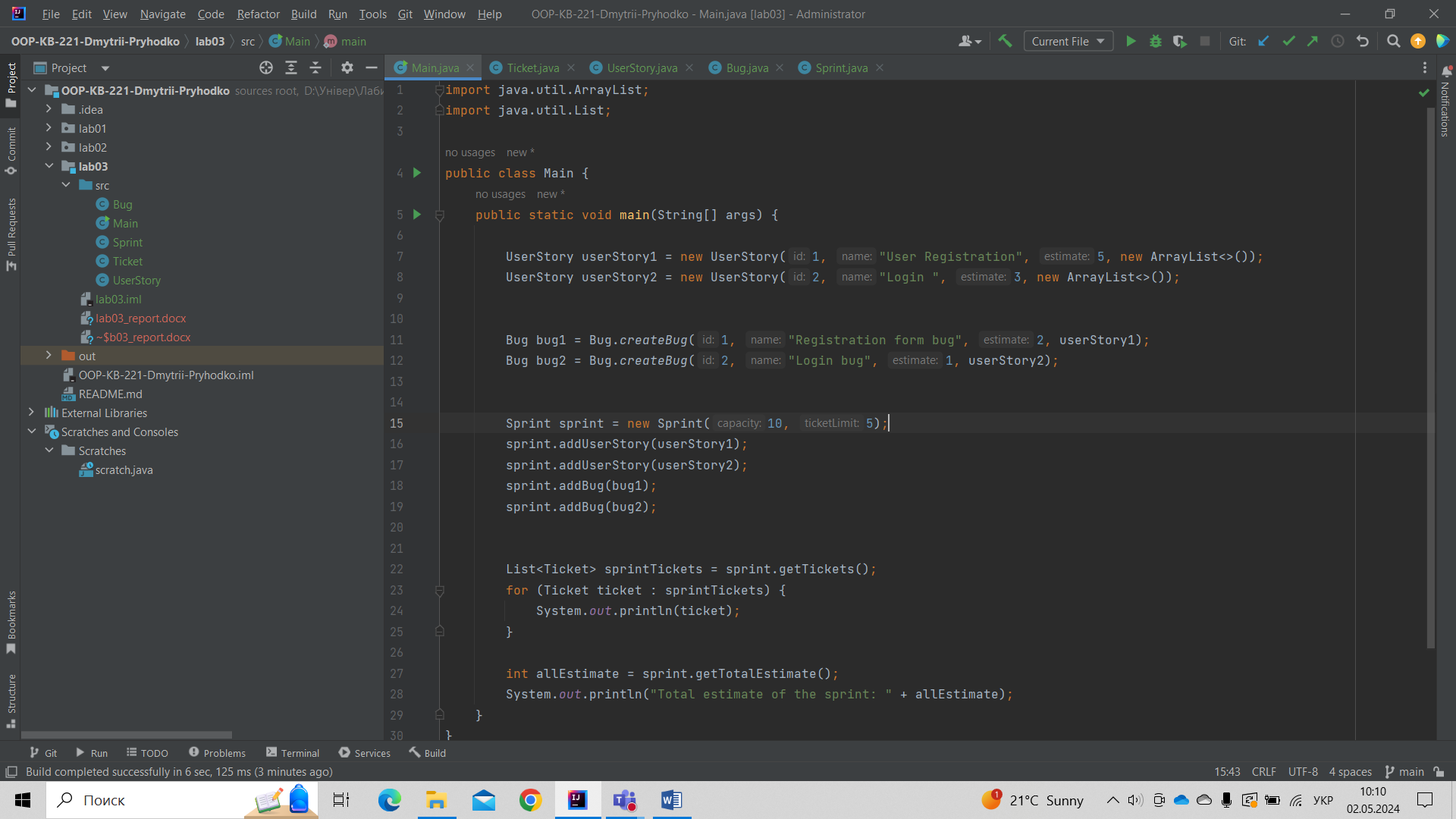


Рис.1 - Main

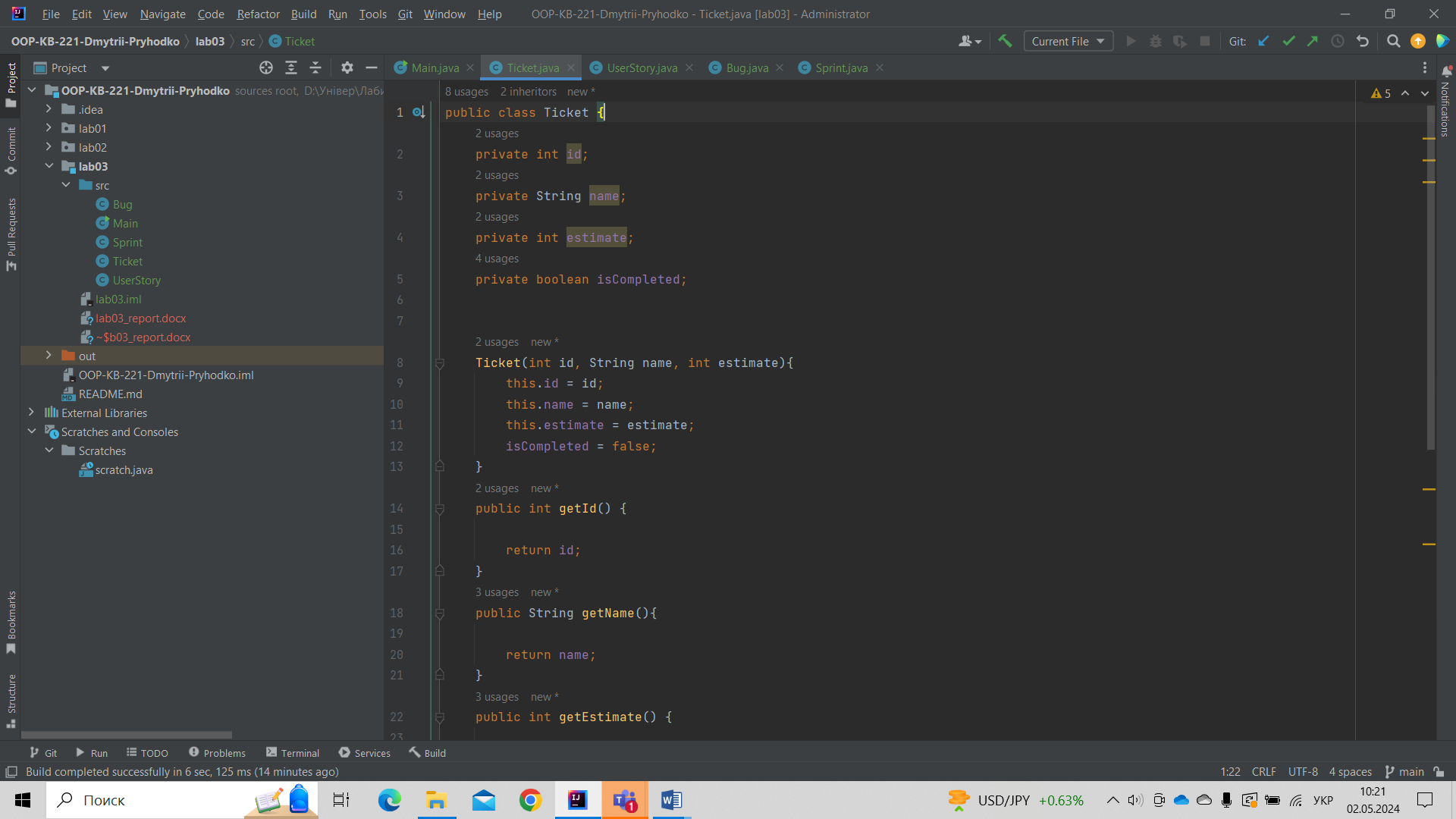


Рис.2 – клас Ticket

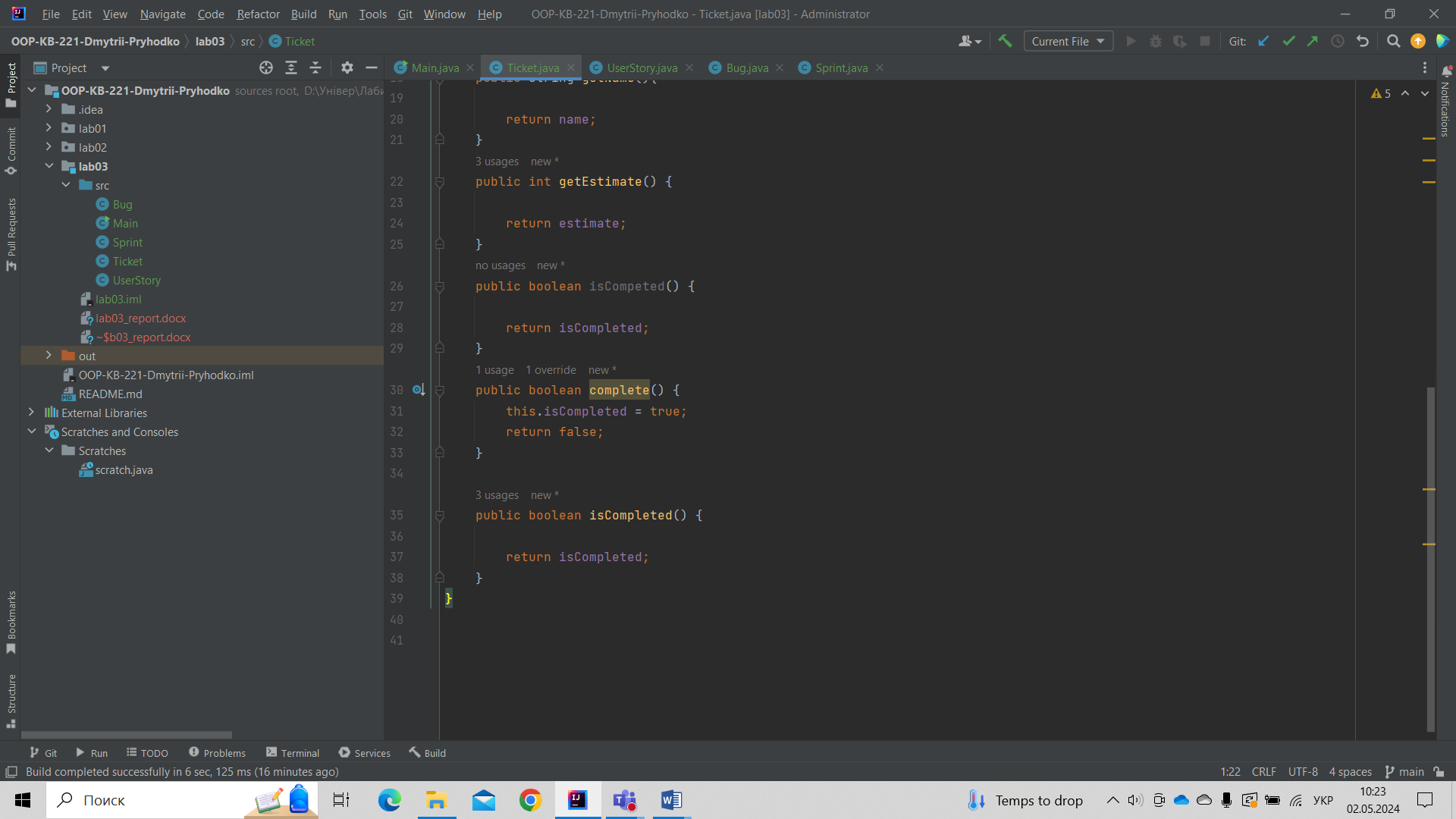


Рис.3 – клас Ticket

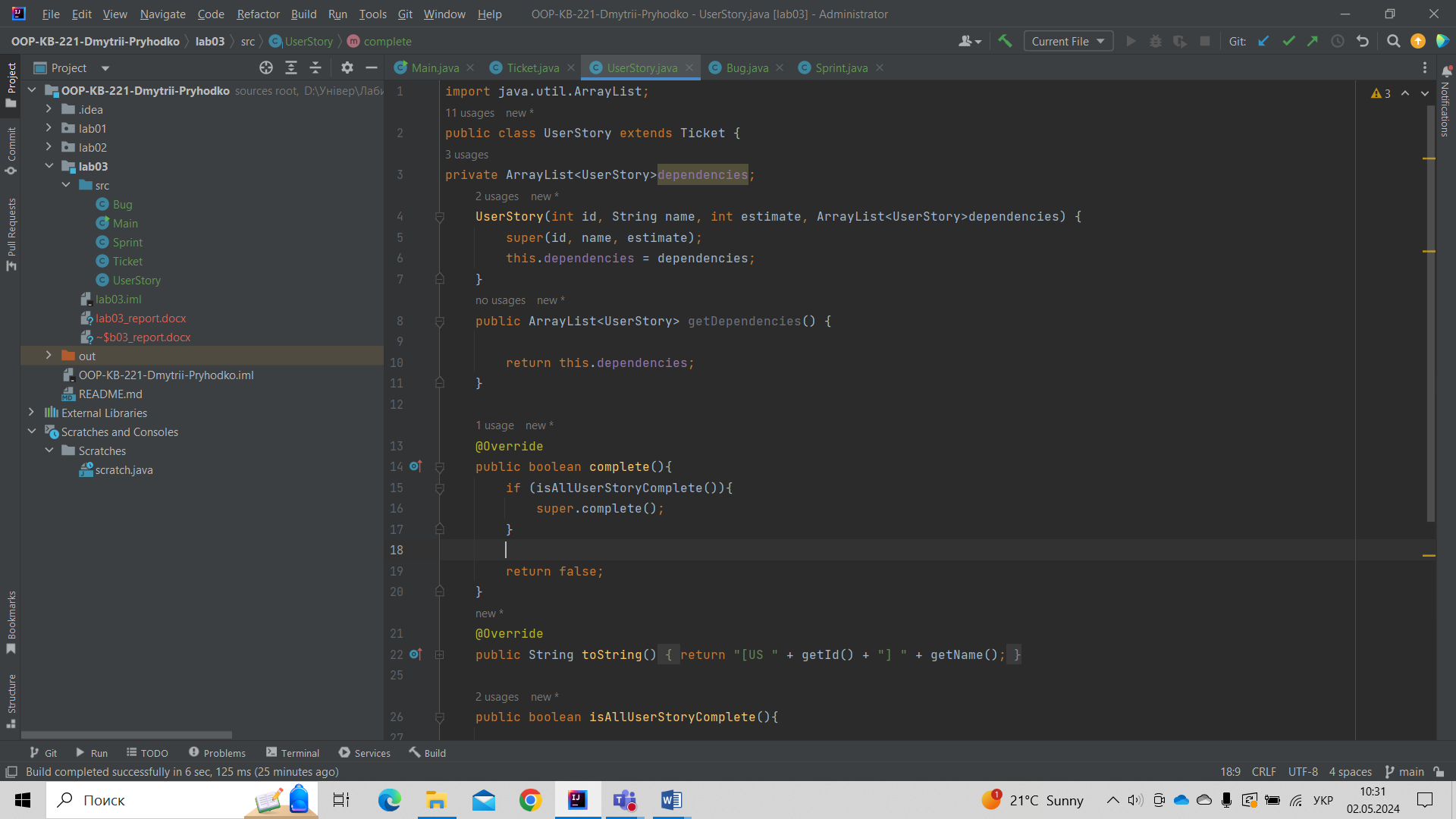


Рис.4 – клас UserStory

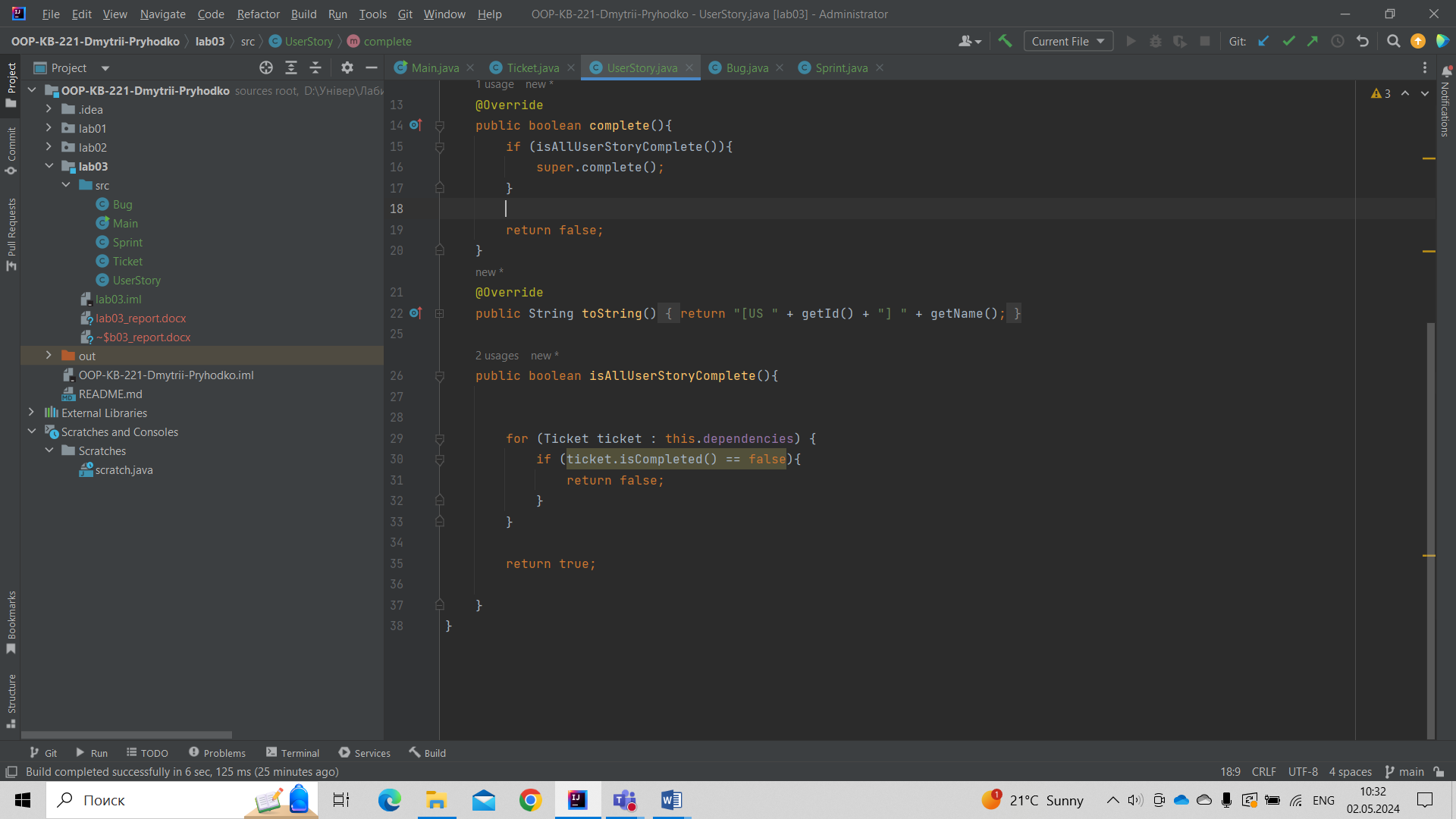


Рис.5 – клас UserStory

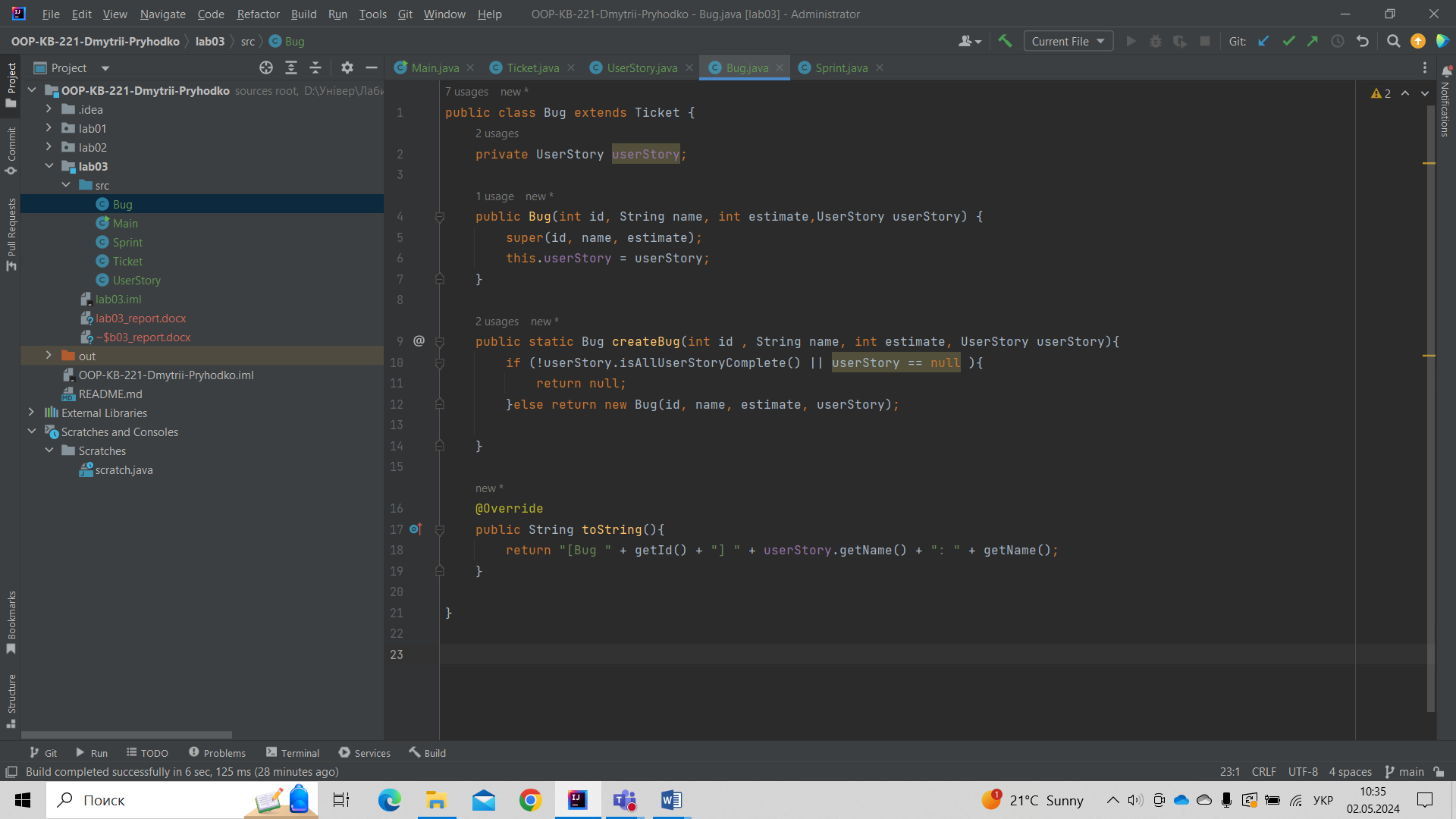


Рис.6 – клас Bug

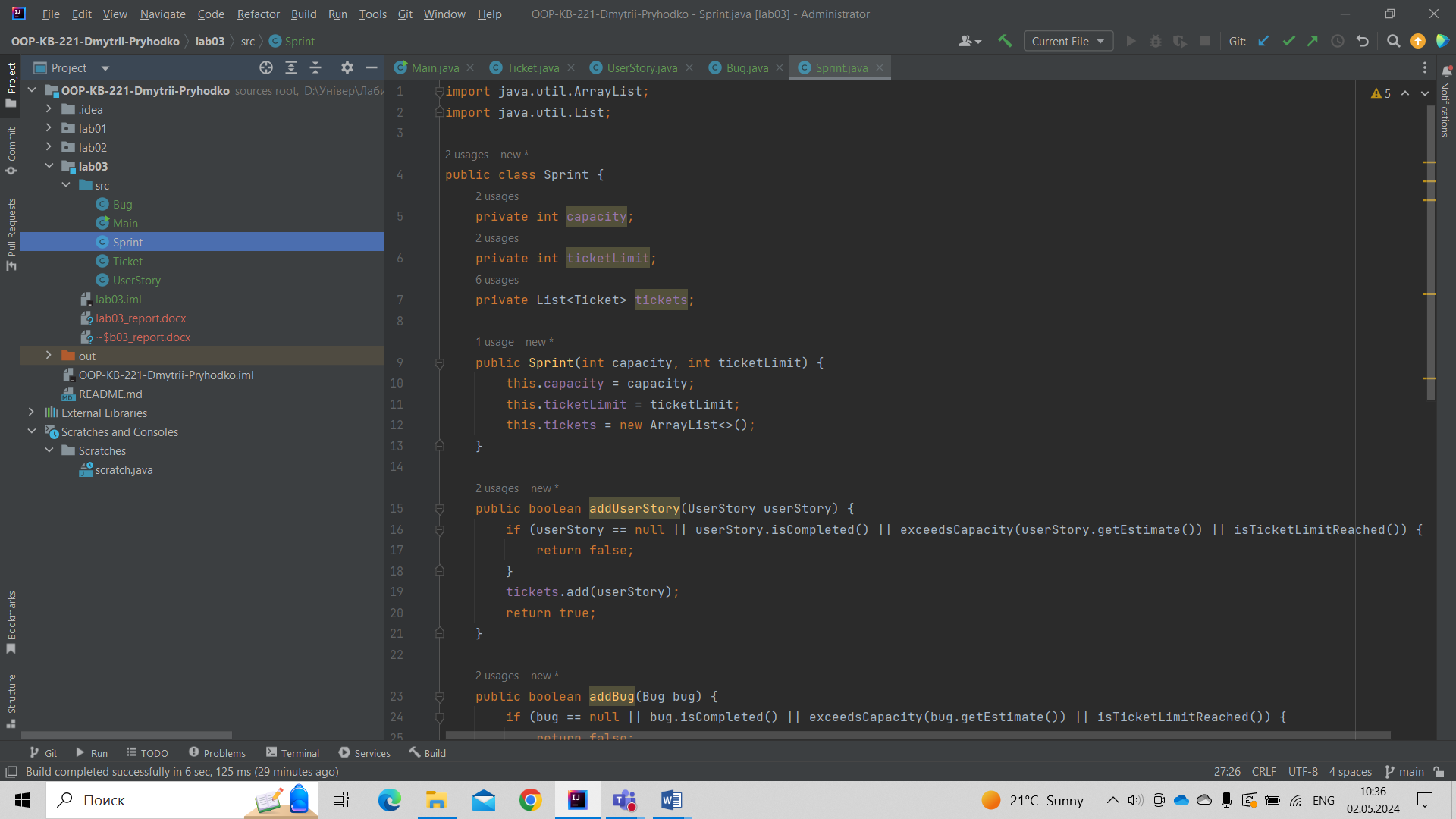


Рис.7 – клас Sprint

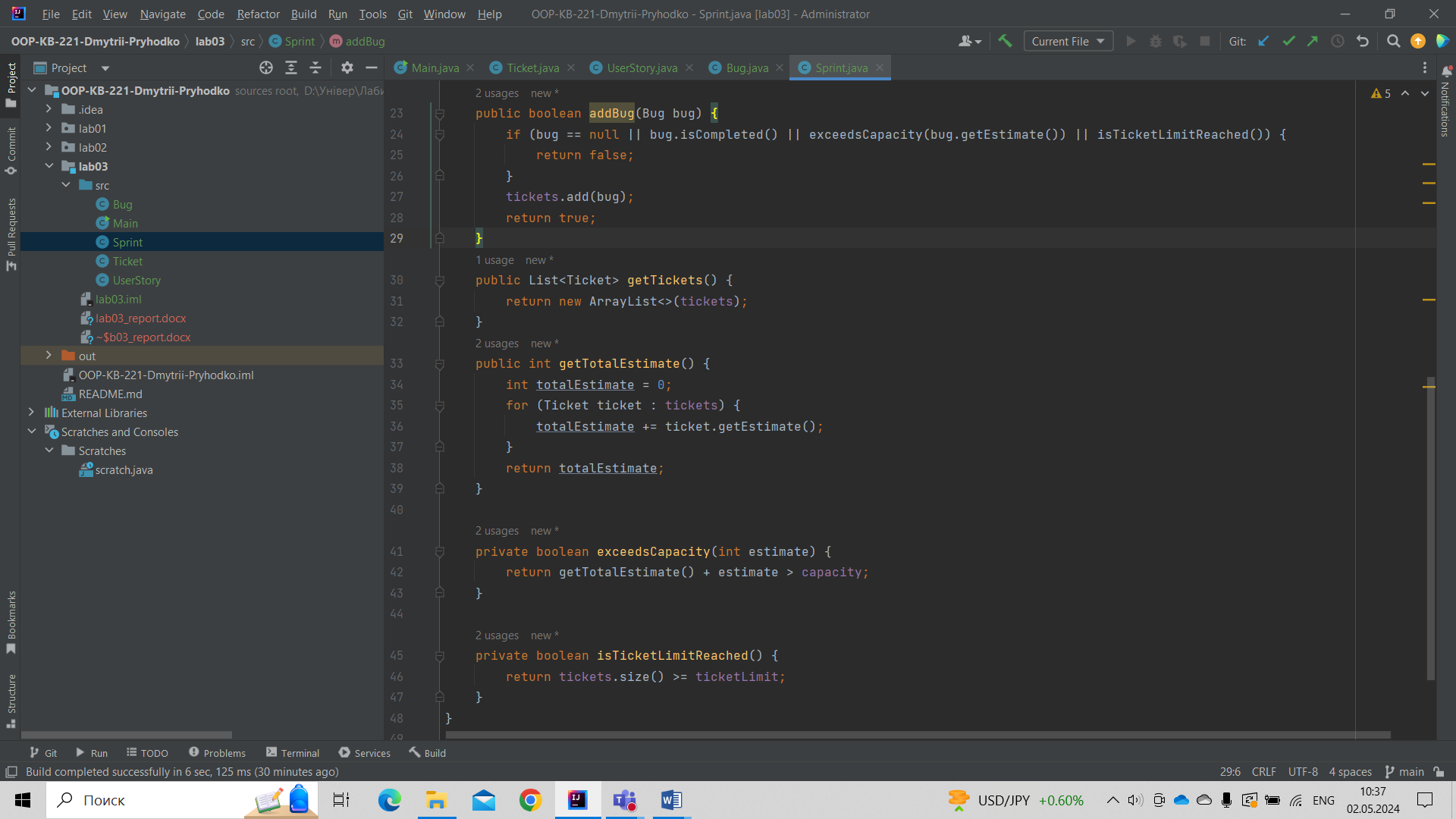


Рис.8 – клас Sprint

**Висновок:** На цій лабораторній роботі я ознайомлювався з базовими принципами ООП та використовував їх у завданні до лабораторної роботи.