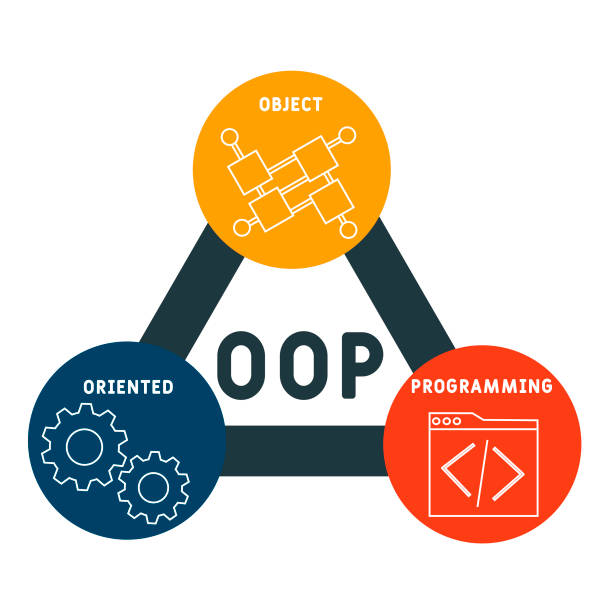
**Lesson 18. Object-oriented programming**

Нарешті ми підійшли до того, що є одним із ключових пунктів у вивченні Java, а саме – **об’єктно-орієнтоване програмування**, або ж коротко **ООП**.

Під ООП розуміють такий підхід до створення програми, при якому програма складається із окремих об’єктів, подібно до того як реальний світ складається з речей.

ООП дозволяє перетворити програмні ідеї в маленькі будівельні блоки. Програма стає схожою на живий організм, де органи складаються з тканин, тканини з клітин і т.д.

Що таке об’єкти, як їх створювати та використовувати нам ще належить з’ясувати. То ж, до роботи!

**Objects**

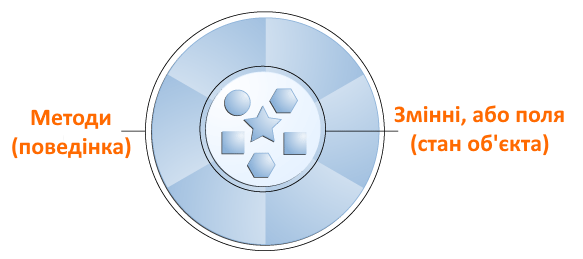
**Об’єкт** – це ключове поняття, на якому базується ООП (як можна зрозуміти із назви), а отже дуже важливо зрозуміти, що вони собою являють. Об’єкти у програмі подібні до об’єктів у реальному світі – звичайних речей, що нас оточують: стіл, олівець, піца тощо.

Речі (об’єкти) у реальному світі мають можна описати за допомогою двох характеристик: усі вони мають **стан** і **поведінку**. Столи, наприклад, мають стан (кількість ніжок, форму дошки, висоту тощо) і поведінку (стояти, утримувати на собі речі), якою нудною вона б не здавалась. Олівці теж мають стан (колір, довжина, жирність тощо) і поведінку (малювати, писати, гострити).

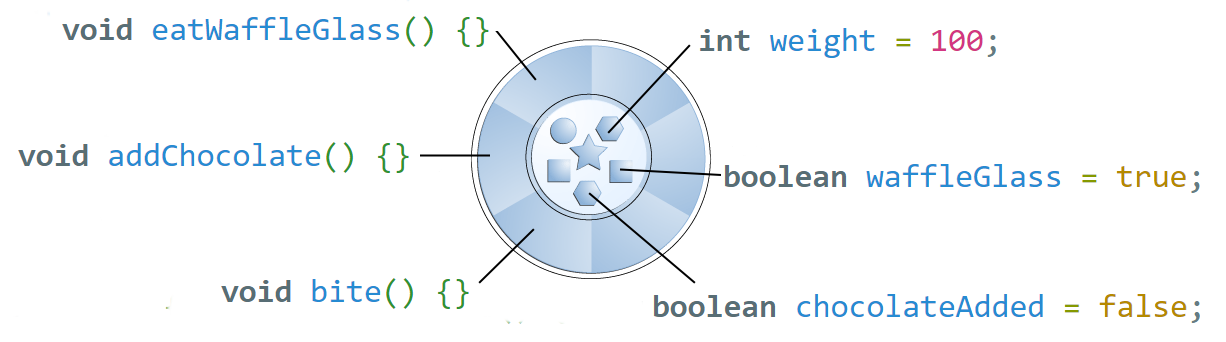
**Завдання.** Розгляньте милих створінь на рисунку нижче, та спробуйте записати якомога більше пунктів, що опишуть їх стан і поведінку:

****

Подібним чином, можемо розглядати і об’єкти у ООП. **Об’єкт у програмуванні** – це сутність, яка поєднує в собі методи і змінні. Методи – описують поведінку об’єкта, а змінні (інколи поля даних, або атрибути даних) – його стан.



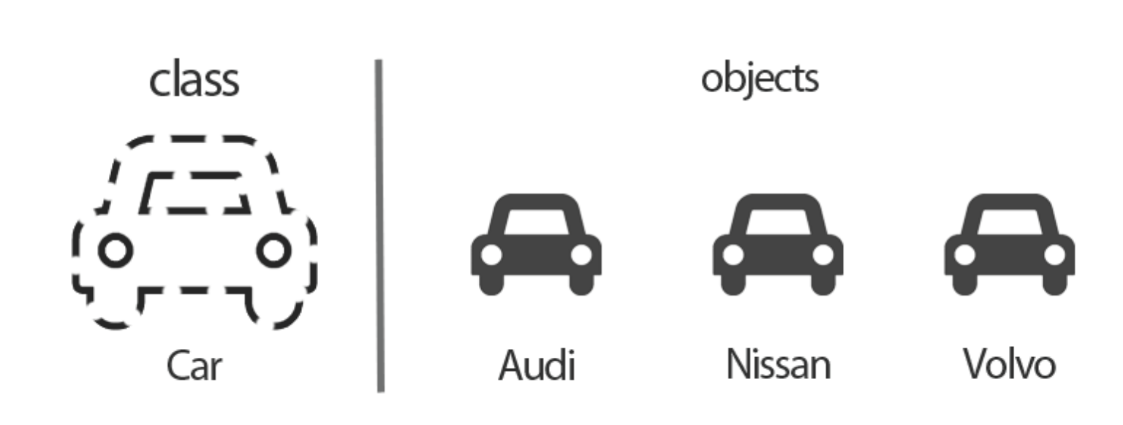
Наприклад якби ми створили об’єкт морозива у програмі, то він міг би відображатись через наступні змінні і методи (на рисунку нижче):



**Classes**

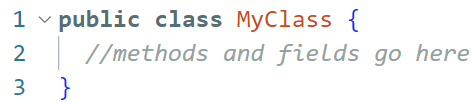
У реальному світі ми можемо спостерігати багато об’єктів, що належать до одного виду/типу, чи будь-якої іншої категорії. Існує величезна кількість порцій морозива, виготовленого одним виробником, за одним і тим самим рецептом, в одній і тій самій упаковці. У світі ООП ми б сказали, що всі ці морозива є об’єктами одного класу.

Отож, **клас** – це свого роду креслення (рецепт, план, схема) за яким, створюються об’єкти. Саме приналежність об’єкта до певного класу, визначає його набір полів і методів. Відтак об’єкт класу *IceCream* матиме зовсім інший набір полів і методів, аніж об’єкт *Car*.

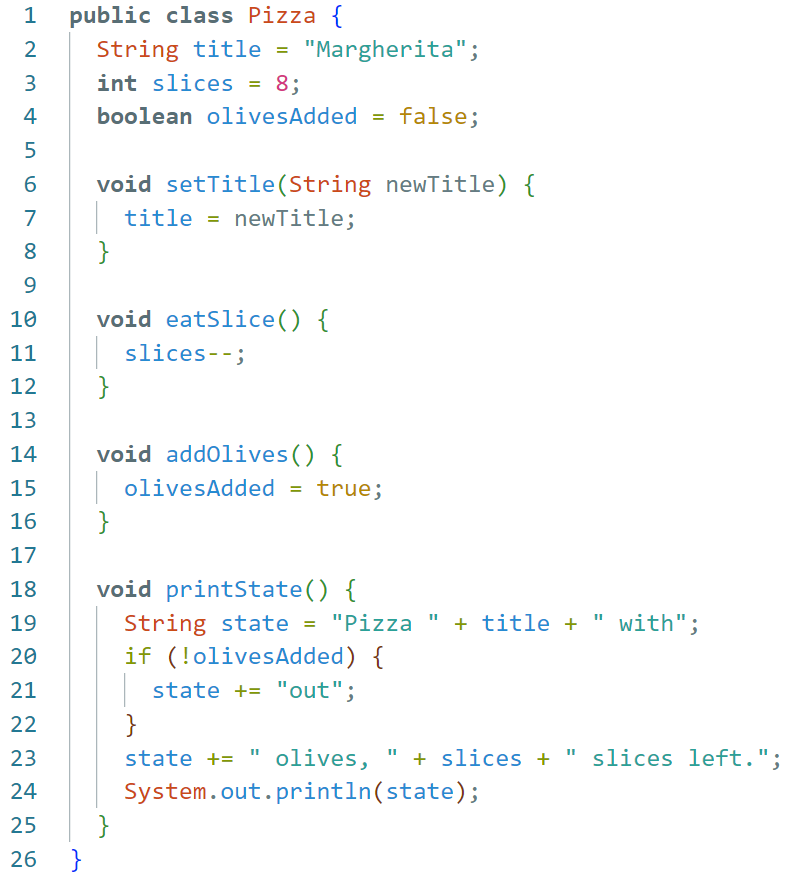


У Java для створення об’єкта ми повинні визначити його клас (або ж тип). Часто ми користуємось уже створеними класами (наприклад String, або Scanner), але особливі можливості для нашої програми відкриваються коли вміємо створювати власні класи. Створивши клас, ми можемо поселити у нашій програмі різноманітні сутності, із яких буде створено необхідний функціонал.

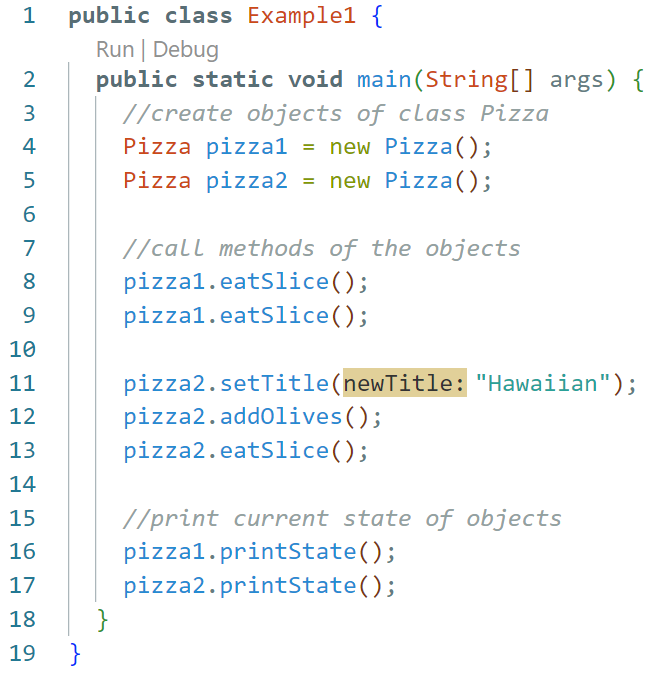
Синтаксис створення класу дуже простий. Необхідно вказати ключове слово **class** і за ним *назву класу*, а перед ним *модифікатор доступу* (наприклад public). У фігурних дужках після назви вказуються поля і методи:



У якості прикладу створимо клас Pizza, який певною мірою відображатиме реальну піцу в нашій програмі.



Важливо розуміти, що даний клас не має коду, який виконується сам по собі (для того, щоб ним скористатись необхідно створити об’єкт класу і використати його в іншому класі цього ж проєкту із методом **main**). Наприклад:



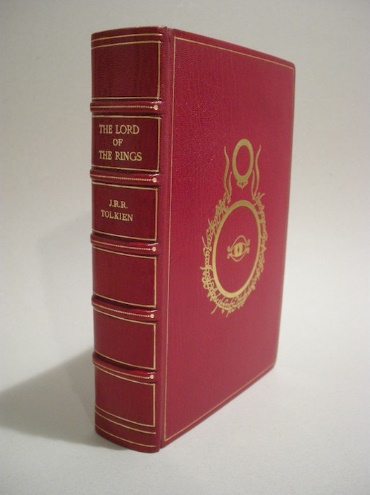
Зверніть увагу на рядки створення об’єктів. Спочатку вказуємо **назвукласу** (або ж тип об’єкта), потім **ім’яоб’єкта**. Після оператора присвоєння значення маємо ключове слово **new** за яким слідує виклик *конструктора класу*. Детальніше ми поговоримо про конструктори наступного разу, а зараз головне запам’ятати, що назва конструктора = назву класу.

Унаслідок виконання коду із методу **main**, у консоль буде виведено такий текст:

*Pizza Margherita without olives, 6 slices left.*

*Pizza Hawaiian with olives, 7 slices left.*

**Tasks**

**Завдання 1**. Створіть клас що буде відображати у програмі сутність книжки. Стан книжки повинен відображатись наступною інформацією:

* назва;
* загальна кількість сторінок;
* поточна сторінка (на якій сторінці книжку відкрито);

Із об’єктом книжки можна виконувати наступні дії:

* задати назву;
* задати кількість сторінок;
* прогорнути певну кількість сторінок уперед;
* прогорнути певну кількість сторінок назад;
* вивести у консоль поточний стан книжки у наступному вигляді:

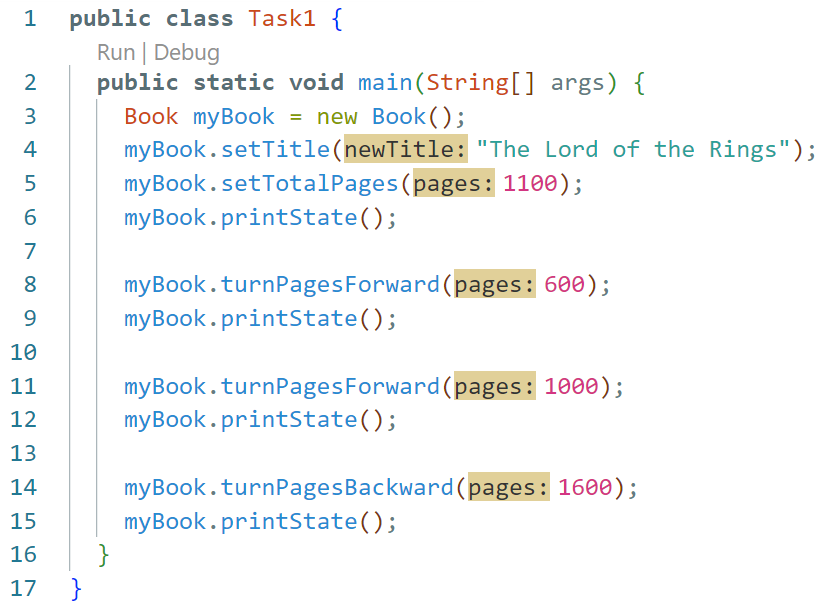
*Title: The Lord of the Rings (page 0 out of 1100)*

Врахуйте, що перелистувати неіснуючі сторінки не можна. Тобто значення поточної сторінки завжди повинно лежати в межах від 0 до загальної кількості сторінок.

У іншому класі (Task1) у методі main створіть один об’єкт (або декілька об’єктів) класу Book і перевірте роботу методів. Наприклад:

**Homework**

Тестові дані:



Очікуваний результат:

*Title: The Lord of the Rings (page 0 out of 1100)*

*Title: The Lord of the Rings (page 600 out of 1100)*

*Title: The Lord of the Rings (page 1100 out of 1100)*

*Title: The Lord of the Rings (page 0 out of 1100)*

**Завдання 1.** Повторіть завдання із класом Book самостійно, але додайте до нього поле, у якому буде зберігатись інформація про автора книги. Крім того, клас повинен мати метод, яким можна буде задати автора.

Враховуючи зазначені зміни, інформація про стан книги повинна відображатись у наступному форматі:

*Title: The Lord of the Rings, Author: J. R. R. Tolkien (page 0 out of 1100)*

