**Lesson 20. Constructors**

Протягом даного заняття ми познайомимось поняттям *конструктора* класу (хоч ми вже і не раз користувались конструкторами, і йдеться зовсім не про Lego), а також тримаємо змогу поекспериментувати із класами та об’єктами. Наприклад, одне з полів зробити об’єктом іншого класу …

**Constructor is …**

Кожного разу коли ми створювали об’єкт класу, ми викликали його конструктор. Вже зі слова «викликали» вже можемо зрозуміти, що **конструктор** – це ніщо інше як метод, однак метод особливий. Особливий він саме тому, що в його задачу входить **створення об’єкта**.

Звідки ж він взявся, запитаєте ви, якщо ми його ні разу не прописували? Справа в тому, що Java автоматично створює для класів конструктор за замовчуванням у тому випадку, якщо ми власноруч його не вказуємо у коді. Конструктор за замовчуванням не має параметрів і просто створює новий об’єкт.

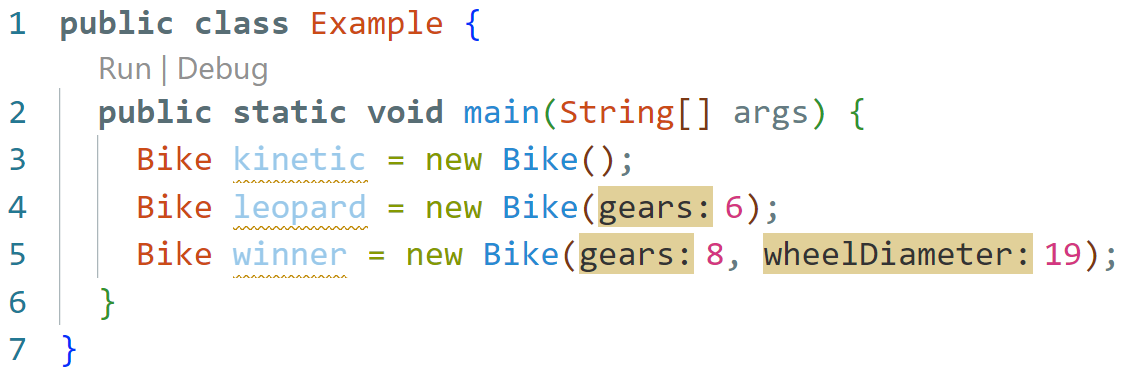
Конструктор за замовчуванням підходить для тих випадків, коли створення об’єкту класу не потребує зазначення додаткової інформації. Якщо додаткові дані таки потрібні, то необхідно створювати власний конструктор із параметрами.

Найчастіше головна задача конструкторів із параметрами – визначення початкових значень полів (усіх, або декількох).

Розглянемо короткий приклад:



Клас Bike має три конструктора, а отже об’єкт Bike можна створити трьома різними способами:



При створення у параметрах можемо задати кількість шестерень, а також діаметр колеса.

При роботі із конструкторами варто пам’ятати наступне:

* Конструктори повинні мати те саме ім’я, що й клас, у якому вони визначені.
* Конструктори не повертають жодного типу.
* Конструктори викликаються лише один раз під час створення об’єкта.

Зверніть увагу, на ключове слово **this** у рядках 9, 14, 15. Воно є вказівкою на поточний об’єкт класу, що виконує той чи інший метод. За допомогою this можемо звертатись в середину об’кта (до його методів чи полів). У даному випадку його було зручно використати, для того щоб не видумувати назви параметрів відмінні від назв полів.

**Tasks**

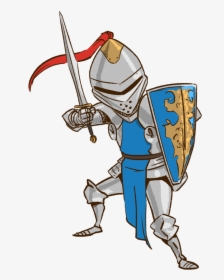
**Завдання 1.** Створіть клас Sword, який буде відображати об’єкт меча. Стан об’єкта меча повинен описуватись відповідно до наступних пунктів:

* базові пошкодження, які можна нанести мечем (повинні задаватись при створенні меча);
* гострота меча лежить в межах від 0 до 5;

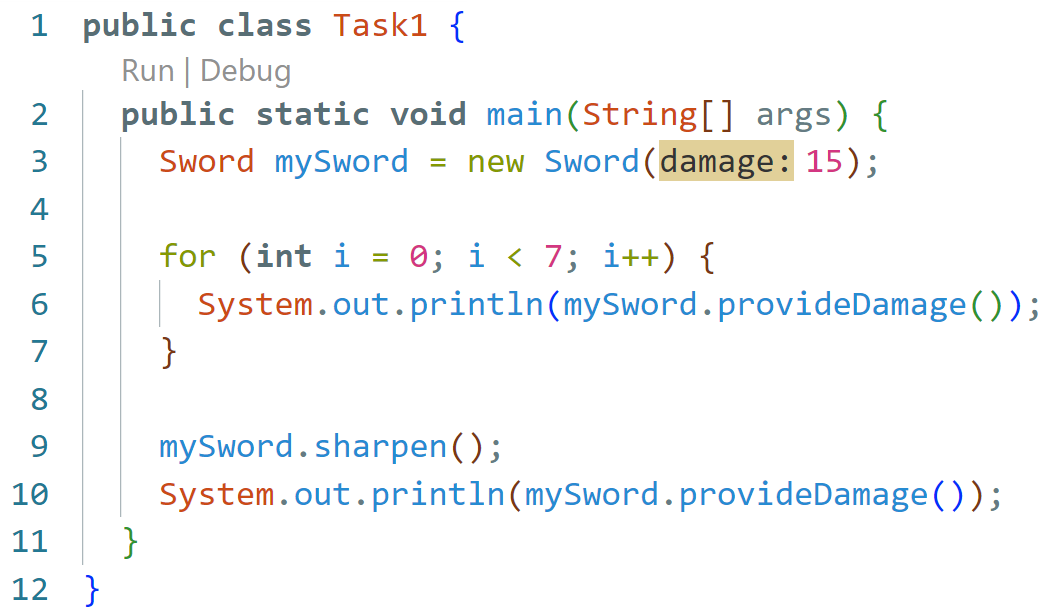


Поведінка об’кта меча повинна мати наступну логіку

* пошкодження, які наносить меч залежать від гостроти (якщо гострота більше 0, то меч може нанести пошкодження, що дорівнюють базовій величині, а інакше – половину від неї);
* кожного разу при нанесенні удару гострота меча зменшується на 1;
* меч можна нагострити (присвоїти гостроті меча значення 5);



Тестові дані:



Очікуваний результат:

*15*

*15*

*15*

*15*

*15*

*7*

*7*

*15*

**Завдання 2.** Створіть клас Knight, який буде відображати об’єкт лицаря. Стан лицаря повинен описуватись відповідно до наступних пунктів:

* максимальна кількість здоров’я (повинна задаватись при створенні об’єкта);
* поточна кількість здоров’я (від 0 до максимального значення);
* меч (об’єкт класу Sword, надається при створенні об’єкта);

Поведінка об’кта лицаря повинна мати наступну логіку:

* лицар може отримувати пошкодження (зменшується поточний стан здоров’я);
* лицар може наносити удар мечем, якщо його здоров’я більше 0;
* лицар може лікуватись (відновлювати частину здоров’я);

Тестові дані:





Очікуваний результат:

*Health: 88 Strike: 10*

*Health: 76 Strike: 10*

*Health: 64 Strike: 10*

*Health: 52 Strike: 10*

*Health: 40 Strike: 10*

*Health: 28 Strike: 5*

*Health: 16 Strike: 5*

*Health: 4 Strike: 5*

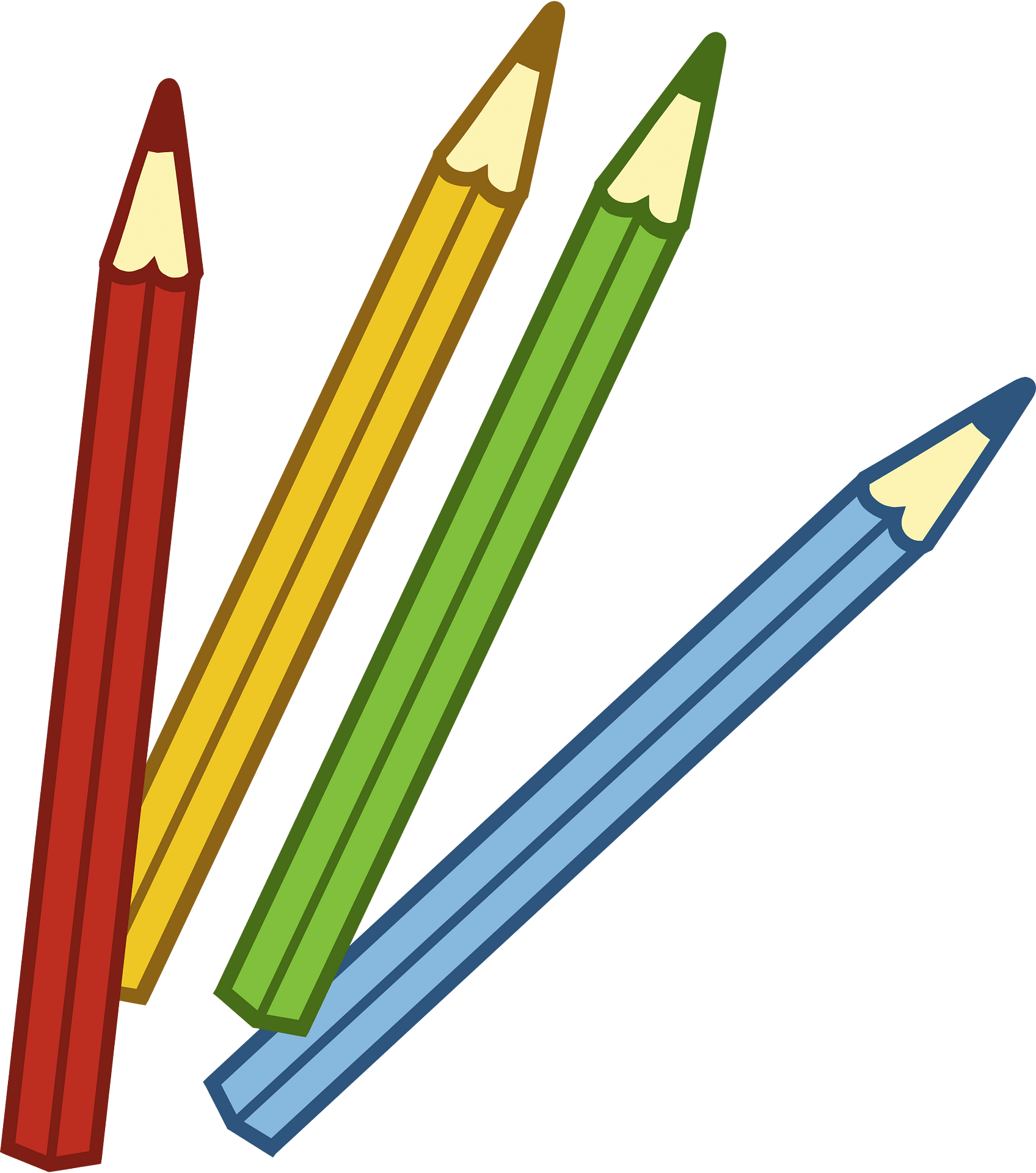
*Health: 0 Strike: 0*

*Health: 0 Strike: 0*

**Завдання 3.** Запрограмуйте двобій двох лицарів. Два об’єкти класу Knight по черзі повинні атакувати один одного, допоки у когось рівень здоров’я не стане менше 1.

**Homework**

**Завдання 1.** Cтворіть клас Pencil, об’єкти якого повинні моделювати роботу звичайного олівця. Стан олівця повинен описуватись наступними полями:



* довжина олівця (в сантиметрах);
* колір;

Для олівця повинна працювати поведінка:

* можна виміряти довжину олівця і отримати інформацію про колір;
* олівець може намалювати малюнок (вивести назву малюнка у консоль разом з інформацією про колір у наступному вигляді: «Pictured: Nice portrait (green)»);
* олівець може намалювати малюнок, якщо довжина олівця більше 0, а кожного разу, коли виконується малюнок, довжина олівця зменшується на 1 см.

Врахуйте, що повинно бути два варіанта створення (два конструктора) олівця: вказати можна тільки колір, або і колір і довжину. Якщо вказується тільки колір, то за замовчуванням довжина – 10 см.

