**РОЗДІЛ 15. КЛАСИ ТА МЕТАКЛАСИ**

Як ви вже знаєте із попередніх розділів, у Pharo все є об’єктами, а кожний об’єкт є екземпляром класу. Самі класи теж не становлять виняток – усі класи є об’єктами, причому класи-об’єкти є екземплярами інших класів. Така об’єктна модель охоплює суть об’єктно-орієнтованого програмування, і є при тому компактною, простою, елегантною та однорідною. Однак наслідки цієї однорідності можуть збити з пантелику новачка.

Зауважимо, що необов’язково повністю розуміти всі ці наслідки, щоб вільно програмувати на Pharo. Тим не менше, мета цього розділу двояка: (1) максимально заглибитись і (2) показати що тут немає ніякої магії чи особливих складнощів: лише однорідно застосовані прості правила. Слідуючи цим правилам, можна завжди розуміти, чому ситуація саме така, а не інакша.

**15.1. Правила для класів та метакласів**

Об’єктна модель Pharo базується на обмеженій кількості підходів, застосованих однорідно. Пригадаємо правила об’єктної моделі, розглянуті в розділі «Об’єктна модель Pharo»:

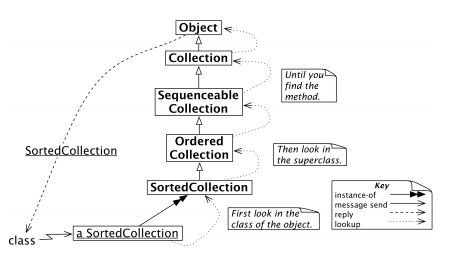
**Правило 1:** Будь-що є об’єктом.

**Правило 2:** Будь-який об’єкт є екземпляром класу.

**Правило 3:** Кожен клас має свій суперклас.

**Правило 4:** Все відбувається через надсилання повідомлень.

**Правило 5:** Пошук методу відбувається згідно ланцюжка наслідування.



**Рис.15.1. Надсилання повідомлення class до відсортованої колекції.**

Як було згадано у вступі до розділу, з Правила 1 випливає що *класи також є об’єктами*, тому Правило 2 каже нам, що класи мають теж бути екземплярами класів. Клас класу називається *метакласом*. Метаклас створюється автоматично кожного разу, коли створюється клас. У переважній більшості випадків не потрібно переживати чи навіть думати про метакласи. Проте кожного разу, коли ви використовуєте браузер для пошуку інформації про клас класу, не буде зайвим пригадати, що насправді ви шукаєте інший клас. Клас і метаклас є окремими класами, попри те, що перший є екземпляром другого.

Для детальнішого розуміння класів і метакласів розширимо правила з розділу «Об’єктна модель Pharo» такими додатковими правилами:

**Правило 6:** Кожен клас є екземпляром свого метакласу.

**Правило 7:** Ієрархія метакласів паралельна ієрархії класів.

**Правило 8:** Кожен клас наслідується від класів Class i Behavior.

**Правило 9:** Кожен метаклас є екземпляром класу Metaclass.

**Правило 10:** Метаклас класу Metaclass є екземпляром класу Metaclass.

Разом, ці 10 правил становлять об’єктну модель Pharo.

Спершу коротко повторимо 5 правил з розділу «Об’єктна модель Pharo» на невеликому прикладі. Після цього детальніше розглянемо нові правила, використовуючи той самий приклад.

**15.2. Повторення об’єктної моделі Pharo.**

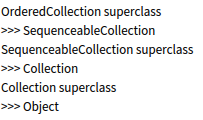
**Правило 1:** Оскільки будь-що є об’єктом, упорядкована колекція у Pharo також є об’єктом.



**Правило 2:** Кожен об’єкт є екземпляром класу. Клас упорядкованої колекції – клас OrderedCollection.



**Правило 3:** Кожен клас має свій суперклас. Суперкласом класу OrderedCollection є SequenceableCollection, а суперкласом SequenceableCollection є Collection.



Розглянемо такий приклад: коли ми надсилаємо повідомлення asSortedCollection, ми перетворюємо упорядковану колекцію у відсортовану колекцію. Покажемо це наглядно:



**Правило 4:** Все відбувається через надсилання повідомлень, звідки випливає, що withAll: - повідомлення до OrderedCollection, asSortedCollection – повідомлення до екземпляру впорядкованої колекції, а superclass – повідомлення до OrderedCollection, SequenceableCollection і Collection. У кожному випадку отримувач є об’єктом, оскільки будь-що є об’єктом, але деякі з цих об’єктів є також класами.

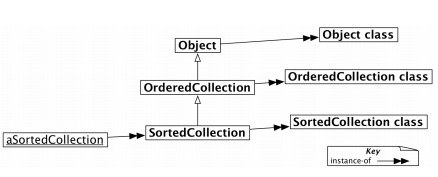
**Правило 5:** Пошук методу відбувається згідно ланцюжка наслідування, тому коли ми надсилаємо повідомлення class до результату (OrderedCollection withAll: #(4 5 6 1 2 3)) asSortedCollection, повідомлення обробляється, коли відповідний метод знайдено в класі Object, як показано на рис.15.1.

**15.3. Кожен клас є екземпляром свого метакласу.**

Як ми вже згадували у розділі 15.1, класи, чиї екземпляри є також класами, називаються метакласами.

Метакласи є неявними

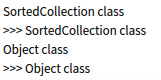
Метакласи автоматично створюються при визначенні класу. Можна сказати, що вони є неявними, оскільки програміст не повинен за них хвилюватись.



**Рис.15.2. Метакласи SortedCollection та їх суперкласи**

Неявний метаклас створюється для кожного створеного класу. Таким чином, кожен метаклас має лише один екземпляр.

На відміну від звичайних класів, метакласи є анонімними. Але завжди можна послатись на них через клас, що є їх екземпляром. Наприклад, класом класу SortedCollection є SortedCollection class, а класом класу Object є Object class.

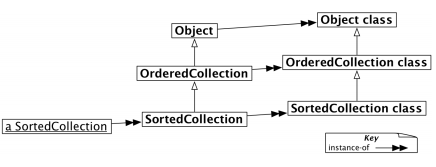


Насправді метакласи не є повністю анонімними, їх ім’я випливає з їх єдиного екземпляру.



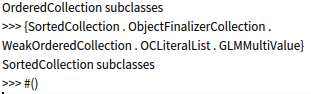
Рис.15.2 показує, як кожен клас є екземпляром свого метакласу. Зазначимо, що ми пропустили класи SequenceableCollection і Collection на рисунку тільки для економії місця. Їх відсутність не змінює загального сенсу.

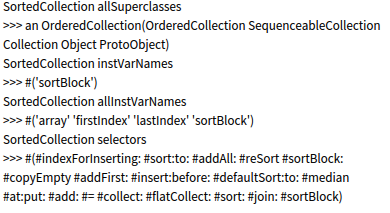
Запити до метакласів



**Рис.15.3. Ієрархія метакласів паралельна ієрархії класів**

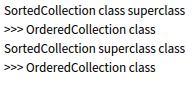
Той факт, що класи також є об’єктами дозволяє легко здійснювати до них запити, надсилаючи повідомлення. Наприклад,



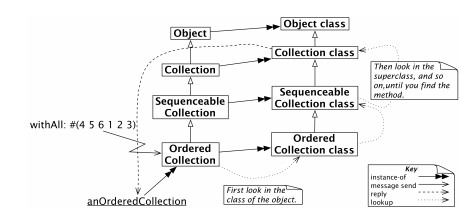


**15.4. Ієрархія метакласів паралельна ієрархії класів**

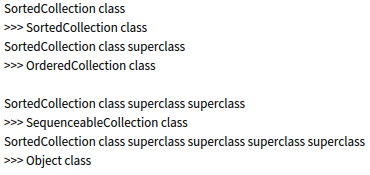
Правило 7 каже, що суперклас метакласу не може бути звичайним класом; він є метакласом суперкласу єдиного екземпляру даного метакласу.



Це і означає, що ієрархія метакласів паралельна ієрархії класів. Рис.15.3 показує, як це працює у випадку SortedCollection.



**Рис.15.4. Пошук методів у класах такий самий, як у звичайних об’єктах**



Однорідність між класами та об’єктами

Повернімось дещо назад і уявімо, що немає різниці між надсиланням повідомлення до об’єкту і до класу. В обох випадках пошук відповідного методу починається із *класу отримувача* і *далі вгору по ланцюжку наслідування*.

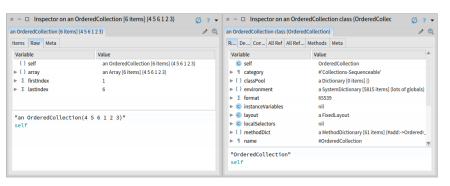
Тому повідомлення, які надсилаються до класів, слідують ланцюжку наслідування метакласів. Розглянемо, наприклад, метод withAll:, реалізований у класі Collection. Коли ми надсилаємо повідомлення withAll: до класу OrderedCollection, пошук відбувається аналогічно, як і з будь-яким іншим повідомленням. Він починається з класу OrderedCollection class (оскільки він починається із отримувача, а отримувачем є OrderedCollection) і рухається вгору по ієрархії метакласів, поки метод не буде знайдено у класі Collection class (див. Рис.15.4). Повертається новий екземпляр класу OrderedCollection.



Єдиний принцип пошуку методів

Таким чином, бачимо, що у Pharo існує єдиний принцип пошуку методів. Класи є всього лише об’єктами і поводяться, як і інші об’єкти. Класи мають можливість створювати нові екземпляри лише завдяки тому, що вони можуть приймати повідомлення new, а метод, який приймає це повідомлення, «вміє» сворювати нові екземпляри.

Зазвичай, об’єкти-некласи не приймають цього повідомлення, однак ніщо не заважає додати метод new у неметаклас, якщо на це є причина.



**Рис. 15.5. Класи також є об’єктами**

Дослідження об’єктів та класів

Оскільки класи є об’єктами, їх також можна досліджувати.

Дослідимо OrderedCollection withAll: #(4 5 6 1 2 3) і OrderedCollection.Зауважимо, що в першому випадку ми досліджуємо екземпляр класу OrderedCollection, а в другому -–сам клас OrderedCollection. Це може бути дещо спантеличуючим, адже рядок заголовка інспектора показує *клас* об’єкта, який досліджується.

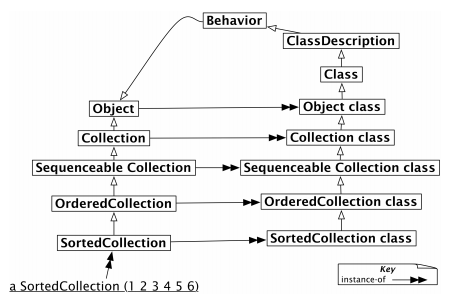
Дослідження OrderedCollection дозволяє побачити суперклас, змінні екземплярів, словник методів і т.д. для класу OrderedCollection, як показано на Рис. 15.5.

**15.5. Кожен метаклас наслідується від класів Class i Behavior.**

Кожен метаклас є класом (з єдиним екземпляром), а тому наслідується від класу Class. Class, у свою чергу, наслідується від своїх суперкласів, ClassDescription i Behavior. Оскільки будь-що у Pharo є об’єктом, всі ці класи у підсумку наслідуються від класу Object. Повну картину можна бачити на Рис.15.6.

Де визначається new?

Щоб зрозуміти важливість факту, що метакласи наслідуються від Class i Behavior, досить подумати, де визначається new, і як воно знаходиться. Коли повідомлення надсилається до класу, воно шукається у ланцюжку метакласів, і, зрештою, у їх суперкласах Class, ClassDescription i Behavior, як показано на Рис.15.7.



**Рис.15.6. Метакласи наслідуються від Class i Behavior**

Питання *«Де визначається new?»* є вирішальним. New вперше визначається у класі Behavior і може бути перевизначене у його підкласах, включаючи будь-які метакласи визначених класів, якщо це необхідно. Тепер, коли повідомлення new надіслане до класу, воно шукається, зазвичай, у метакласах цього класу, продовжуючи по ланцюжку суперкласів до класу Behavior, якщо воно не було перевизначене десь на цьому шляху.

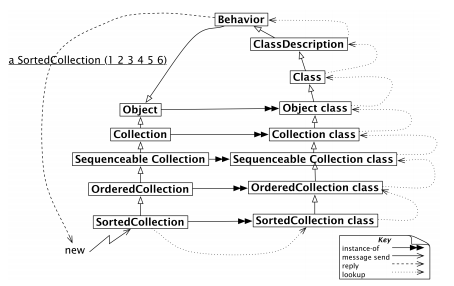
Зазначимо, що результат надсилання SortedCollection new є екземпляром SortedCollection, але *не* Behavior, хоча метод і шукається в класі Behavior, new завжди повертає екземпляр self, класу, який отримав повідомлення, навіть якщо воно реалізоване в іншому класі.



**Поширена помилка.** Поширеною помилкою є шукати new у суперкласі отримувача. Це саме стосується і new: , стандартного повідомлення для створення об’єкта вказаного розміру. Наприклад, Array new: 4 створює масив із 4 елементів. Ви не знайдете цього методу ні в класі Array, ні в жодному із його суперкласів. Замість цього вам потрібно шукати його в класі Array class та його суперкласах, оскільки пошук починається саме з нього (див Рис.15.7).

Призначення класів Behavior, ClassDescription i Class

Behavior надає мінімальний набір можливостей, необхідний для об’єктів, які мають екземпляри; сюди входять: посилання на суперклас, словник методів і формат класу. Формат класу – це ціле число, що позначає відмінності між вказівником та невказівником, компактом та некомпактом, а також базовий розмір екземплярів. Behavior наслідується від класу Object, тому він, як і всі його підкласи, можуть поводитись як об’єкти.



**Рис.15.7. new – звичайне повідомлення, що шукається в ланцюжку метакласів**

Behavior також є базовим інтерфейсом для компілятора. Він містить методи, необхідні для створення словника методів, компілювання методів, створення екземплярів (наприклад, new, basicNew, new: , basicNew: ), управління ієрархією класів (наприклад, superclass: , addSubclass: ), доступу до ієрархії класів (superclass, subclasses) і запитів (hasMethods, includesSelector, canUnderstand:, inheritsFrom:, isVariable).

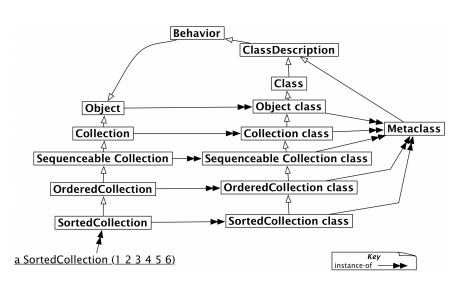
ClassDescription – абстрактний клас, що надає функціональність, необхідну для двох його прямих нащадків, Class i Metaclass. ClassDescription розширює базову функціональність Behavior: іменовані екземпляри-змінні, категоризація методів у протоколи, підтримка змін, їх логування, та більшість необхідних для змін можливостей.

Class представляє загальну поведінку всіх класів. Він містить ім’я класу, компіляцію методів, сховище методів та екземпляри-змінні. Він надає конкретне представлення іменам змінних класів і загальному пулу змінних (addClassVarName:, addSharedPool:, initialize). Оскільки метаклас є класом з єдиним екземпляром (не метакласом), всі метакласи зрештою наслідуються від класу Class (див. Рис. 15.9).

**15.6. Кожен метаклас є екземпляром класу Metaclass.**

Оскільки метакласи теж об’єкти, постає питання: вони теж мусять бути екземплярами якогось класу, але якого? Метакласи – також об’єкти, вони є екземплярами класу Metaclass, як показано на Рис.15.8. Екземпляри класу Metaclass – анонімні метакласи, кожен із яких має точно один екземпляр, який є класом.

Metaclass представляє загальну поведінку метакласів. Він містить методи для створення екземплярів (subclassOf: ), створення ініціалізованих екземплярів єдиного екземпляру метакласу, ініціалізації змінних класу, екземплярів метакласу, компіляції методів та інформації про клас (наслідувальні посилання, змінні екземплярів тощо)



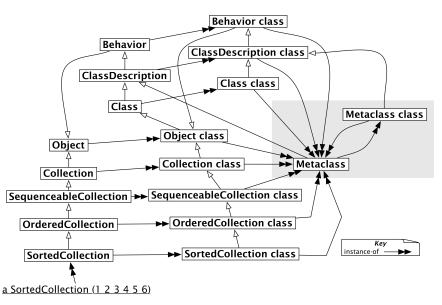
**Рис.15.8. Кожен метакласу – екземпляр класу Metaclass**

**15.7.** **Метаклас класу Metaclass є екземпляром класу Metaclass**

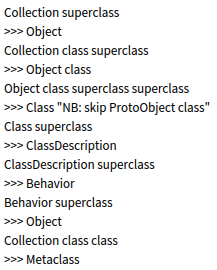
Останнє питання, на яке потрібно дати відповідь: а що є класом класу Metaclass class? Відповідь проста: це метаклас, а отже, він повинен бути екземпляром класу Metaclass, як і всі інші метакласи в системі (див. Рис. 15.9.).

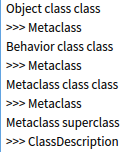
Рис.15.9. показує, що всі метакласи є екземплярами класу Metaclass, включаючи і метаклас самого класу Metaclass. Порівнявши Рис.15.8 і Рис.15.9, можна побачити, що ієрархія метакласів ідеально віддзеркалює ієрархію класів, на всьому шляху, аж до Object class.

Наведений нижче приклад показує, як можна звертатись до ієрархії класів, щоб продемонструвати правильність Рис. 15.9. (Насправді ви побачите, що ми дещо прибрехали – Object class superclass --> ProtoObject class, не Class. У Pharo потрібно піднятись на один рівень вище, щоб досягти класу Class)



**Рис.15.9. Всі метакласи є екземплярами класу Metaclass, навіть екземпляр класу Metaclass.**





**15.8. Підсумки розділу**

Цей розділ надав можливість заглянути в глибину однорідної об’єктної моделі, і більш ретельно пояснив, як організовані класи. Якщо ви «губитесь» чи спантеличені, то завжди потрібно пам’ятати, що ключ – передавання повідомлень: потрібно шукати метод у класі отримувача. Це працює для *будь-якого* отримувача. Якщо метод не знайдено в класі отримувача, пошук продовжується у його суперкласах.

• Кожен клас є екземпляром свого метакласу. Метакласи є неявними. Метаклас створюється автоматично кожного разу, коли створюється клас, який є його єдиним екземпляром. Простіше кажучи, метаклас – це клас, єдиним екземпляром якого є клас.

• Ієрархія метакласів паралельна ієрархії класів. Пошук методу в ієрархії метакласів аналогічний пошуку методу в звичайних класах і слідує ланцюжку наслідування метакласів.

• Кожен метаклас наслідується від Class i Behavior. Кожен клас є Class’ом. Оскільки метакласи – також класи, то і вони наслідуються від Class. Behavior реалізовує поведінку, спільну для всіх сутностей, що мають екземпляри.

• Кожен метаклас є екземпляром класу Metaclass. ClassDescription містить все, що є спільного в класах Class i Metaclass.

• Метаклас класу Metaclass є екземпляром класу Metaclass. Відношення *є екземпляром* створює замкнутий круг, і таким чином, клас класу класу Metaclass – Metaclass.