УДК 075.8

Р.В. ГЕЛЕМБ'ЮК

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ В СКА MAPLE

Abstract: *It is described software library that extends facilities of computer algebra system Maple with including support of object-oriented programming (OOP) in designing mathematical libraries. It is shown necessity of supporting OOP in Maple and it is analysed the range of application. It is demonstrated the process of applying the library.*
Key words: *Maple, object-oriented programming, computer algebra systems.*

Анотація: *Описано програмну бібліотеку, яка розширює можливості системи комп'ютерної алгебри Maple введенням підтримки об'єктно-орієнтованого підходу (ООП) у програмуванні математичних бібліотек. Показано необхідність підтримки ООП в Maple та проаналізовано сферу застосування. Продемонстровано процес використання бібліотеки.*
Ключові слова: *Maple, об'єктно-орієнтоване програмування, системи комп'ютерної алгебри.*

Аннотация: *Описана программная библиотека, которая расширяет возможности системы компьютерной алгебры Maple введением поддержки объектно-ориентированного подхода (ООП) в программировании математических библиотек. Показана необходимость поддержки ООП в Maple и проанализирована область применения. Продемонстрирован процесс применения библиотеки.*
Ключевые слова: *Maple, объектно-ориентированное программирование, системы компьютерной алгебры.*

1. Вступ

У процесі розробки програмної бібліотеки для Maple, яка реалізує можливості технології розвинення функцій за нев'язками [1], виникла необхідність у створенні моделі бібліотеки у вигляді ієрархії класів з використанням об'єктно-орієнтованого програмування (ООП). Такий підхід, на думку автора, значно спростить реалізацію бібліотеки, дозволить розділити функціональність як на логічному, так і на програмному рівні, а також полегшить подальше її удосконалення та доробку, в тому числі і іншими розробниками.

ООП було вибрано як базову технологію, оскільки в контексті поставленої задачі такий підхід має певні переваги.

ООП є парадигма програмування, в якій основними концепціями є поняття об'єктів та класів [2]. ООП в даний час є одним із лідерів в області прикладного програмування. Така позиція не випадкова. Дана технологія володіє рядом переваг перед іншими існуючими парадигмами.

2. Аналіз технології ООП в контексті поставленої задачі

Ключові поняття ООП – клас та об'єкт. Клас – це тип, який описує роботу об'єктів – екземплярів. Клас можна порівняти з кресленням, згідно з яким створюються об'єкти. Звичайно класи проектують таким чином, щоб їх об'єкти відповідали об'єктам предметної області. Об'єкт поряд з поняттям «клас» є важливим поняттям об'єктно-орієнтованого підходу в програмуванні. Під об'єктом розуміємо деяку сутність у віртуальному просторі, яка володіє визначеним станом і поведінкою. Як правило, при розгляді об'єктів виділяється те, що об'єкти належать одному чи декільком класам, які, у свою чергу, визначають поведінку (є моделлю) об'єкта.

Об'єкт як структура даних являє собою іменований програмний компонент, який містить власні локальні дані й здатний виконувати певні дії над ними [3].

ООП має такі переваги:

ISSN 1028-9763. Математичні машини і системи, 2009, № 1

1) Зменшення кількості помилок. За рахунок того, що можна розбити програму на велику кількість класів, у кожного класу можуть бути свої дані, до яких ніхто ніколи ззовні не може дістати доступу.

2) Прискорення налагодження програми. За рахунок того, що більшість даних і методів строго прив'язані до свого класу й інші класи не мають до них доступу, пошук помилки, як правило, обмежується рамками лише одного класу, а не всієї програми.

3) Набагато прискорюється модифікація програми.

Основні концепції:

• система складається з об'єктів;

• об'єкти деяким чином взаємодіють між собою;

• кожен об'єкт характеризується своїм станом і поведінкою;

• стан об'єкта задається значенням полів даних;

• поведінка об'єкта задається методами.

СКА Maple має досить потужні можливості для програмування власних процедур. Проте вбудовані можливості для програмування мають, на нашу думку, такі недоліки:

• відсутність повнофункціональної підтримки принципів об’єктно-орієнтованого програмування:

• нерозвинутий редактор коду, що значно впливає на продуктивність при великій кількості коду.

З огляду на це, пропонується реалізувати логіки бібліотеки на мові програмування Java. Тим самим отримати можливість у повній мірі використовувати ООП, а також можливості широкого вибору середовищ розробки програм з використанням усіх досягнень у сфері оптимізації роботи програміста. При створенні проекту ми будемо використовувати NetBeans IDE (http://www.netbeans.org/). Таким чином, для включення технології ООП в Maple пропонується платформа двонаправленого зв’язка: Maple-Java.

Для прикладу розглянемо бібліотеку із 3-х класів, яку необхідно реалізувати в Maple (UML-схема).

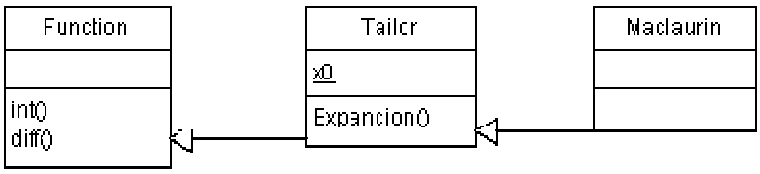

Рис. 1. UML-схема бібліотеки із 3-х класів

3. Особливості реалізації ООП в Maple

Вже декілька років СКА Maple є найбільш потужним продуктом для програмування математичних задач [4, 5]. Починаючи з версії Maple 9.5, в пакет включені нові можливості, які вносять нові потужності у процес програмування та роблять його прозорим і ефективним. Це об'єктно-орієнтоване програмування і поліморфізм. Розробники програмного продукту Maple досить слабо анонсували ці нові можливості. Документація по цих питаннях практично відсутня.

Вивчивши можливості Maple 10 щодо об'єктно-орієнтованого програмування, був зроблений висновок, що оголошена підтримка цієї парадигми є передчасною.

Розглянемо можливості об'єктно-орієнтованого програмування в Maple 10, описані в роботі [6].

Ще до версії 9.5 була можливість створювати модулі, які вміщували функції та процедури користувача, а також змінні.

Робота з модулем виконувалася таким чином [4]:

• завантаження модуля оператором with;

• виклик функцій та процедур модуля;

• функції та процедури могли змінювати статичні змінні модуля.

Починаючи з версії 9.5, модуль почав розглядатися як клас (у розумінні об'єктно-орієнтованого підходу). Тобто внутрішні змінні модуля почали виконувати роль властивостей, а функції та процедури – роль методів класу.

Для того, щоб модуль можна було використовувати як клас, він повинен містити процедуру ModuleApply. Ця процедура виконує роль конструктора класу.

Проте назвати цю нову функціональність підтримкою ООП можна лише умовно, оскільки не реалізовано процедури створення об'єкта певного класу. Тобто всі можливості залишаються такими ж, як і в попередніх версіях, за виключенням того, що одна із процедур модуля виконує функцію конструктора.

Продемонструємо приклад опису класу та його застосування, який подано в роботі [6].

```
> restart;
> Stack := module()
option package;
export ModuleApply, Push, Pop, Top;

  ModuleApply := proc() # Stack object constructor invoked with Stack(args)
                  [args];
                end:
  Push := proc(s) [op(s),args[2..-1]] end:
  Pop := proc(s) if nops(s) > 1 then s[1..-2]; else [] end end:
  Top := proc(s) if nops(s) > 1 then s[-1] end end:
end module:
```

Опис класу (модуля)

```
> with( Stack );
```

$$[ModuleApply, Pop, Push, Top]$$

```
> a := Stack( L, M, N, O );
```

$$a := [L, M, N, O].$$

Створення об’єкта класу. Насправді, це виклик функції ModuleApply. В попередніх версіях Maple такий же ефект можна було б досягнути, виконавши функцію a:=Stack[ModuleApply](L,M,N,O) і без «підтримки» ООП.

Далі наводяться приклади застосування методів класу до об’єктів.

```
> Top( a );
s := Pop( a );
s := Pop( a );
s := Push( a, P );
```

$$\mathrm{O}$$

$$s := [L, M, N]$$

$$s := [L, M, N]$$

$$s := [L, M, N, \mathrm{O}, P]$$

```
> b := Stack();
b := Push( b, A, B );
```

$$b := [\,]$$

$$b := [A, B]$$

```
> c:=Stack(1,2,3,4);
```

$$c := [1, 2, 3, 4]$$

```
> Top(a);Top(b);Top(c);
```

$$\mathrm{O}$$

$$B$$

$$4$$

Як ми бачимо із наведених програм, про ООП тут не йдеться. Те, що розробниками Maple названо конструктором, насправді є звичайною процедурою, яка повертає значення, яке, у свою чергу, може бути використане іншими методами модуля чи взагалі будь-якою іншою функцією (з урахуванням типу даних). За принципами ООП, результатом роботи конструктора має бути посилання на створений об’єкт. У випадку Maple в результаті ініціалізації (створення) не створюється об’єкт класу. Фактично результатом є структура одного із підтримуваних типів даних.

Властивості класу, як і раніше, є статичними для методів класу.

Тобто можна стверджувати, що парадигми ООП не підтримуються пакетом Maple (на даний момент останньою є 12 версія). А декларована розробниками підтримка ООП є планами на майбутнє чи, можливо, рекламним ходом.

Слід зауважити, що інша нова функціональність декларована, починаючи з цієї ж версії, – поліморфізм, підтримується у відповідності із стандартами, описаними в роботі [2].

Для реалізації автором математичної бібліотеки із використанням потужностей ООП було використано можливості Maple для інтеграції із іншими програмами.

4. Інтеграція Maple із зовнішніми програмами

Розглянемо підтримку роботи внутрішніх програм із зовнішніми. Як відомо [4], Maple може експортувати свою функціональність. Тобто існує можливість виклику функцій Maple із зовнішніх програм. А також реалізована функція імпорту функціональності. Тобто Maple може виконувати функції, реалізовані в зовнішніх бібліотеках.

В обох випадках зовнішні програми (чи бібліотеки) можуть бути реалізовані із використанням цілого ряду технологій та мов програмування. В даному випадку було вибрано мову програмування Java, оскільки дана технологія добре інтегрується із Maple, а також є однією із найкращих реалізацій парадигм ООП.

4.1. Експорт функціональності

Технологія OpenMaple – це набір функцій, що надають доступ до алгоритмів та структур даних Maple, скомпільованих на C, Java, чи Visual Basic-програмах.

Java OpenMaple являє собою інтерфейс між обчислювальним ядром Maple та Java-програмою. Цей інтерфейс реалізований з використанням Java-класів та Java-інтерфейсів. Класи дозволяють Java-програмам викликати ядро Maple. Інтерфейси використовуються для того, щоб ядро Maple викликало Java-класи або методи класів. Тобто призначення інтерфейсів – повернення результату до Java-програми для обробки та правильного виводу.

Слід зауважити, що інтерфейси використовуються тільки для отримання результатів після запиту класом функції ядра Maple і не можуть бути використані для оберненої задачі, тобто імпорту функцій в ядро Maple.

4.2. Імпорт функціональності

Команда **define_external** зв'язує функцію, описану в зовнішній бібліотеці, з ядром Maple та організовує інтерфейс між процедурами Maple і вказаною функцією. Функція може бути декларована в бібліотеці DLL у Windows, розподіленій бібліотеці в UNIX чи Java-класі.

5. Компоненти бібліотеки «ООП в Maple»

Опишемо реалізовану бібліотеку, яка дає можливість у повній мірі використовувати ООП для розробників математичних додатків (рис. 2).

Дана схема коротко зображує процес отримання бібліотеки та процесу її використання.

Бібліотека для розробників складається із 2 частин:

1. Java-клас, в якому реалізовано інтерфейси комунікації з ядром Maple. Цей клас має бути батьківським класом для математичних пакетів, які будуть розроблятись з використанням ООП на Java. Тобто, використовуючи наслідування класів, програміст зосереджується на логіці своєї

програми і не витрачає час на технічні нюанси обміну даними та командами між програмою й обчислювальним ядром.

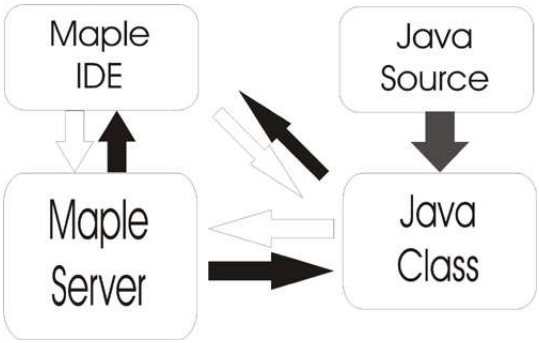

Рис. 2. Процес отримання і використання бібліотеки

2. Maple-модуль. У цьому модулі описано процедуру підключення та ініціалізації Java-програми для подальшого використання в середовищі Maple.

Задачі даної процедури:

• завантажити клас у пам'ять;

• створити посилання на кожен метод класу в середовищі Maple;

• описати деякі типи даних, потрібні для збереження контексту об'єктів;

• перевантажити деякі оператори для коректної роботи з описаними типами даних (використовуючи можливості поліморфізму).

Процес розробки математичної бібліотеки проходить таким чином:

1. Створити Java-проект, використовуючи зручне середовище розробки (наприклад NetBeans).

2. Включити в проект бібліотеку MapleOOP.class.

3. Побудувати ООП-модель на базі класу MapleOOP. Тобто найвищі в ієрархії моделі класи повинні наслідувати клас MapleOOP.

4. Реалізувати функціональність та скомпілювати проект.

5. Включити модуль MapleOOP в середовищі Maple. Викликати процедуру initooplib, аргумент якої – шлях до класу (файлу) щойно скомпільованого проекту.

UML-діаграма наведеного на початку статті прикладу зміниться на рис. 3.

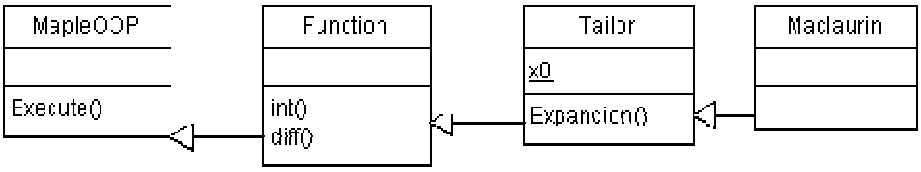

Рис. 3. UML-діаграма бібліотеки

6. Приклад роботи із класом, описаним на Java, з використанням розробленої технології

```
> restart;
> with(mapleoop):
> initooplib("c:/path/to/class/mylib.class");
```

```
> a:=Maclaurin(exp(x));
> a->int();
        exp(x)
> a->diff();
        exp(x)
> a->Expansion();
        1+1*x+1/2*x^2+1/6*x^3+1/24*x^4+1/120*x^5+O(x^6)
```

7. Технічні особливості реалізації бібліотеки MapleOOP

Серед найбільших проблем при реалізації даної бібліотеки можна назвати збереження контексту об'єкта. Це пов'язано із особливостями виклику (імпорту) зовнішніх процедур. Проблема полягає в тому, що виклик якогось із методів класу (в тому числі і конструктора) проходить незалежно від стану документа чи ядра Maple. Процес підключення класу та виклику модуля проходить таким чином:

- підключення та завантаження класу;
- створення об'єкта класу;
- виклик метода із відповідними аргументами;
- повернення результату;
- знищення об'єкта.

Як бачимо, об'єкт створюється та знищується кожного разу при виклику метода. Таким чином, всі властивості об'єкта, змінені конструктором чи якимсь із методів, втрачаються [7, 8].

З огляду на це, було введено спеціальний тип – структуру для збереження контексту об'єкта від виклику до виклику. Застосування цього підходу виглядає таким чином:

- конструктор повертає структуру, в якій збережено інформацію про об'єкт (клас, значення властивостей);
- при виклику метода структура із інформацією про об'єкт знову передається до конструктора, де проходить встановлення властивостей об'єкта;
- викликається метод;
- стан об'єкта зберігається в тій же структурі.

8. Висновки

Можливості актуальних на даний час версій СКА Maple не дозволяють застосовувати парадигми ООП для програмування додатків. З огляду на те, що можливості для програмування в цій СКА залишаються найбільшими з-поміж інших пакетів цього ж класу, є актуальним включення переваг ООП в Maple. Описаний у статті підхід дозволяє використати всі потужності об'єктно-орієнтованого підходу в Maple, привносить додаткові можливості та дозволяє оптимізувати роботу програміста.

СПИСОК ЛИТЕРАТУРЫ

1. Теслер Г.С. Новая кибернетика. – Киев: Логос, 2004. – 404 с.
2. Грэхем Иан Объектно-ориентированные методы. Принципы и практика. – М.: Вильямс, 2004. – С. 880.
3. Казимир В.В. Об'єктно-орієнтоване програмування: Навчальний посібник. – Київ: Дім «Слово», 2008. – 192 с.

4. Аладьев В.З. и др. Программирование и разработка приложений в Maple / В.З. Аладьев, В.К. Бойко, Е.А. Ровба. – Гродно, Таллинн, 2007. – 456 с.
5. Аладьев В.З. Основы программирования в Maple. – Таллинн, 2006. – 300 с.
6. http://www.maplesoft.com/applications/app_center_view.aspx?AID=1485.
7. Роганов Е.А. Основы информатики и программирования: Учебное пособие. – М.: МГИУ, 2001. – 315 с.
8. Кьоу Джим, Джеанини Марио Объектно-ориентированное программирование. Просто и понятно. – Санкт-Петербург: Питер, 2005. – 240 с.

Стаття надійшла до редакції 21.10.2008