3. СТРУКТУРА ПРОГРАММ

Цель – изучить принципы структурной декомпозиции приложений.

Задачи*:*

– научиться создавать и импортировать пакеты;

– изучить принцип структуризации стандартных библиотек Java и .NET Framework;

– научиться формировать структуру программных модулей при разработке сложного приложения.

Формируемые компетенции: ПК-11; ПК-12; ПК-30.

Теоретическая часть

Основы структурной организации исходного кода. В предыдущих лабораторных работах создавались классы и демонстрировались методы использования возможностей стандартной библиотеки Java SE. Для эффективной разработки и поддержки ООП-приложений необходимо принимать дополнительные меры по структурированию кода. Необходимо понимать, что классы следует определять в отдельных файлах (для java – это файлы с расширением *.java; при использовании C# – *.cs). Распределение классов по отдельным файлам не единственная возможность для программиста для оптимизации исходных кодов приложения.

В стандартную библиотеку Java API входят сотни классов. Каждый программист в ходе работы добавляет к ним десятки своих классов. Множество классов растет и становится неуправляемым. Уже давно принято отдельные классы, решающие какую-то одну определенную задачу, объединять в библиотеки классов (объединение по функциональному признаку). Но библиотеки классов, кроме стандартной библиотеки, не являются частью языка.

Разработчики Java включили в язык дополнительную конструкцию – пакеты (packages). Все классы Java распределяются по паке-

там. Кроме классов, пакеты могут содержать интерфейсы и вложенные подпакеты (subpackages). Образуется древовидная структура пакетов и подпакетов.

Эта структура в точности отображается на структуру файловой системы. Все файлы с расширением class (содержащие байт-коды), образующие один пакет, хранятся в одном каталоге файловой системы. Подпакеты образуют подкаталоги этого каталога.

Каждый пакет создает одно пространство имен (namespace). Это означает, что все имена классов, интерфейсов и подпакетов в пакете должны быть уникальны. Имена в разных пакетах могут совпадать, но это будут разные программные единицы. Таким образом, ни один класс, интерфейс или подпакет не может оказаться сразу в двух пакетах. Если надо в одном месте программы использовать два класса с одинаковыми именами из разных пакетов, то имя класса уточняется именем пакета: <имя_пакета>.<Имя_класса>. Такое уточненное имя называется полным именем класса (fully qualified name).

Все эти правила, опять-таки, совпадают с правилами хранения файлов и подкаталогов в каталогах, только в файловых системах для разделения имен каталогов в пути к файлу обычно используется наклонная черта или двоеточие, а не точка.

Еще одно отличие от файловой системы: подпакет не является частью пакета, и классы, находящиеся в нем, не относятся к пакету, а только к подпакету. Поэтому для того, чтобы создать подпакет, не надо предварительно создавать пакет. С другой стороны, включение в программу пакета не означает включение его подпакетов. Пакетами пользуются еще и для того, чтобы добавить к уже имеющимся правам доступа к членам класса private, protected. и public (уже рассматривались в предыдущих лабораторных). В Java существует еще один, «пакетный» уровень доступа.

Если член класса не отмечен ни одним из модификаторов private, protected, public, то по умолчанию к нему осуществляется пакетный доступ (default access), т. е. к такому члену может обратиться любой метод любого класса из того же пакета. Пакеты ограничивают и доступ к классу целиком – если класс не помечен модификатором public, то все его члены, даже открытые, public, не будут видны из других пакетов.

Следует обратить внимание на то, что члены с пакетным доступом не видны в подпакетах данного пакета.

Пакеты и подпакеты. Чтобы создать пакет, необходимо в первой строке java-файла с исходным кодом записать строку

package имя_пакета;

например:

pc kag lesson2;

Тем самым создается пакет с указанным именем lesson2 и все классы, записанные в этом файле, попадут в пакет lesson2. Повторяя эту строку в начале каждого исходного файла, включаем в пакет новые классы.

Имя подпакета уточняется именем пакета. Чтобы создать подпакет с именем, например, subpack, следует в первой строке исходного файла написать:

package имя_пакета. имя_подпакета;

и все классы этого файла и всех файлов с такой же первой строкой попадут в данный подпакет. Можно создать и подпакет подпакета произвольной вложенности.

Поскольку строка «package имя_пакета;» только одна и это обязательно первая строка файла, каждый класс попадает только в один пакет или подпакет.

Соглашение «Code Conventions» рекомендует записывать имена пакетов строчными буквами. Тогда они не будут совпадать с именами классов, которые, по соглашению, начинаются с прописной буквы. Кроме того, соглашение советует использовать в качестве имени пакета или подпакета доменное имя своего сайта, записанное в обратном порядке, например:

com.sun.developer

Это обеспечит уникальность имени пакета во всем Интернете.

Вы можете не указывать пакет при использовании интегрированной среды разработки. Компилятор всегда создает для таких классов безымянный пакет (unnamed package), которому соответствует текущий каталог (current working directory) файловой системы.

Безымянный пакет служит обычно хранилищем небольших пробных или промежуточных классов. Большие проекты лучше хранить в

пакетах. Более того, некоторые программные продукты Java вообще не работают с безымянным пакетом. Поэтому в технологии Java рекомендуется все классы помещать в пакеты.

Библиотека классов Java SE 7 API хранится в пакетах java, javax, org. Пакет java содержит только подпакеты applet, awt, beans, dyn, io, lang, math, net, nio, rmi,security, sql, text, util и ни одного класса. Эти пакеты имеют свои подпакеты, например пакет создания графического пользовательского интерфейса и графики java.awt содержит классы, интерфейсы и подпакеты color, datatransfer, dnd, event, font, geom, im, image, print. Необходимо понимать, что количество и состав пакетов Java SE API меняется с каждой новой версией.

Права доступа к членам класса. В предыдущих лабораторных работах при написании кода спецификаторы доступа указывались как перед определением классов, так и перед полями и методами, принадлежащими классам. Для усвоения принципов распределения классов по пакетам рассмотрим пример:

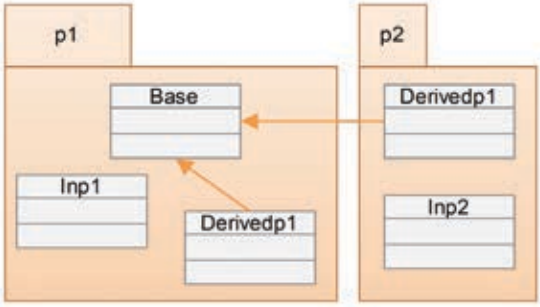

Рис. 3.1. Распределение классов по пакетам

В файле Base.java описаны три класса: Inp1, Base и класс Derivedp1, расширяющий класс Base. Эти классы размещены в пакете p1.

В файле Inp2.java описаны два класса: Inp2 и класс Derivedp2, расширяющий класс Base. Эти классы находятся в другом пакете p2. Класс Base должен быть помечен при своем описании в пакете p1 модификатором public, иначе из пакета p2 не будет видно ни одного его члена.

В пакете p2, доступ ограничен в большей степени. Из независимого класса можно обратиться только к открытым, public, полям класса другого пакета. Из подкласса можно обратиться еще и к защищенным, protected, полям, но только унаследованным непосредственно, а не через экземпляр суперкласса.

Все указанное относится не только к полям, но и к методам.

В табл. 3.1 суммируется информация о доступности полей и методов класса в зависимости от контекста и модификатора доступности.

Таблица 3.1

Особенности доступа к полям класса

	Класс	Пакет	Пакет и подклассы	Все классы
private	+			
«package»	+	+		
protected	+	+	*	
public	+	+	+	+

Импорт пакетов и классов. В коде на языке Java присутствует оператор import, имеющий отношение к структурной организации исходного кода.

Компилятор «видит» типы, определенные только в текущем пакете и стандартном пакете java.lang. С целью использования всех возможностей библиотеки, а также для сокращения имен типов (чтобы не писать полное имя типа) необходимо указывать компилятору какие типы и пакеты будут использоваться в программе. Для этого используется оператор import. Указывается ключевое слово import и через пробел полное имя класса, завершенное точкой с запятой. Сколько классов надо указать, столько операторов import и пишется. Можно использовать конструкцию:

```
import имя_пакета.*;
```

Существует разновидность оператора импорта для поиска статических полей и методов. Например:

```
import static java.lang.Math.*;
```

После подобного импорта можно использовать все статические метода класса Math без указания имени класса. Очевидно, что опе-

ратор import по своему значению бдизок к конструкции using языка C#, но не соответствует include в языке C++, так как не подключает никакие файлы.

Можно сформулировать следующие правила построения исходных файлов Java:

– в первой строке файла может быть необязательный оператор package;

– в следующих строках могут быть необязательные операторы import;

– далее идут описания классов и интерфейсов;

– среди классов файла может быть только один открытый public-класс.

– имя файла должно совпадать с именем открытого класса, если такой существует.

Методика и порядок выполнения работы

1. Спроектируйте структуру приложения для решения задачи в соответствии с вариантом: продумайте структуру файлов, пакетов; необходимые классы.
2. Реализуйте проект приложения в соответствии с разработанной структурой.
3. Проанализируйте структурную организацию программ на языке C#. В чем сходство и в чем различия с технологией Java?

Индивидуальное задание

Объявите требуемые переменные, значения переменным пользователь должен присваивать в процессе выполнения программы (считываются с консоли), выведите на экран с использованием форматной строки значения переменных и результат вычисления выражения во всех возможных форматах:

Вариант	Задание
1	Приложение осуществляет сложение обыкновенных дробей, то есть чисел, $\frac{1}{3}, \frac{7}{12}$ и т.д. Требуется реализовать только сложение. Исходные слагаемые пользователь вводит с клавиатуры
2	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести среднее арифметическое для каждой строки и результирующую матрицу, в которой все элементы заменены по следующим правилам: - если элемент меньше чем среднее арифметическое строки, то выводится 0; - если элемент больше чем среднее арифметическое строки, то выводится 1; - если не выполняются предыдущие правила, то элемент не заменяется
3	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести минимальный и максимальный элементы в матрице (и их индексы) и результирующую матрицу, которая является транспонированной по отношению к исходной
4	Реализуйте программу для выполнения операций над комплексными числами: сложение, умножение, вычисление модуля комплексного числа. Исходные комплексные числа пользователь вводит с клавиатуры
5	Приложение осуществляет умножение обыкновенных дробей, то есть чисел $\frac{1}{3}, \frac{7}{12}$ и т.д. Требуется реализовать только умножение. Исходные множители пользователь вводит с клавиатуры
6	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести максимальный элемент матрицы и результирующую матрицу, в которой все элементы, большие среднего арифметического, заменены на 0

7	Реализуйте приложение, осуществляющее сокращение обыкновенной дроби. Дробь пользователь вводит с клавиатуры
8	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести минимальный элемент матрицы и результирующую матрицу, в которой все элементы, большие среднего арифметического в данной строке, заменены на минимальный элемент матрицы
9	Приложение осуществляет деление обыкновенных дробей, то есть чисел $\frac{1}{3}, \frac{7}{12}$ и т.д. Требуется реализовать только деление. Исходные дроби пользователь вводит с клавиатуры
10	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести среднее арифметическое элементов матрицы и результирующую матрицу, в которой все элементы большие среднего арифметического матрицы, заменены на –1
11	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести средние арифметические для каждой строки и результирующую матрицу, в которой все четные элементы заменены на среднее арифметическое в данной строке
12	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести средние арифметические для каждого столбца и результирующую матрицу, в которой все элементы, заменяются по правилу: четные элементы заменяются на 1, нечетные – на 0
13	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести минимальный элемент для каждой строки и результирующую матрицу, в которой все элементы, которые делятся на 3, заменены на минимальный элемент во всей матрице

14	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести максимальный элемент для каждого столбца и результирующую матрицу, которая является транспонированной по отношению к исходной
15	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести среднее арифметическое для каждой строки и результирующую матрицу, в которой все элементы заменены по следующим правилам: - если элемент делится на 2, то вывести 2; - если элемент делится на 3, то вывести 3; - если не выполняются предыдущие правила, то элемент не заменяется
16	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести среднее арифметическое для каждой строки и результирующую матрицу, в которой все элементы заменены по следующим правилам: - если элемент меньше чем среднее арифметическое строки, то выводится 0; - если элемент больше чем среднее арифметическое строки, то выводится 1; - если не выполняются предыдущие правила, то элемент не заменяется
17	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести минимальный и максимальный элементы в матрице (и их индексы) и результирующую матрицу, которая является транспонированной по отношению к исходной
18	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести максимальный элемент матрицы и результирующую матрицу, в которой все элементы, большие максимального, заменены на 0

19	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести минимальный элемент матрицы и результирующую матрицу, в которой все элементы, большие среднего арифметического в данной строке, заменены на минимальный элемент матрицы
20	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести среднее арифметическое элементов матрицы и результирующую матрицу, в которой все элементы, меньшие среднего арифметического, заменены на 0, а большие среднего арифметического – на 1
21	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести средние арифметические для каждой строки и результирующую матрицу, в которой все четные элементы заменены на среднее арифметическое в данной строке
22	Дана исходная матрица размером $M \times N$. Вывести исходную матрицу. Вывести среднее арифметическое для каждого столбца и результирующую матрицу, в которой все элементы заменены по следующим правилам: - если элемент делится на 2, то вывести 2; - если элемент делится на 5, то вывести 5; - если не выполняются предыдущие правила, то вывести 0

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать следующие данные.

1. Номер и название лабораторной работы; задачи.

3. Ответы на контрольные вопросы.

4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы, подписанный студентом, сдается преподавателю.

Контрольные вопросы

1. Что такое пакет?
2. Какой оператор необходимо использовать для импортирования класса?
3. Какие правила рекомендуется выполнять при именовании классов и пакетов?
4. Назовите правила, которые необходимо выполнять при определении классов и распределении их по файлам и пакетам?
5. Укажите различные варианты оператора import.
6. Укажите сходство механизмов структурной организации программ в технологиях Java и .NET Framework.
7. Укажите отличия механизмов структурной организации программ в технологиях Java и .NET Framework.

Литература: 4, 7, 8.