ЛАБОРАТОРНАЯ РАБОТА 8. ОБОБЩЕННЫЕ ТИПЫ

7.1. Цель и содержание

Цель лабораторной работы: изучить принципы работы с обобщенными типами Java.

Задачи лабораторной работы:

– изучить теоретические принципы реализации обобщений;

– научиться определять обобщенные классы;

– научиться реализовывать приложения с использованием обобщенных типов;

– научиться применять ограничения при реализации обобщенных типов.

7.1 Механизм обобщенного программирования.

Обобщения (generics) представляют собой наиболее существенное изменение в языке программирования Java со времен версии 1.0. Появление обобщений в Java 5.0 стало результатом первых требований к спецификации Java (Java Specification Requests – JSR 14), которые были сформулированы в

1999 г. Группа экспертов потратила около пяти лет на разработку спецификаций и тестовые реализации. Обобщения понадобились потому, что они позволяют писать более безопасный код, который легче читается, чем код, перегруженный переменными Object и приведениями типов. Механизм обобщения максимально используется классами коллекций.

Обобщенное программирование означает написание кода, который может быть многократно использован с объектами многих различных типов. Например, неэффективно программировать отдельные классы для коллекционирования объектов String и Int. И делать это нет необходимости – единственный ArrayList или Vector может собирать объекты любого класса. Это лишь один пример обобщенного программирования, который использовался в предыдущих лабораторных работах.

Реализовать обобщение можно средствами наследования реализации, но подобная стратегия способствует «разбуханию» кода и требут множественных конвертаций и приведений типов (подумайте почему).

7.2 Реализация механизма обобщения.

Такие обобщенные классы, как ArrayList, использовать легко. И большинство программистов Java просто применяют типы вроде ArrayList<String> – как если бы они были частью языка, подобно массивам String[].

Но реализовать обобщенный класс не так просто. Чаще всего данные классы будут использоваться как библиотечные. Программисты, которые будут использовать обобщенный код, попытаются подставлять всевозможные классы вместо ваших параметров типа. Они ожидают, что все будет работать без досадных ограничений и запутанных сообщений об ошибках. задача разработчика как обобщенного программиста – предвидеть все возможные будущие применения обобщенного класса.

Прикладным программистам, вероятно, не придется писать много обобщенного кода. Разработчики библиотеки Java уже выполнили самую

тяжелую работу и предусмотрели параметры типа для всех классов коллекций. В качестве эмпирического правила можно сказать так: от применения параметров типа выигрывает только тот код, в котором традиционно присутствует много приведений от очень общих типов (таких как Object или интерфейс Comparable).

Рассмотрение обобщений начнем с примера:

1. Требуется реализовать класс, который инкапсулирует поля и методы и описывает сущность «Пара элементов различного типа». Назовем класс MyPair.

2. Реализовать класс, описывающий сущность «Нетривиальный список», который отличается от коллекций Java ограниченным набором операций. Данный класс назовем MyCollection.

На рис. 6.1 представлен пример определения типа MyPair.

```
3  public class MyPair<T, U> {
4      private T first;
5      private U second;
6
7      public MyPair(T pF, U pS){
8          first = pF;
9          second = pS;
10     }
11
12     public T getFirst() {
13         return first;
14     }
15
16     public U getSecond() {
17         return second;
18     }
19 }
```

Рисунок 7.1 – Обобщенный класс для представления пары элементов.

Здесь T и U – параметры типа. В приведенном классе присутствуют поля first типа T и second типа U, но таких типов в данном контексте не существует. Такое объявление становится возможным, так как при объявлении класса используется MyPair<T, U> вместо простого MyPair. Указание переменных типа T и U в угловых скобках после имени класса говорит об обобщенном характере определяемого класса. Вместо T и U при объявлении переменной

типа MyPair должны быть подставлены конкретные (не обобщенные) типы, например:

```
5  public class Starter {
6
7      public static void main(String[] args) {
8          MyPair<String, Integer> p1
9              = new MyPair<String, Integer>("Nikolaev E.I.", 26);
10
11         MyPair<String, Vector<Float>> p2
12         = new MyPair<String, Vector<Float>>(
13                 "Nikolaev E.I.",
14                 new Vector<Float>());
15     }
16 }
```

Рисунок 7.2 – Использование обобщенного класса.

При объявлении переменной p1 обобщенный класс конкретизируется путем присвоения переменным типа конкретных типов: T = String, U = Integer. Создается объект обобщенного типа, который содержит конкретные данные.

При создании объекта p2 используется конкретизация: T = String, U = Vector<Float>. То есть переменные типов могут представлять как угодно сложные типы.

Из примера видно, что обобщенные типы выступают своего рода фабрикой классов.

На рис. 7.3 приводится определение класса MyCollection.

```
4  public class MyCollection<T> {
5      private Vector<T> vec = new Vector<T>();
6
7      public MyCollection(T item){
8          vec.add(item);
9      }
10
11     public void Add(T item){
12         vec.add(item);
13     }
14
15     public void Remove(T item){
16         vec.remove(item);
17     }
18 }
```

Рисунок 7.3 – Объявление класса MyCollection.

Очевидно, что при объявлении обобщенного класса можно использовать любое количество переменных типов.

При использовании обобщенных типов возникает вопрос о возможных значениях переменных типов. Иногда класс или метод нуждается в наложении ограничений на переменные типов. Операция ограничения имеет следующий синтаксис:

```
public class MyCollection<T extends Comparable> {
```

Подобное определение указывает на то, что в качестве типа T может выступать любой тип, реализующий интерфейс Comparable.

Возможны такие варианты ограничения:

```
public class MyCollection<T extends Object & Comparable & Serializable> {
```

Ограничивающие типы разделяются знаком &, потому что запятые используются для разделения переменных типа.

Как и в наследовании Java, вы можете иметь столько интерфейсных подтипов, сколько хотите, но только один из ограничивающих типов может быть классом. Если вы используете для ограничения класс, он должен быть первым в списке ограничений.

Существуют ограничения, которые накладываются на код, использующий обобщенные типы:

1. Параметры типа не могут принимать примитивные типы. Нельзя подставить примитивные тип вместо параметра типа. То есть не бывает Pair<double>, а только Pair<Double>. Причина в «подчистке» типов (компилятор работает только с конкретными типами). После подчистки класс Pair имеет поля типа Object, и их нельзя использовать их для хранения значений double.

2. Исследование типов времени выполнения работает только с «сырыми» типами. Это означает, что код

```
if(p1 instanceof MyPair)
```

выполняет только проверке принадлежности переменной p1 типу MyPair без учета конкретных значений параметра типа. Нельзя проверить:

```
if(p1 instanceof MyPair<Float, String>)
```

3. Массивы параметризованных типов не разрешены.

4. Нельзя создавать экземпляры переменных типов. Например нельзя в методах обобщенного класса использовать: T ob = new T(), где T – параметр обобщенного типа.

5. Переменные типа в статическом контексте обобщенных классов не разрешены.

7.3 Обобщенные методы.

Кроме обобщенных классов, программист может определять обобщенные методы, которые принадлежат классу, который не является обобщенным. На рис. 6.4 представлено определение класса, содержащего метод WriteLine, который оборачивает вывод в консоль, выполняемый средствами библиотеки Java:

```
3  public class Console {
4      public static <T> void WriteLine(T mes){
5          System.out.println(mes);
6      }
7
8      public static <T> void Write(T mes){
9          System.out.print(mes);
10     }
11 }
```

Рисунок 7.4 – Обобщенный метод.

Обратите внимание, что переменная типа вставляется после модификаторов (public static в данном случае) и перед типом возврата. Использование созданных методов и консольный вывод программы показан на рис. 7.5.

```
public static void main(String[] args) {
    Console.WriteLine("Hello, world!");
    Console.Write(55.0f);
}
```

```
Hello, world!
55.0
```

а) б)

Рисунок 7.5 – Использование методов: а – код метода main; б – результат работы программы.

При вызове обобщенного метода в качестве параметров передаются действительные типы.

7.4 Ограничения переменных типов.

В предыдущих подразделах рассматривались обобщенные классы и методы, использующие переменные типа; при этом никаких сведений о типе, который представлен переменной типа программист не указывает. На практике может возникнуть потребность ограничить возможные варианты использования обобщения. Например: переменный тип может быть только классом, производным от Object (подтипом Object); переменный тип может быть только типом, реализующим определенный интерфейс и т.д. Может возникнуть необходимость наложить несколько ограничений на переменные типа.

На рис. 7.6 представлен пример подобных ограничений.

```
5  public class Console {
6      public static <T extends Object> void WriteLine(T mes){
7          System.out.println(mes);
8      }
9
10     public static <T extends Object & Comparable> void Write(T mes){
11         System.out.print(mes);
12     }
13 }
```

Рисунок 7.6 – Ограничение переменной типа.

Из примера видно, что ограничение указывается в угловых скобках после ключевого лова extends. Через знак & перечисляются интерфейсы, которые должен реализовывать переменный тип. Супертип может быть только один, он указывается первым, сразу после extends.

7.6. Методика и порядок выполнения работы

1. Реализуйте библиотечный класс на основе технологии обобщения в соответствии с вариантом индивидуального задания. При реализации библиотеки потребуется определить дополнительные классы для представления сущностей предметной области. Необходимо реализовать обобщение таким образом, чтобы не допустить передачу недопустимого типа, то есть требуется корректно определить ограничение переменной типа.

2. В функции main реализуйте логику приложения для демонстрации возможностей библиотеки, особенно возможности использования различных типов вместо переменных типа.

Индивидуальное задание.

Вариант	Библиотека для реализации
1, 11	Библиотека для манипулирования геометрическими объектами.
2, 12	Библиотека для манипулирования любыми числовыми типами.
3, 13	Библиотека для манипулирования должностными ставками (менеджер, директор, …).
4, 14	Библиотека для манипулирования объектами, представляющими печатные издания.
5, 15	Библиотека для манипулирования объектами, представляющими автомобили организации.
6, 16	Библиотека для манипулирования объектами, представляющими объекты недвижимости.
7, 17	Библиотека для манипулирования объектами, представляющими банковские услуги.
8, 18	Библиотека для манипулирования объектами, представляющими офисную технику.
9, 19	Библиотека для манипулирования объектами, представляющими основные объекты информационной системы торгового предприятия: склады, товары, товарные группы, …
10, 20	Библиотека для манипулирования объектами, представляющими основные объекты информационной системы управления предприятием: сотрудники, оргтехника, задания, проекты.

7.7. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.

3. Ответы на контрольные вопросы.

4. Диаграмма классов, 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.

7.8. Контрольные вопросы

1. Что такое обобщенный тип? Что такое переменная типа?

2. Для чего используется механизм обобщения? Поясните синтаксис при реализации данного механизма. Приведите примеры.

3. Приведите примеры использования механизма обобщения при реализации типов стандартной библиотеки Java.

4. Что такое ограничения на переменные типа? Для чего используется данный механизм?

5. Какие зарезервированные слова используются для реализации механизма обобщения?