|  |
| --- |
| **ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ** |
|  |
|  |
|  |
| **Практическое занятие №9.**  **Тема: Проектирование и использование обобщенных классов** |
|  |
|  |
|  |

**Цель:** получение практических навыков проектирования обобщений в Java и их использования для различных типов данных.

**Методические рекомендации**

**Введение в обобщения**

Применение обобщений позволило создавать классы, интерфейсы и методы, работающие безопасным по отношению к типам способом с разнообразными типами данных. Многие алгоритмы логически идентичны, независимо от того, к каким типам они применяются. Например, механизм, поддерживающий стеки, является одним и тем же в стеках, хранящих элементы типа Integer, String, Object или Thread. Благодаря обобщениям можно определить алгоритм один раз независимо от конкретного типа данных, а затем применять его к обширному разнообразию типов данных без каких-нибудь дополнительных усилий.

В следующей программе определены два класса. Первый - это обобщенный класс **Gen**, а второй **-** **GenDemo**, класс использующий класс **Gen**.

//Простой обобщенный класс. Здесь Т — это параметр типа, который

//будет заменен реальным типом при создании объекта.

**class Gen<T> {**

**Т ob;** // объявление объекта типа Т

//Передать конструктору ссылку на объект типа Т.

**Gen(Т о) {**

**ob = о ;**

**}**

//Вернуть ob.

**Т getob() {return ob;}**

// Показать тип Т.

**void showType() {**

**System.out.println("Типом T является" + ob.getClass().getName());**

**}**

**}**

// Демонстрация обобщенного класса**.**

**class GenDemo {**

**public static void main(String args[]) {**

//Создать Gen-ссылку для Integer.

**Gen<Integer>iOb;**

//Создать объект Gen<Integer> и присвоить ссылку на iOb (применение // автоупаковки для инкапсуляции значения 88 в объект Integer).

**iOb = new Gen<Integer>(88);**

//Показать тип данных, используемый iOb.

**iOb.showType();**

//Получить значение iOb, никакого приведения не нужно

**int v = iOb.getob();**

**System.out.println("значение: " + v);**

// Создание объект Gen для String.

**Gen<String> strOb = new Gen<String>("Обобщенный тест.");**

//Показать тип данных, используемый strOb.

**strOb.showType();**

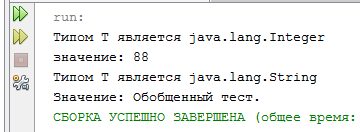
//Получить значение strOb. Опять -приведение не требуется.

**String str = strOb.getob();**

**System.out.println("Значение: " + str);**

**}**

Результат работы этой программы.



Угловые скобки являются указателем обобщения:

**class Gen<T> {**

Здесь **Т** —имя *параметра типа.*Это имя используется в качестве заполнителя, куда будет подставлено имя реального типа, переданного классу **Gen** при создании реальных типов.

Далее тип**Т** используется для объявления объекта по имени **ob**.

**Т ob;** // объявляет объект типа Т

Параметр типа **Т** также может быть использован для указания типа возвращаемого значения метода, как в случае метода **getob ( ):**

**Т getob() { return ob;**}

Так как объект **ob** тоже имеет тип **Т**, его тип совместим с типом, возвращаемым методом **getob ().**

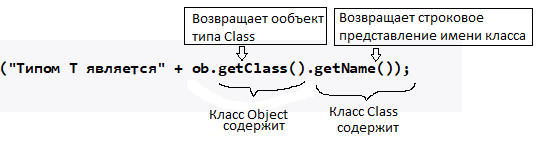
Особенности присутствуют в методе **showType ().**

**void showType() {**

**System.out.println("Типом T является" + ob.getClass().getName());**

**}**

Этот метод  отображает конкретный тип данных, который будет использоваться вместо типа **Т** при его использовании. Реализуется это вызовом метода **getName ()** объекта класса **Class**, возвращенным вызовом метода **getClass ()** объекта **ob**.



Метод **getClass ()** определен в классе **Object** и потому является членом всех классов. Он возвращает объект класса **Class**,  который содержит информацию о типе класса объекта, для которого он вызван. Класс **Class** определяет метод **getName (),** который возвращает строковое представление имени класса.

Класс **GenDemo** демонстрирует обобщенный класс **Gen**. Сначала он создает версию класса **Gen** для целых чисел, как показано ниже.

**Gen<Integer> iOb;**

Следующая строка присваивает объекту **iOb** ссылку на экземпляр целочисленной версии класса **Gen**.

**iOb = new Gen<Integer>(88);**

Далее в классе **GenDemo** объявляется объект типа **Gen<String>.**

**Gen<String>strOb = newGen<String>(" Обобщенный тест");**

Поскольку аргументом типа является **String**, класс **String**  подставляется вместо параметра **Т**внутри класса **Gen**. Это создает (концептуально) строковую версию класса **Gen**, что и демонстрируют остальные строки программы.

#### Обобщенный класс с двумя параметрами типа данных

Для обобщенного типа можно объявлять более одного параметра типа. Чтобы указать два или более параметров типа, используется разделенный запяты­ми список. Например, следующий класс  TwoGen - это вариант класса Gen, кото­рый принимает два параметра.

//Простой обобщенный класс с двумя параметрами типа: Т и V

**class TwoGen<T,V>{**

**T ob1;**

**V ob2;**

//Передать конструктору ссылки на объекты типа Т и типа V.

**TwoGen(Т o1, V о2) {**

**ob1 = o1; оЬ2 = о2;**

**}**

// Показать типы Т и V

**void showTypes() {**

**System.out.println("Тип T: " + ob1.getClass().getName());**

**System.out.println("Тип V: " + ob2.getClass().getName());**

**}**

**T getob1()**

**{ return ob1;}**

**V getob2()**

**{ return ob2;}**

**class SimpGen {**

**public static void main(String args[]) {**

**TwoGen<Integer, String> tgObj =**

**new TwoGen<Integer, String>(88, "Обобщения");**

//Показать типы

**tgObj.showTypes();**

//Получить и показать значения

**int v = tgObj.getob1();**

**System.out.println("Значение: " + v);**

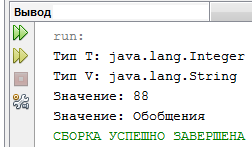
**String str = tgObj.getob2();**

**System.out.println("Значение: " + str);**

**}**

**}**

Результат работы этой программы.



В данном примере класс **Integer** подставляется вместо параметра **Т**, а класс **String** – вместо параметра **V**.

Хотя два аргумента в примере отличаются, допустимо передать в параметрах два одинаковых типа. Например, следующая строка кода вполне корректна.

**TwoGen<String,String> х=new TwoGen<String,String>("A","В");**

В этом случае оба аргумента параметров типа **V** и **Т**будут иметь тип **String**.

**Практикум**

В соответствии с *индивидуальным заданием* (см. приложение) разработать обобщенный класс, реализация которого (массивом объектов) обеспечивает определение двух вычисляемых показателей.

Отчет оформляется по общеустановленным правилам в *электронном виде* со следующим содержанием:

1. титульный лист,
2. тема и цель лабораторной работы,
3. задание на лабораторную работу,
4. текст программы с комментариями,
5. результаты работы программы и
6. выводы по созданному проекту и реализации обобщенного класса.

ПРИЛОЖЕНИЕ

**Варианты заданий**

|  |  |  |
| --- | --- | --- |
| **№**  **п/п** | **Обобщенный класс** | **Вычисляемые показатели** |
|  | Компьютер | Экземпляр с наибольшей оперативной памятью  Количество компьютеров заданной фирмы |
|  | Товар | Сумма покупки  Количество товаров одного типа |
|  | Грузовое транспортное средство | Максимальный пробег на полном бензобаке  Количество грузовиков заданной фирмы |
|  | Представитель университета | Количество студентов, обучающихся у конкретного преподавателя  Общее количество студентов |
|  | Транспортное средство | Максимальная емкость бензобака  Количество транспортных средств заданного типа |
|  | Компьютер | Самый дешевый компьютер  Количество компьютеров одной марки |
|  | Программный продукт | Последняя версия  Количество программных продуктов заданной фирмы |
|  | Документ | Количество документов на заданную фамилию  Количество документов, номера которых заканчиваются нулём |
|  | Периферийное устройство компьютера | Минимальная цена устройства  Количество устройств определённой фирмы |
|  | Строительный товар | Сумма покупки  Количество товаров заданного типа |
|  | Страховой полис | Количество полисов на заданную фамилию  Полис с наименьшим номером |
|  | Периферийное устройство компьютера | Средняя цена  Количество устройств заданной фирмы |
|  | Недвижимость | Максимальная жилая площадь квартиры  Количество квартир заданной компании |
|  | Часы | Самый дорогой экземпляр  Количество часов определённого производителя |
|  | Товар | Количество товаров заданной фирмы  Общая сумма покупки |
|  | Магнитная карта для проезда на транспорте | Количество карт без поездок  Количество карт заданного типа |
|  | Транспортное средство | Количество транспортных средств, выпущенных после определённого года  Количество транспортных средств заданной фирмы |
|  | Канцелярские товары | Количество товаров определённой фирмы  Общая сумма покупки |
|  | Программный продукт | Количество продуктов, выпущенных заданной фирмой  Самая старая версия |
|  | Документ | Документ, выданный раньше всех других  Количество документов определённого типа |
|  | Путешествие (тур) | Самый дешевый тур на 7 и более дней  Число туров в выбранную страну |
|  | Товар | Самый дорогой товар  Количество товаров определённого бренда |
|  | Страховой полис | Полис с максимальным сроком действия  Количество полисов на заданную фамилию |
|  | СУБД | Количество СУБД заданного производителя  Последняя по году выпуска версия |
|  | Предприятие малого бизнеса | Название предприятия с максимальным числом сотрудников  Число парикмахерских |
|  | Представление | Минимальное число зрителей в зале  Представление с самым коротким названием |
|  | Запоминающее устройство | Экземпляр с минимальным размером  Число устройств одной фирмы |
|  | Принтер | Представитель с наибольшей производительностью  Число принтеров одного производителя |
|  | Книга | Количество книг одного автора  Название книги с наибольшим числом страниц |
|  | Телефон | Самая новая модель  Число телефонов одной фирмы |