БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

по дисциплине «Программирование на языке Java»

Выполнила:

студентка группы 10702121

Алюнина П.А.

Проверил:

Сидорик В.В.

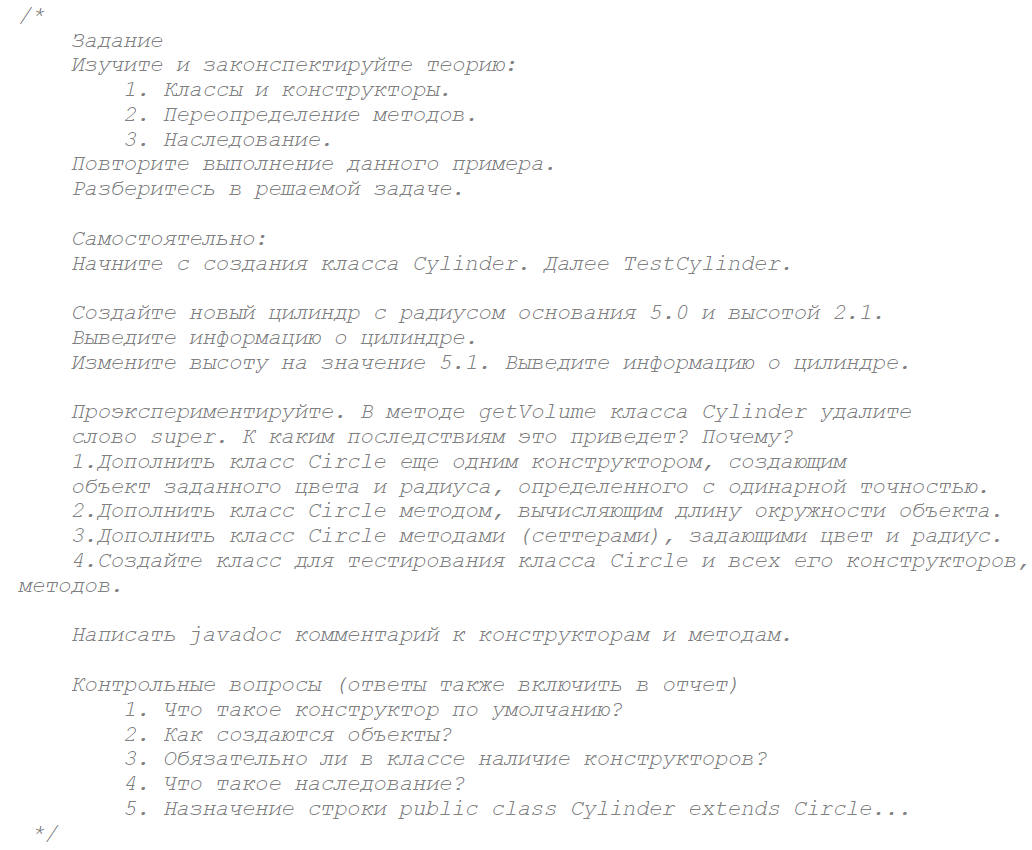
Минск 2023

**Цель**: изучить структуру и принцип работы классов и конструкторов, переопределение методов и наследование, динамического массива в языке программирования Java.

**Задачи**:

1. Изучить и законспектировать теорию по структуре и принципу работы классов и конструкторов, переопределение методов и наследование.
2. Выполнить программы.
   1. Выполнить программу демонстрирующую переопределение методов и наследование.
   2. Выполнить программу демонстрирующую работу таких ключевых слов как this и super.
   3. Выполнить программу демонстрирующую работу с использованием динамического массива.
3. Разобраться в работе коде программ с скриншотами результатов после выполнения программ.
4. Ответить на поставленные контрольные вопросы, ссылаясь на предоставленную литературу в начале семестра.
5. Составить глоссарий по темам классы и динамические массивы.

**Задание №61: TestCylinder**

****

Чтобы объявить один класс наследником от другого, надо использовать после имени класса-наследника ключевое слово **extends**, после которого идет имя базового класса.

public class Cylinder extends Circle {

}

Если в базовом классе определены конструкторы, то в конструкторе производного классы необходимо вызвать один из конструкторов базового класса с помощью ключевого слова **super**.

public class Cylinder extends Circle {public Cylinder() { *// Constructor 1* super(); *// вызов конструктора суперкласса Circle()*  
 }

}

Производный класс может определять свои методы, а может переопределять методы, которые унаследованы от базового класса. Перед переопределяемым методом указывается аннотация @Override. Данная аннотация в принципе необязательна.

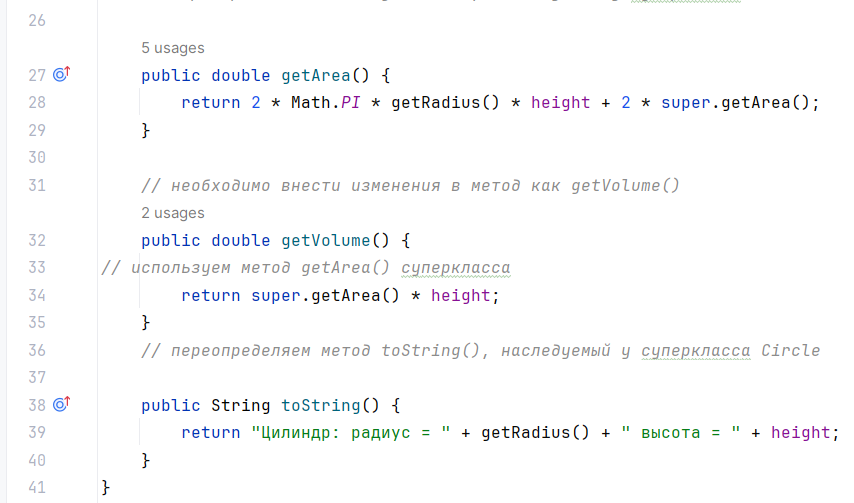
При переопределении метода он должен иметь уровень доступа не меньше, чем уровень доступа в базовом класса. Например, если в базовом классе метод имеет модификатор public, то и в производном классе метод должен иметь модификатор **public**.

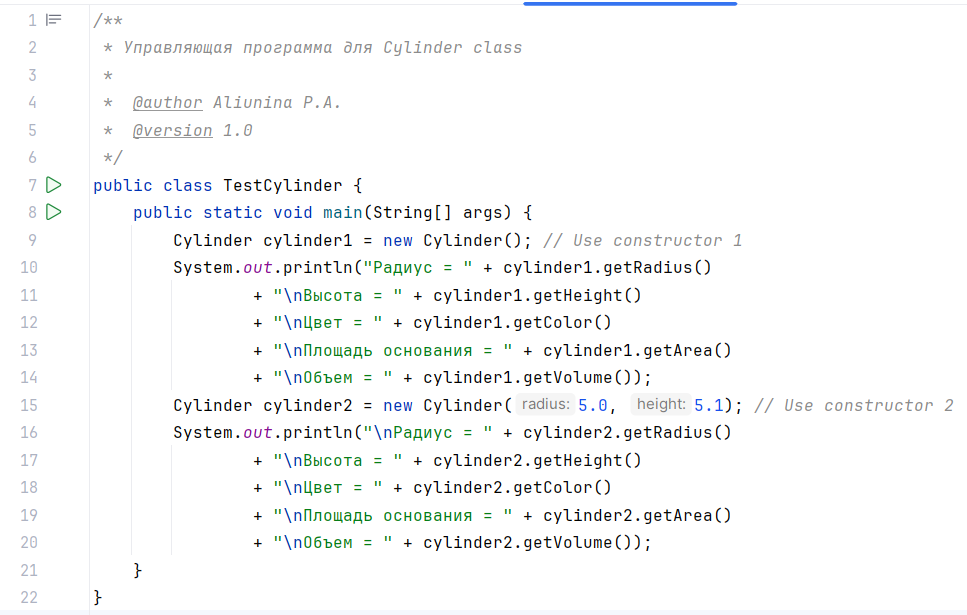
public class Cylinder extends Circle {  
 *// Переопределяем метод getArea(), наследуемый у суперкласса Circle* @Override  
 public double getArea() {  
 }

}

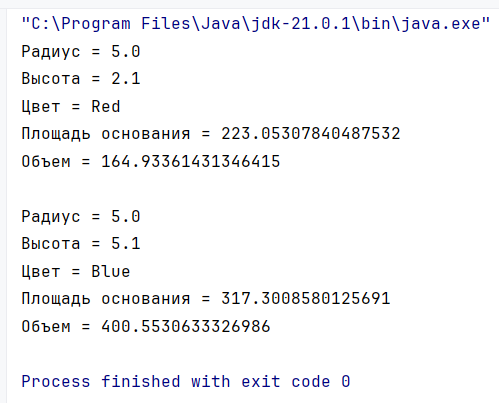
**Код программы**

****

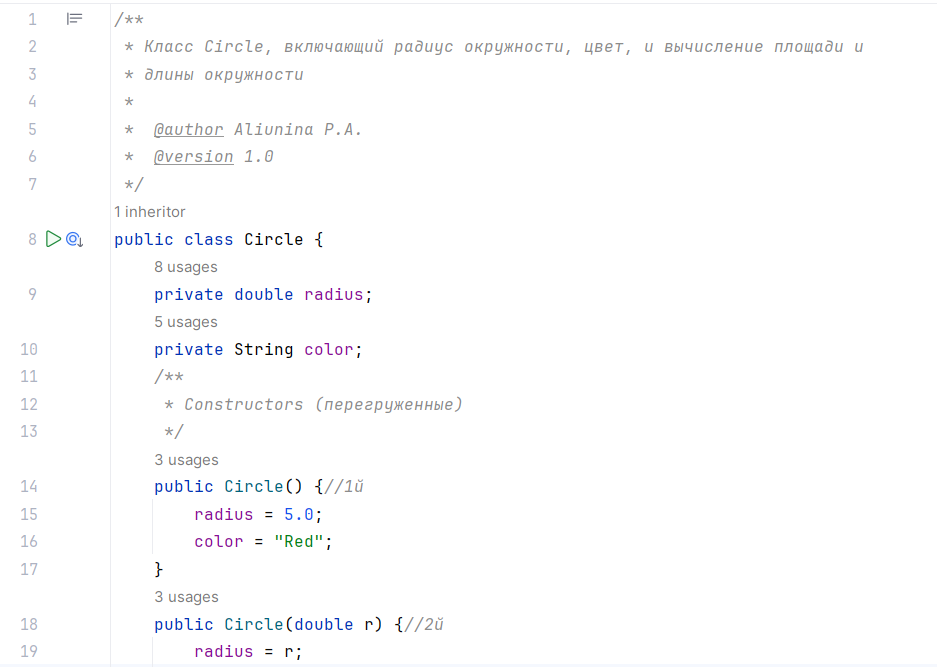
****

****

**Результаты выполнения**

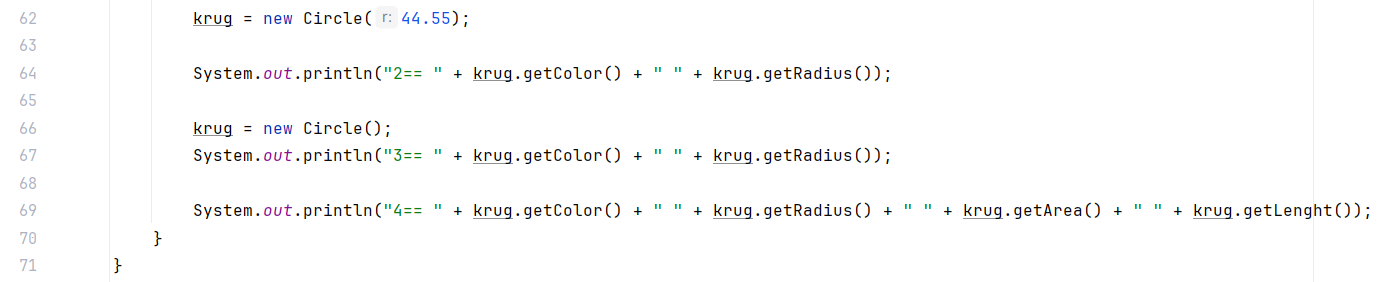
****

**Индивидуальное задание**

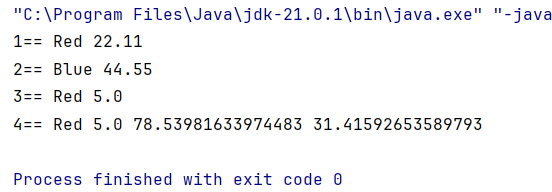
****

****

****

****

**Результаты выполнения**

****

**Ответы на контрольные вопросы**

1. Что такое конструктор по умолчанию?

Это конструктор, который не имеет параметров. Конструктор по умолчанию может объявляться в классе явным образом или генерироваться автоматически.

Для того, чтобы иметь реализацию конструктора по умолчанию и объявлять объект класса с использованием конструктора по умолчанию, его нужно задавать явно.

1. Как создаются объекты?

Для создания объекта Box используется выражение new Box(). Оператор new выделяет память для объекта Box. И затем вызывается конструктор по умолчанию, который не принимает никаких параметров. В итоге после выполнения данного выражения в памяти будет выделен участок, где будут храниться все данные объекта Box. А переменная myBox получит ссылку на созданный объект.

Пример:

Box myBox = new Box()

1. Обязательно ли в классе наличие конструкторов?

Конструктор определяет действия, выполняемые при создании объекта класса, и является важной частью класса. Как правило, программисты стараются явно указать конструктор. Если явного конструктора нет, то Java автоматически создаст его для использования по умолчанию.

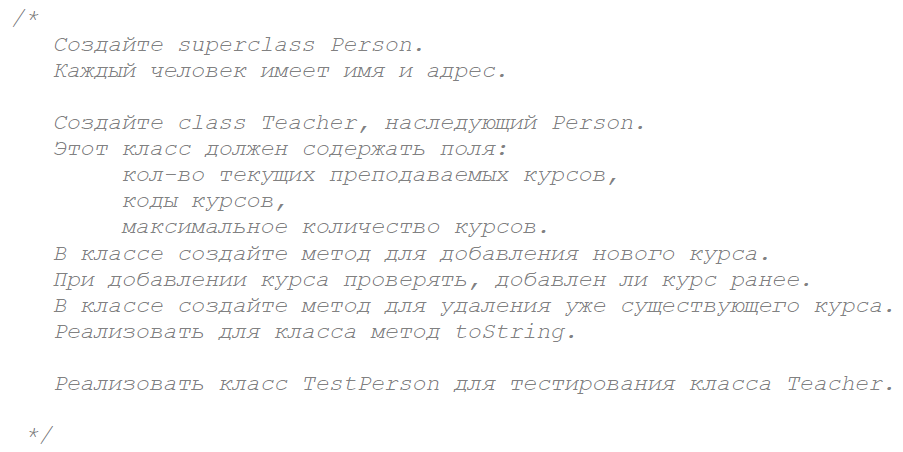
1. Что такое наследование?

Одним из фундаментальных понятий объектно-ориентированного программирования является наследование, поскольку оно позволяет создавать иерархические классификации. Используя наследование, можно создать общий класс, который определяет характеристики, общие для набора связанных элементов. Затем этот класс может наследоваться другими, более специализированными классами, каждый из которых будет добавлять свои уникальные характеристики.

1. Назначение строки public class Cylinder extends Circle...

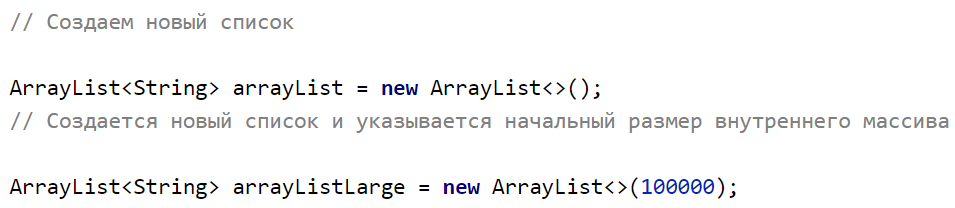
Это значит, что класс Cylinder наследует класс Circle, то есть класс Cylinder наследует все те же поля и методы, которые есть в классе Circle

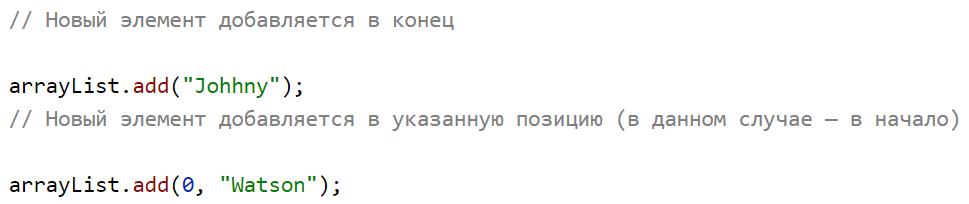
**Задание №62: TestTeacher**

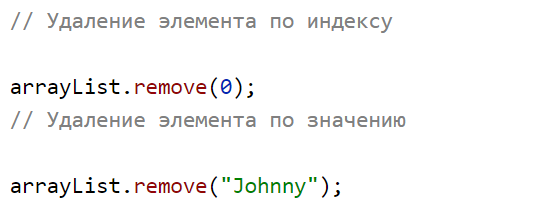


**Динамический массив** используется для обработки наборов однородных данных, размер которых неизвестен на момент написания программы. В Java эту роль играют в основном классы ArrayList и LinkedList.

*ArrayList* — это классический массив, который может расширяться в момент выполнения программы. В его основе лежит обычный массив: его размер при создании — 10 элементов. При увеличении размера емкость увеличивается.







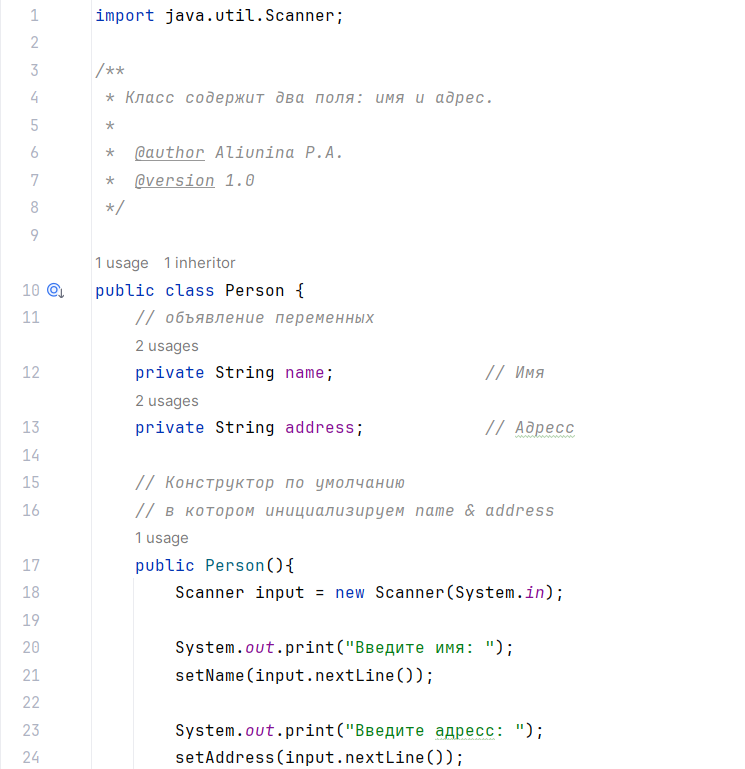
При поиске элемента в списке можно использовать методы **indexOf()** и **lastIndexOf()**:

int index = arrayList.indexOf("значение");

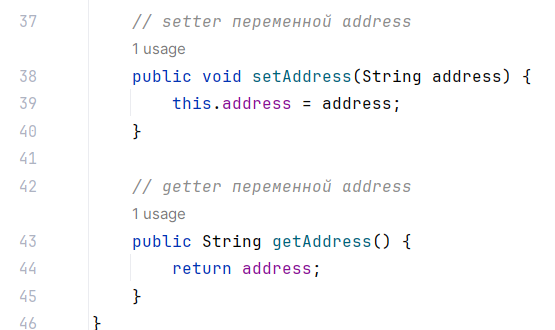
Также можно использовать метод **contains()** для получения факта наличия элемента в списке:

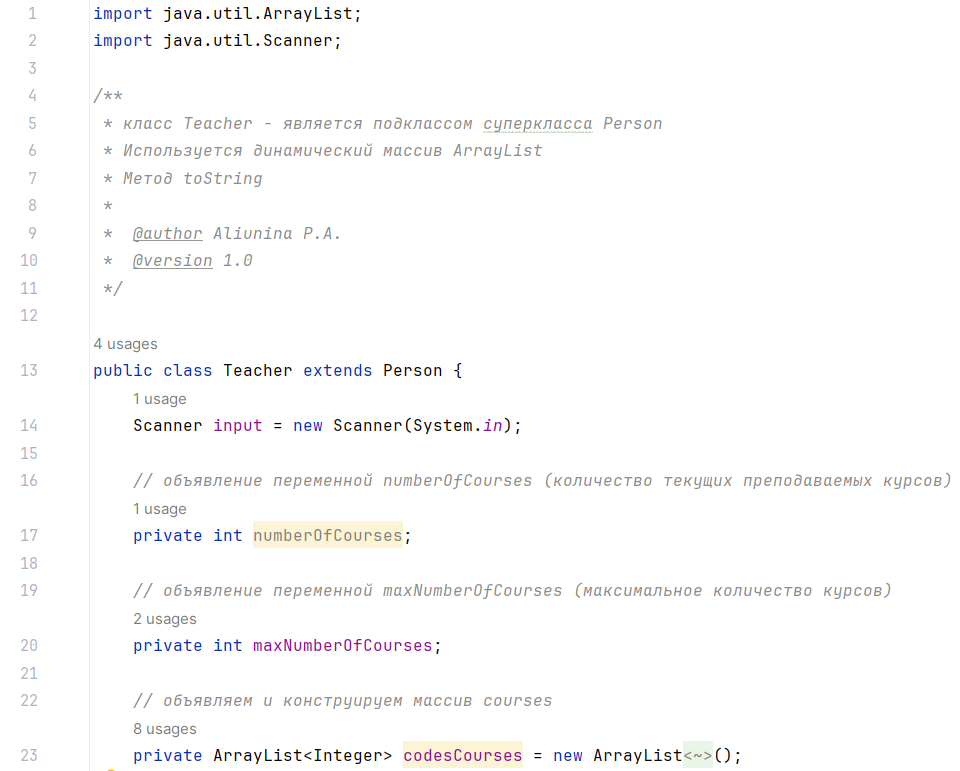
boolean bool = arrayList.contains("значение");

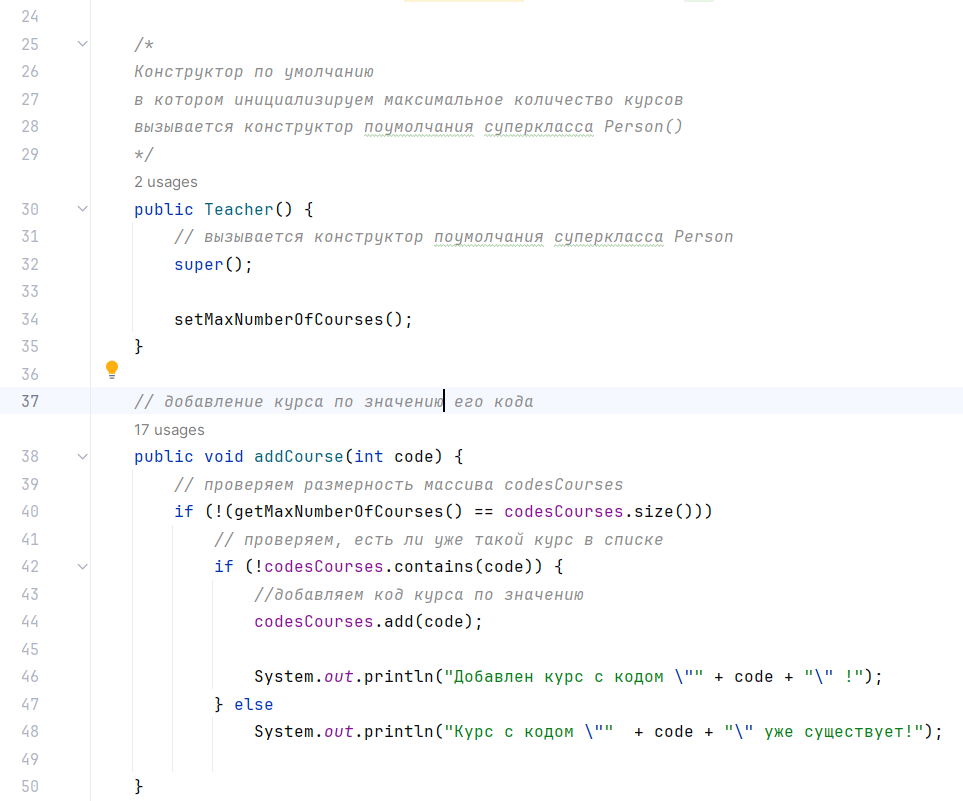
**Код программы**







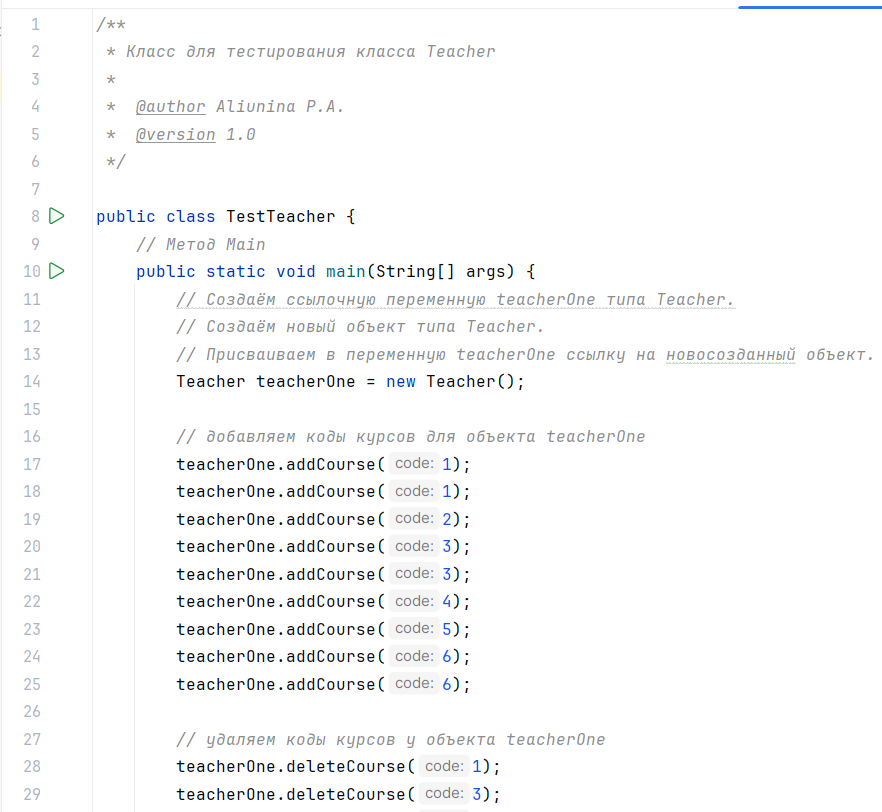


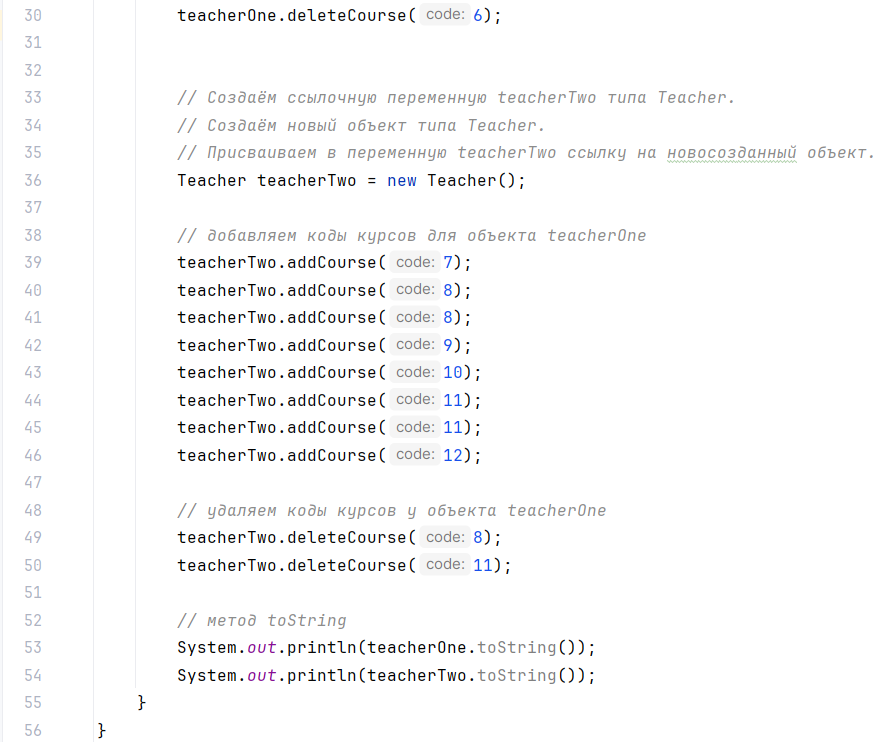


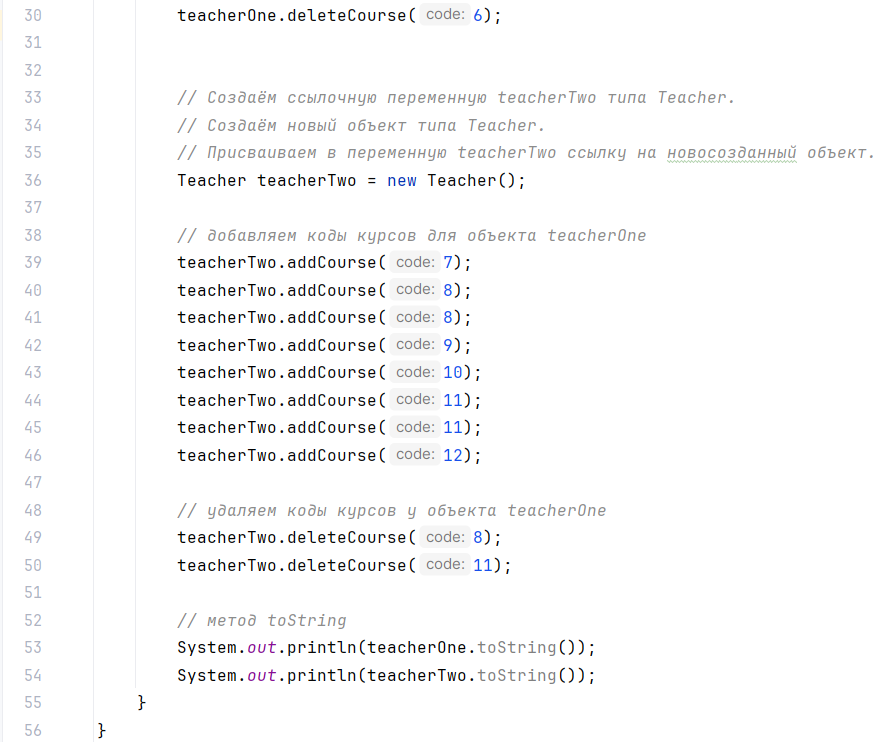




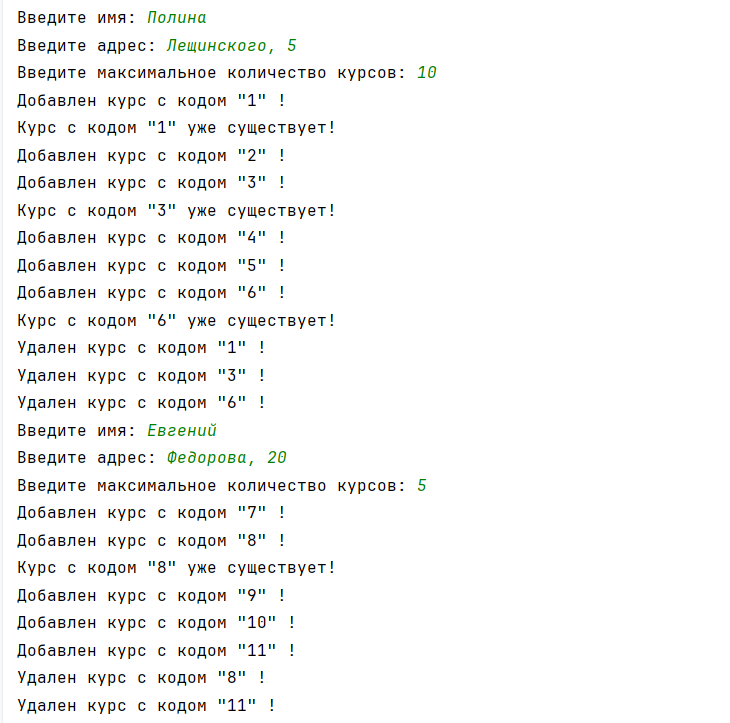


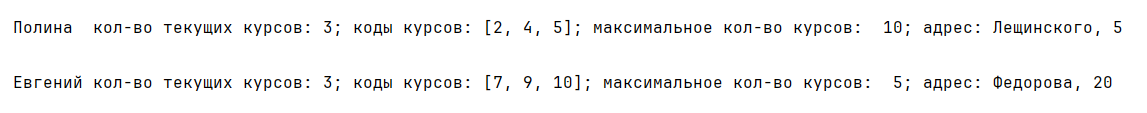






**Результаты выполнения**

****

****

Глоссарий

**Динамический массив** (англ. *dynamic array*) – это массив, который умеет изменять свой размер во время выполнения программы.

**Список массивов** (англ. *arrayList*)— это классический массив, который может расширяться в момент выполнения программы.

**Наследование** (англ. *inheritance*) – это механизм, который позволяет строить на основе первоначального класса другие, добавляя в классы новые поля данных и методы.

**Суперкласс** (англ. *superclass*) – это наследуемый класс.

**Подкласс** (англ. *subclass*) – это наследующий класс или специализированная версия суперкласса.