|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**ОТЧЁТ ПО ПРАКТИЧЕСКИМ РАБОТАМ №1-4**

**по дисциплине «Программирование на языке Джава»**

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил:  Студент группы Шмелев Д.В. | « » 2021г. | (подпись) | Шмелев Д.В. |
|  |  |  |  |
| Принял:  Ассистент кафедры ИиППО ИИТ | « » 2021г. | (подпись) | Литвинов В.В. |

Москва 2021 г.

Оглавление

[ПРАКТИЧЕСКАЯ РАБОТА №1. КЛАССЫ, КАК НОВЫЕ ТИПЫ ДАННЫХ. ПОЛЯ ДАННЫХ И МЕТОДЫ 3](#_Toc89904418)

[Цель работы 3](#_Toc89904419)

[Теоретическое введение 3](#_Toc89904420)

[Постановка задачи 4](#_Toc89904421)

[Программный код 5](#_Toc89904422)

[Вывод программы 6](#_Toc89904423)

[Вывод 6](#_Toc89904424)

[ПРАКТИЧЕСКАЯ РАБОТА №2. ИСПОЛЬЗОВАНИЕ UML ДИАГРАММ В ОБЪЕКТНООРИЕНТИРОВАННОМ ПРОГРАММИРОВАНИИ 7](#_Toc89904425)

[Цель работы 7](#_Toc89904426)

[Теоретическое введение 7](#_Toc89904427)

[Постановка задачи 8](#_Toc89904428)

[Программный код 9](#_Toc89904429)

[Вывод программы 10](#_Toc89904430)

[Вывод 10](#_Toc89904431)

[ПРАКТИЧЕСКАЯ РАБОТА № 3. НАСЛЕДОВАНИЕ. АБСТРАКТНЫЕ СУПЕРКЛАССЫ И ИХ ПОДКЛАССЫ В JAVA. 11](#_Toc89904432)

[Цель работы 11](#_Toc89904433)

[Теоретическое введение 11](#_Toc89904434)

[Постановка задачи 11](#_Toc89904435)

[Программный код 12](#_Toc89904436)

[ 12](#_Toc89904437)

[Вывод программы 12](#_Toc89904438)

[ 12](#_Toc89904439)

[Вывод 12](#_Toc89904440)

[ПРАКТИЧЕСКАЯ РАБОТА № 13](#_Toc89904441)

[Цель работы 13](#_Toc89904442)

[Теоретическое введение 13](#_Toc89904443)

[Постановка задачи 13](#_Toc89904444)

[Программный код 13](#_Toc89904445)

[ 13](#_Toc89904446)

[Вывод программы 13](#_Toc89904447)

[ 13](#_Toc89904448)

[Вывод 13](#_Toc89904449)

[ПРАКТИЧЕСКАЯ РАБОТА № 14](#_Toc89904450)

[Цель работы 14](#_Toc89904451)

[Теоретическое введение 14](#_Toc89904452)

[Постановка задачи 14](#_Toc89904453)

[Программный код 14](#_Toc89904454)

[ 14](#_Toc89904455)

[Вывод программы 14](#_Toc89904456)

[ 14](#_Toc89904457)

[Вывод 14](#_Toc89904458)

[ПРАКТИЧЕСКАЯ РАБОТА № 15](#_Toc89904459)

[Цель работы 15](#_Toc89904460)

[Теоретическое введение 15](#_Toc89904461)

[Постановка задачи 15](#_Toc89904462)

[Программный код 15](#_Toc89904463)

[ 15](#_Toc89904464)

[Вывод программы 15](#_Toc89904465)

[ 15](#_Toc89904466)

[Вывод 15](#_Toc89904467)

# ПРАКТИЧЕСКАЯ РАБОТА №1. КЛАССЫ, КАК НОВЫЕ ТИПЫ ДАННЫХ. ПОЛЯ ДАННЫХ И МЕТОДЫ

## Цель работы

Цель данной практической работы – освоить на практике работу с классами на Java.

## Теоретическое введение

В Java, класс является определением объектов одного и того же вида. Другими словами, класс — это тип данных, создаваемый программистом для решения задач. Он представляет из себя шаблон, или прототип, который определяет и описывает статические свойства и динамическое поведение, общие для всех объектов одного и того же вида.

Экземпляр класса - реализация конкретного объекта типа класс. Другими словами, экземпляр экземпляра класса. Все экземпляры класса имеют аналогичные свойства, как задано в определении класса. Например, вы можете определить класс с именем "Студент " и создать три экземпляра класса "Студент": " Петр", " Павел" и " Полина ". Термин "Объект " обычно относится к экземпляру класса. Но он часто используется свободно, которые могут относиться к классу или экземпляру.

Графически можно представить класс в виде UML диаграммы как прямоугольник в виде как трех секций, в котором присутствует секция наименования класса, секция инкапсуляции данных и методов (функций или операций) класса.

Рассмотрим подробнее диаграмму класса. Имя (или сущность): определяет класс.

Переменные (или атрибуты, состояние, поля данных класса): содержит статические атрибуты класса, или описывают свойства класса (сущности предметной области).

Методы (или поведение, функции, работа c данными): описывают динамическое поведение класса. Другими словами, класс инкапсулирует статические свойства (данные) и динамические модели поведения (операции, которые работают с данными) в одном месте (“контейнере” или “боксе”), представленном на рисунке в виде прямоугольника.

1) Класс, тип данных, определяемый программистом, абстрактный тип данных, повторно-используемый программный объект, который имитирует реальные сущности предметной области. Класс можно представить графически в виде контейнера на UML диаграмме, который состоит из трех условных частей и содержит имя, переменные и методы.

2) Класс инкапсулирует статическое состояние объекта, его атрибуты или свойства данных в виде переменных класса и поведение объекта в виде методов, которые могут реализовывать определенные алгоритмы.

3) Значения переменных или поля данные составляют его состояние. Методы создает свои модели поведения.

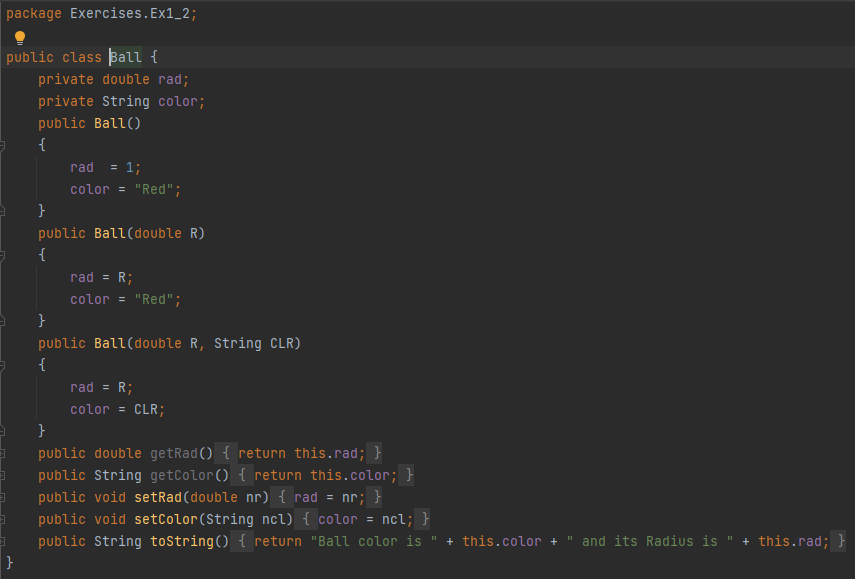
Экземпляр класса — это представление (или реализация) конкретного представителя класса.

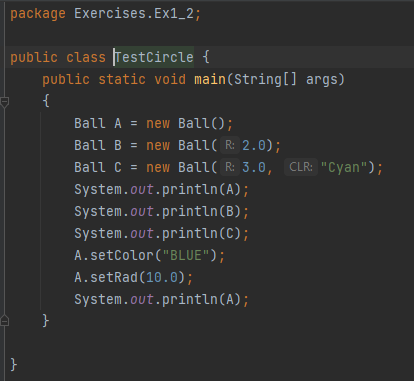
## Постановка задачи

Необходимо реализовать простейший класс на языке программирования Java. Не забудьте добавить метод toString() к вашему классу. Так-же в программе необходимо предусмотреть класс-тестер для тестирования класса и вывода информации об объекте.

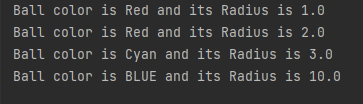
Упражнение 2. Реализуйте простейший класс «Мяч»

## Программный код





## Вывод программы



## Вывод

В результате выполнения данной практической работы мы смогли освоить на практике работу с классами на Java, научились писать классы, методы, переменные и взаимодействовать с ними.

# ПРАКТИЧЕСКАЯ РАБОТА №2. ИСПОЛЬЗОВАНИЕ UML ДИАГРАММ В ОБЪЕКТНООРИЕНТИРОВАННОМ ПРОГРАММИРОВАНИИ

## Цель работы

Цель данной работы – это научиться работать с UML-диаграммами классов.

## Теоретическое введение

Язык моделирования Unified Modeling Language (UML) является стандартом де-факто с 1998 года для проектирования и документирования объектно-ориентированных программ. Средствами UML в виде диаграмм можно графически изобразить класс и экземпляр класса.

Графически представляем класс в виде прямоугольника, разделенного на три области – область именования класса, область инкапсуляции данных и область операций (методы).

Имя (или сущность) : определяет класс.

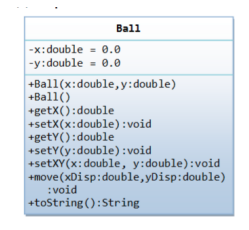
Переменные (или атрибуты, состояние, поля данных класса): содержит статические атрибуты класса, или описывают свойства класса (сущности предметной области).

Методы (или поведение, функции, работа c данными): описывают динамическое поведение класса. Другими словами, класс инкапсулирует статические свойства (данные) и динамические модели поведения (операции, которые работают с данными) в одном месте (“коробке” или прямоугольнике).

Приведенные выше диаграммы классов описаны в соответствии с UML нотацией. Класс представляется в этой нотации как прямоугольник, разделенный на три области, одна содержит название, две вторых содержат переменные (поля данных класса) и методы класса, соответственно. Имя класса выделено жирным шрифтом и находится посредине. Экземпляр (объект класса) также представляется в виде прямоугольника, разделенного на три части, в первой части помещается надпись с именем экземпляра, например в instanceName:Classname и выделенная подчеркиванием ( название\_экземпляра : имя\_класса).

## Постановка задачи

По UML диаграмме класса, представленной на рисунке 2.5. написать программу, которая состоит из двух классов. Один из них Ball должен реализовывать сущность мяч, а другой с названием TestBall тестировать работу созданного класса. Класс Ball должен содержать реализацию методов, представленных на UML. Диаграмма на рисунке описывает сущность Мяч написать программу



Класс Ball моделирует движущийся мяч. В состав класса входят:

• Две переменные с модификатором private (поля данных класса): х, у, которые описывают положение мяча на поле.

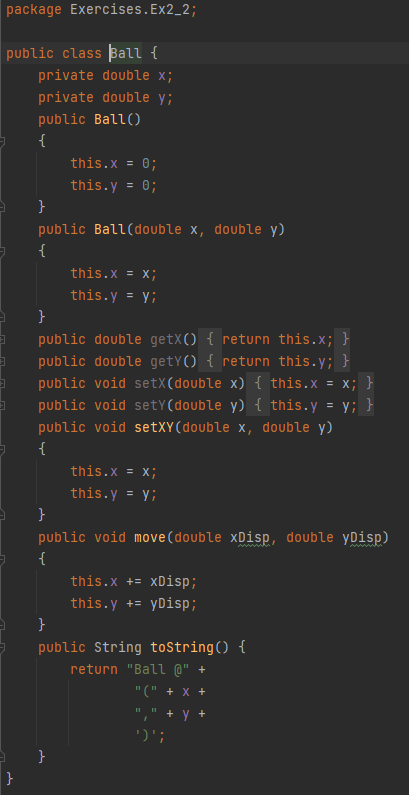
• Конструкторы, public методы получения и записи значений для private переменных.

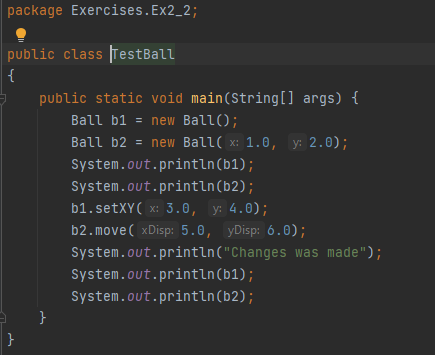
• Метод setXY (), который задает положение мяча и метод setXYSpeed(), чтобы задать скорость мяча

• Метод move() , позволяет переместить мяч, так что что увеличивает х и у на данном участке на xDisp и yDisp, соответственно.

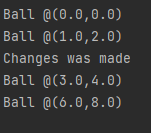
• Метод toString(), который возвращает "Ball @ (х , у) " .

## Программный код





## Вывод программы



## Вывод

В результате выполнения данной практической работы, мы научились создавать классы, основываясь на UML диаграммах

# ПРАКТИЧЕСКАЯ РАБОТА № 3. НАСЛЕДОВАНИЕ. АБСТРАКТНЫЕ СУПЕРКЛАССЫ И ИХ ПОДКЛАССЫ В JAVA.

## Цель работы

Цель работы: освоить на практике работу с абстрактными классами и наследованием на Java.

## Теоретическое введение

Класс, содержащий абстрактные методы, называется абстрактным классом. Такие классы при определении помечаются ключевым словом abstract.

Абстрактный метод внутри абстрактного класса не имеет тела, только прототип. Он состоит только из объявления и не имеет тела.

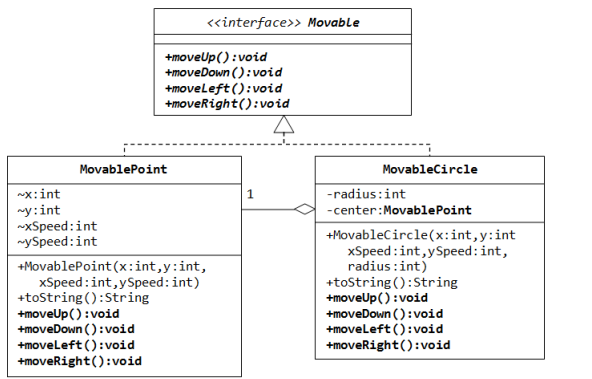
По сути, мы создаём шаблон метода. Например, можно создать абстрактный метод для вычисления площади фигуры в абстрактном классе Фигура. А все другие производные классы от главного класса могут уже реализовать свой код для готового метода. Ведь площадь у прямоугольника и треугольника вычисляется по разным алгоритмам и универсального метода не существует.

Если вы объявляете класс, производный от абстрактного класса, но хотите иметь возможность создания объектов нового типа, вам придётся предоставить определения для всех абстрактных методов базового класса. Если этого не сделать, производный класс тоже останется абстрактным, и компилятор заставит пометить новый класс ключевым словом abstract.

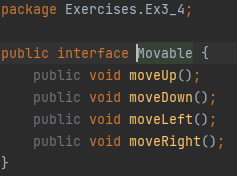
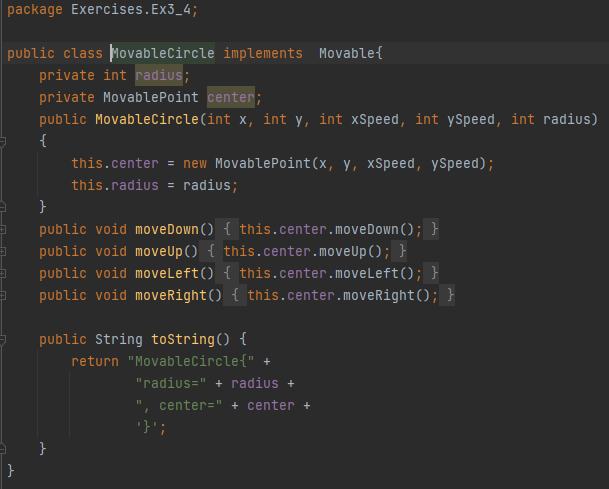
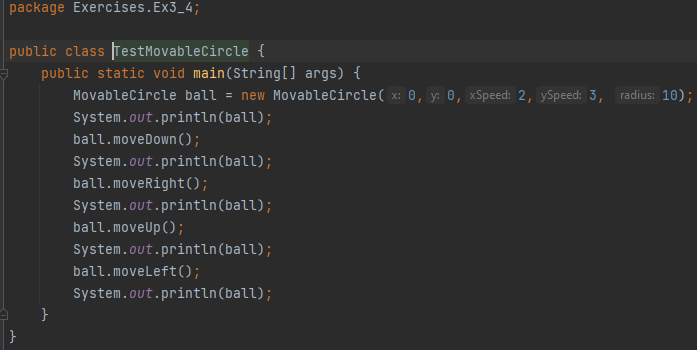
Абстрактный класс не может содержать какие-либо объекты, а также абстрактные конструкторы и абстрактные статические методы. Любой подкласс абстрактного класса должен либо реализовать все абстрактные методы суперкласса, либо сам быть объявлен абстрактным.

## Постановка задачи

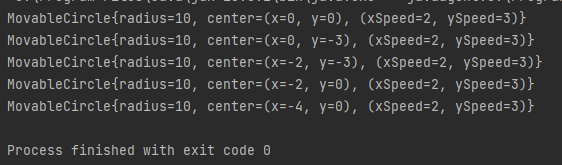
Напишите два класса MovablePoint и MovableCircle - которые реализуют интерфейс Movable.



## Программный код

## Вывод программы



## Вывод

В результате выполнения данной практической работы, мы научились работать и создавать абстрактные классы, изучили наследование, и научились создавать и реализовывать интерфейсы.

# ПРАКТИЧЕСКАЯ РАБОТА № 4. СОЗДАНИЕ GUI. СОБЫТИЙНОЕ ПРОГРАММИРОВАНИЕ В JAVA.

## Цель работы

Цель работы: введение в событийное программирование на языке Java.

## Теоретическое введение

Данная практическая работа посвящена закреплению практических навыков по созданию приложений на Java c использованием следующих элементов GUI:

• Текстовые поля и области ввода текста; • Менеджеры компоновки компонентов;

• Слушатель мыши;

• Создание меню.

Text Fields - текстовое поле или поля для ввода текста (можно ввести только одну строку). Примерами текстовых полей являются поля для ввода логина и пароля, например, используемые, при входе в электронную почту.

Пример создания объекта класса JTextField:

JTextField jta = new JTextField (10);

В параметрах конструктора задано число 10, это количество символов, которые могут быть видны в текстовом поле. Текст веденный в поле JText может быть возвращен с помощью метода getText(). Также в поле можно записать новое значение с помощью метода setText(String s).

Как и у других компонентов, мы можем изменять цвет и шрифт текста в текстовом поле.

Ответственность за выполнение проверки на наличие ошибок в коде лежит полностью на программисте, например, чтобы проверить произойдет ли ошибка, когда в качестве входных данных в JTextField ожидается ввод числа. Компилятор не будет ловить такого рода ошибку, поэтому ее необходимо обрабатывать пользовательским кодом.

Компонент TextArea похож на TextField, но в него можно вводить более одной строки. В качестве примера TextArea можно рассмотреть текст, который мы набираем в теле сообщения электронной почты.

Менеджер BorderLayout: Разделяет компонент на пять областей (WEST, EAST, NOTH, SOUTH and Center). Другие компоненты могут быть добавлены в любой из этих компонентов пятерками.

Обратите внимание, что мы можем, например, добавить панели JPanel в эти области и затем добавлять компоненты этих панелей. Мы можем установить расположение этих JPanel используя другие менеджеры.

С помощью менеджера GridLayout компонент может принимать форму таблицы, где можно задать число строк и столбцов.

Если компоненту GridLayout задать 3 строки и 4 столбца, то компоненты будут принимать форму таблицы, показанной выше, и будут всегда будут добавляться в порядке их появления

Иногда бывает нужно изменить размер и расположение компонента в контейнере. Таким образом, мы должны указать программе не использовать никакой менеджер компоновки, то есть (setLayout (нуль)).

Добавление меню в программе Java проста. Java определяет три компонента для обработки:

• JMenuBar: который представляет собой компонент, который содержит меню.

• JMenu: который представляет меню элементов для выбора.

• JMenuItem: представляет собой элемент, который можно кликнуть из меню.

Подобно компоненту Button (на самом деле MenuItems являются подклассами класса AbstractButton). Мы можем добавить ActionListener к ним так же, как мы делали с кнопками

## Постановка задачи

Напишите интерактивную программу с использованием GUI имитирует таблицу результатов матчей между командами Милан и Мадрид. Создайте JFrame приложение, у которого есть следующие компоненты GUI: • одна кнопка JButton labeled “AC Milan”

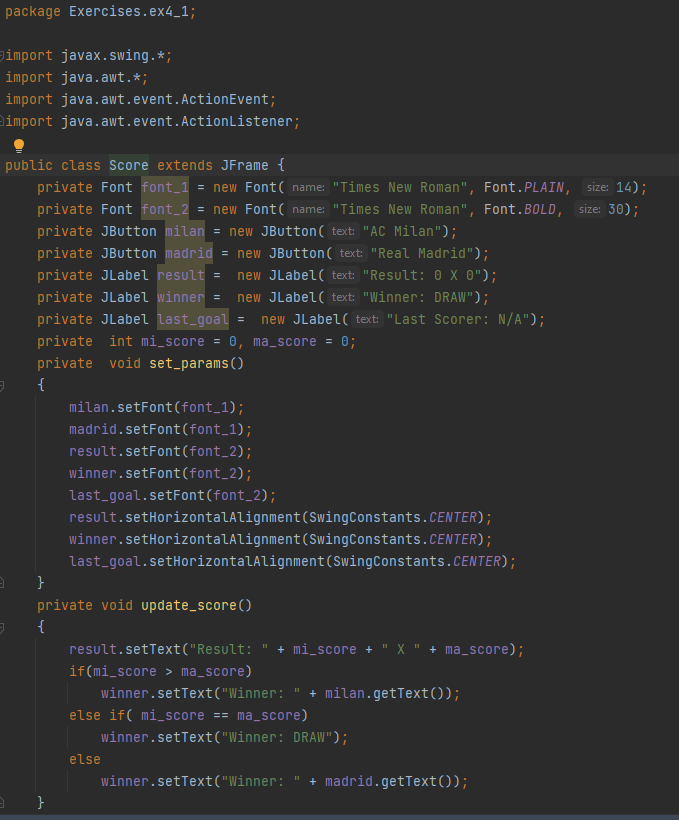
• другая JButton подписана “Real Madrid” • надпись JLabel содержит текст “Result: 0 X 0”

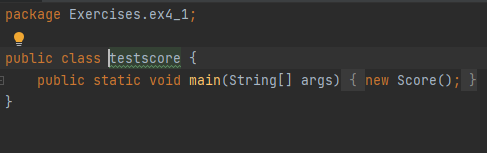
• надпись JLabel содержит текст “Last Scorer: N/A”

• надпись Label содержит текст “Winner: DRAW”;

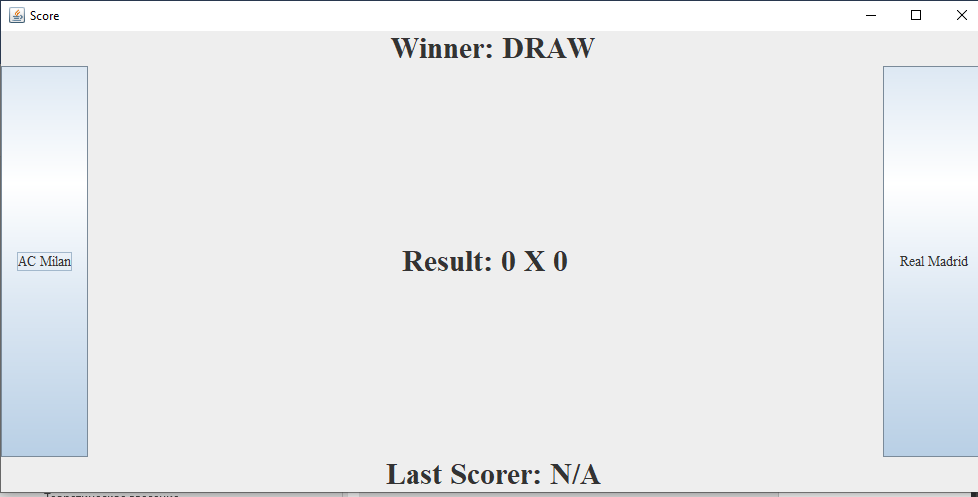
Всякий раз, когда пользователь нажимает на кнопку AC Milan, результат будет увеличиваться для Милана, сначала 1 X 0, затем 2 X 0 и так далее. Last Scorer означает последнюю забившую команду. В этом случае: AC Milan. Если пользователь нажимает кнопку для команды Мадрид, то счет приписывается ей. Победителем становится команда, которая имеет больше кликов кнопку на соответствующую, чем другая.

## Программный код



## Вывод программы

## Вывод

В результате выполнения …

# ПРАКТИЧЕСКАЯ РАБОТА № 5. ПРОГРАММИРОВАНИЕ РЕКУРСИИ В JAVA

## Цель работы

Цель работы: разработка и программирование рекурсивных алгоритмов на языке Java.

## Теоретическое введение

В контексте языка программирования рекурсия — это некий активный метод (или подпрограмма) вызываемый сам по себе непосредственно, или вызываемой другим методом (или подпрограммой) косвенно. В первую очередь надо понимать, что рекурсия — это своего рода перебор. Вообще говоря, всё то, что решается итеративно можно решить рекурсивно, то есть с использованием рекурсивной функции.

Так же, как и у перебора (цикла) у рекурсии должно быть условие остановки — базовый случай (иначе также, как и цикл, рекурсия будет работать вечно — infinite). Это условие и является тем случаем, к которому рекурсия идет (шаг рекурсии). При каждом шаге вызывается рекурсивная функция до тех пор, пока при следующем вызове не сработает базовое условие и не произойдет остановка рекурсии (а точнее возврат к последнему вызову функции). Всё решение сводится к поиску решения для базового случая. В случае, когда рекурсивная функция вызывается для решения сложной задачи (не базового случая) выполняется некоторое количество рекурсивных вызовов или шагов, с целью сведения задачи к более простой. И так до тех пор, пока не получим базовое решение.

Итак, рекурсивная функция состоит из:

• условие остановки или же базового случая или условия;

• условие продолжения или шага рекурсии — способ сведения сложной задачи к более простым подзадачам

## Постановка задачи

12. Вывести нечетные числа последовательности. Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Выведите все нечетные числа из этой последовательности, сохраняя их порядок. В этой задаче нельзя использовать глобальные переменные и передавать какие-либо параметры в рекурсивную функцию. Функция получает данные, считывая их с клавиатуры. Функция не возвращает значение, а сразу же выводит результат на экран. Основная программа должна состоять только из вызова этой функции.

13. Вывести члены последовательности с нечетными номерами Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Выведите первое, третье, пятое и т.д. из введенных чисел. Завершающий ноль выводить не надо. В этой задаче нельзя использовать глобальные переменные и передавать какие-либо параметры в рекурсивную функцию. Функция получает данные, считывая их с клавиатуры. Функция не возвращает значение, а сразу же выводит результат на экран. Основная программа должна состоять только из вызова этой функции.

14. Цифры числа слева направо Дано натуральное число N. Выведите все его цифры по одной, в обычном порядке, разделяя их пробелами или новыми строками. При решении этой задачи нельзя использовать строки, списки, массивы (ну и циклы, разумеется). Разрешена только рекурсия и целочисленная арифметика

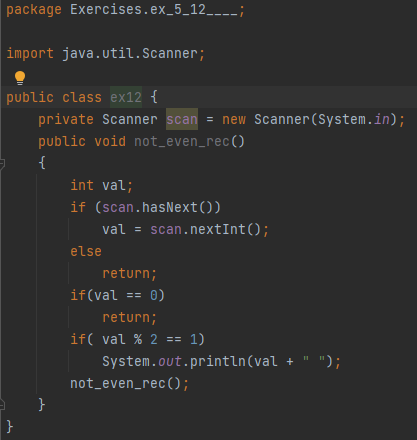
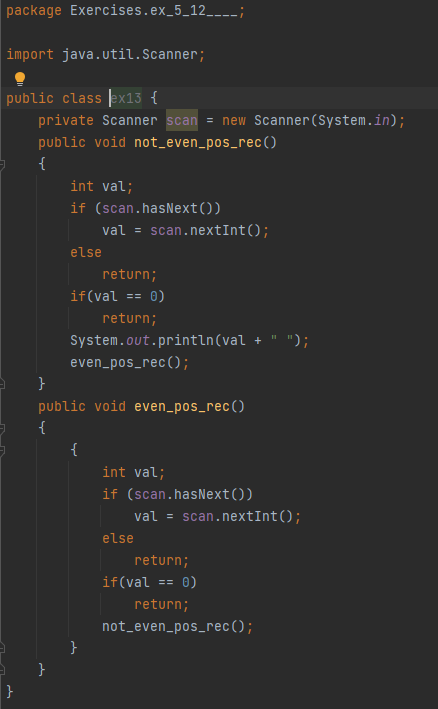
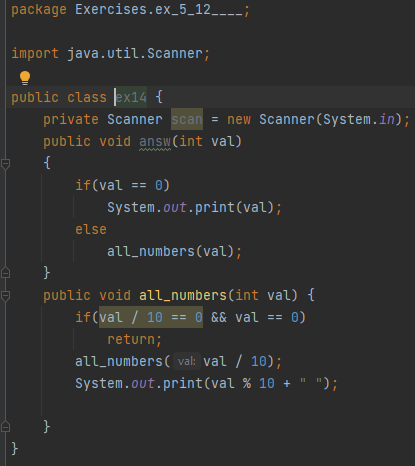
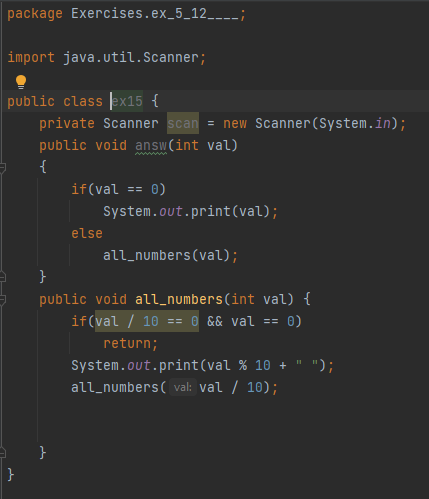
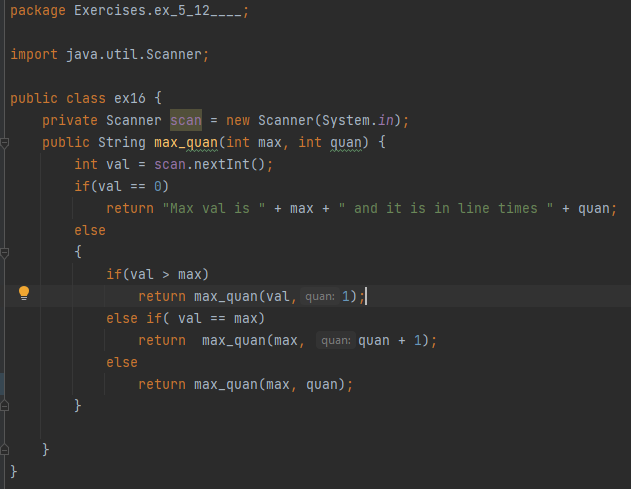
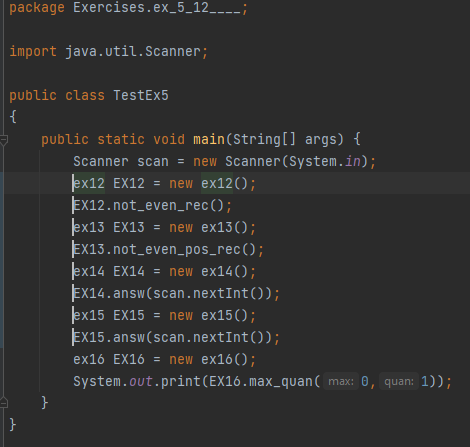
15. Цифры числа справа налево Дано натуральное число N. Выведите все его цифры по одной, в обратном порядке, разделяя их пробелами или новыми строками. При решении этой задачи нельзя использовать строки, списки, массивы (ну и циклы, разумеется). Разрешена только рекурсия и целочисленная арифметика.

16. Количество элементов, равных максимуму Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Определите, какое количество элементов этой последовательности, равны ее наибольшему элементу.

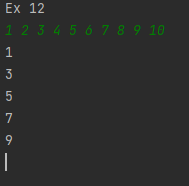
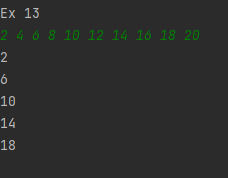
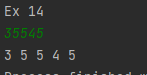
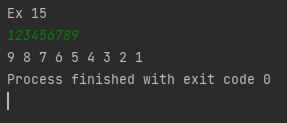
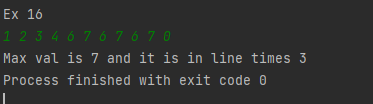
В этой задаче нельзя использовать глобальные переменные. Функция получает данные, считывая их с клавиатуры, а не получая их в виде параметра. В программе на языке Python функция возвращает результат в виде кортежа из нескольких чисел, и функция вообще не получает никаких параметров. В программе на языке C++ результат записывается в переменные, которые передаются в функцию по ссылке. Других параметров, кроме как используемых для возврата значения, функция не получает.

Гарантируется, что последовательность содержит хотя бы одно число (кроме нуля)

## Программный код

## Вывод программы

## Вывод

В результате выполнения …

# ПРАКТИЧЕСКАЯ РАБОТА № №6. ТЕХНИКИ СОРТИРОВКИ В JAVA

## Цель работы

Цель работы: освоение на практике методов сортировки с использованием приемов программирования на объектно-ориентированном языке Java.

## Теоретическое введение

## Постановка задачи

## Программный код

## 

## Вывод программы

## 

## Вывод

В результате выполнения …