# Практическая работа №1

# Тема: «Объектно-ориентированное программирование в Java.»

# Цель практической работы:

Цель данной практической работы – освоить на практике

работу с классами на языке программирования Java.

# Задание:

# Реализуйте простейший класс «Мяч»

# Код программы:

Book.java

**public class** Book {  
 String **name**;  
 String **author**;  
 **int length**;  
  
 **public** Book(String name, String author, **int** length) {  
 **this**.**name** = name;  
 **this**.**author** = author;  
 **this**.**length** = length;  
 }  
 @Override  
 **public** String toString() {  
 **return "Book{"** +  
 **"name='"** + **name** + **'\''** +  
 **", author='"** + **author** + **'\''** +  
 **", length="** + **length** +  
 **'}'**;  
 }  
   
}

Main.java

**public class** Main {  
 **public static void** main(String[] args) {  
 Book book = **new** Book(**"jvav for dumies"**, **"kitty cute"**,100);  
 System.***out***.println(book.toString());  
 }  
}

# Результат работы программы:



**Заключение**

В ходе практической работы я освоил работу с классами в Java на примере класса Book.

# Практическая работа №2

# Тема: «Использование UML-диаграмм при разработке программ»

# Цель практической работы:

Цель данной практической работы – освоить на практике проектирование и документирование кода при помощи UML-диаграмм классов

# Задание:

# По диаграмме класса UML описывающей сущность Автор написать программу которая состоит из двух классов Author и TestAuthor. Класс Author должен содержать реализацию методов, представленных на диаграмме

# Код программы:

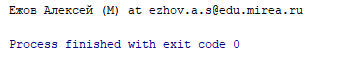
TestAuthor.java

**public class** TestAuthor {  
 **public static void** main(String[] args) {  
 Author author = **new** Author(**"Ежов Алексей"**, **"ezhov.a.s@edu.mirea.ru"**,**'M'**);  
 System.***out***.println(author.toString());  
 }  
}

Author.java

**public class** Author {  
 **private** String **name**;  
 **private** String **email**;  
 **private char gender**;  
  
 **public** Author(String name, String email, **char** gender) {  
 **this**.**name** = name;  
 **this**.**email** = email;  
 **this**.**gender** = gender;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public** String getEmail() {  
 **return email**;  
 }  
  
 **public char** getGender() {  
 **return gender**;  
 }  
  
 **public void** setEmail(String email) {  
 **this**.**email** = email;  
 }  
  
 @Override  
 **public** String toString() {  
 **return name** + **' '** +**"("** + **gender** + **")"**+**" at "** + **email**;  
 }  
}

# **Результат работы программы:**



**Заключение**

В ходе данной практической работы я приобрел навыки разработки и проектирования программы на основе UML-диаграмм классов

# Практическая работа №3

# Тема: «Абстрактные суперклассы и производные классы в Java»

# Цель практической работы:

Цель данной практической работы – освоить на практике работу с абстрактными классами и наследованием на Java

# Задание:

Написать два класса MovablePoint и MovableCircle - которые

реализуют интерфейс Movable.

# Код программы:

“Main.java”

**public class** Main {  
 **public static void** main(String[] args) {  
 MovableCircle circle = **new** MovableCircle(0,0,5,5,10);  
 System.***out***.println(circle.toString());  
 }  
}

“Movable.java”

**public interface** Movable {  
 **void** moveUp();  
 **void** moveDown();  
 **void** moveLeft();  
 **void** moveRight();  
}

“MovablePoint.java”

**public class** MovablePoint **implements** Movable {  
 **int x**;  
 **int y**;  
 **int xSpeed**;  
 **int ySpeed**;  
  
 **public** MovablePoint(**int** x, **int** y, **int** xSpeed, **int** ySpeed) {  
 **this**.**x** = x;  
 **this**.**y** = y;  
 **this**.**xSpeed** = xSpeed;  
 **this**.**ySpeed** = ySpeed;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "MCNVI.MovablePoint{"** +  
 **"x="** + **x** +  
 **", y="** + **y** +  
 **", xSpeed="** + **xSpeed** +  
 **", ySpeed="** + **ySpeed** +  
 **'}'**;  
 }  
  
 @Override  
 **public void** moveUp() {  
 **y**=**y**+**ySpeed**;  
 }  
  
 @Override  
 **public void** moveDown() {  
 **y**=**y**-**ySpeed**;  
 }  
  
 @Override  
 **public void** moveLeft() {  
 **x**=**x**-**xSpeed**;  
 }  
  
 @Override  
 **public void** moveRight() {  
 **x**=**x**+**xSpeed**;  
 }  
}

“MovableCircle.java”

**public class** MovableCircle **implements** Movable {  
 **private int radius**;  
 **private** MovablePoint **center**;  
  
 **public** MovableCircle(**int** x, **int** y, **int** xSpeed, **int** ySpeed, **int** radius) {  
 **this**.**center**=**new** MovablePoint(x,y,xSpeed,ySpeed);  
 **this**.**radius**=radius;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "MovableCircle{"** +  
 **"radius="** + **radius** +  
 **", center="** + **center** +  
 **'}'**;  
 }  
  
 @Override  
 **public void** moveUp() {  
 **center**.**y**=**center**.**y**+**center**.**ySpeed**;  
 }  
  
 @Override  
 **public void** moveDown() {  
 **center**.**y**=**center**.**y**-**center**.**ySpeed**;  
 }  
  
 @Override  
 **public void** moveLeft() {  
 **center**.**x**=**center**.**x**-**center**.**xSpeed**;  
 }  
  
 @Override  
 **public void** moveRight() {  
 **center**.**x**=**center**.**x**+**center**.**xSpeed**;  
 }  
}

# Результат работы программы:



**Заключение**

В ходе практической работы я научился работать с абстрактными классами и наследованием в Java

# Практическая работа №4

# Тема: «Событийное программирование.»

# Цель практической работы:

Цель данной практической работы – введение в событийное программирование и программирование графического интерфейса пользователя в языке программирования Java.

# Задание:

Создать графический интерфейс пользователя на основе библиотеки Java Swing, добавить две кнопки JButton и три текстовых блока JLabel, сконфигурировать слушатель для кнопок.

# Код программы:

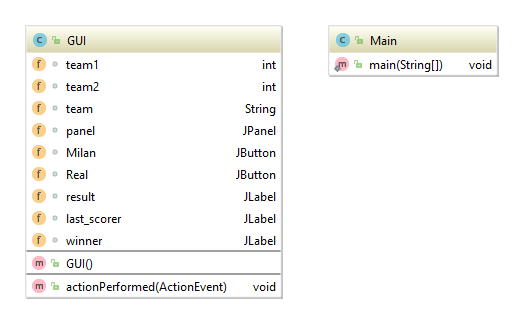
“Main.java”

**public class** Main {  
 **public static void** main(String[] args) {  
 GUI example = **new** GUI();  
 example.setVisible(**true**);  
 example.setSize(320,240);  
 example.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);  
 example.setResizable(**false**);  
 }  
}

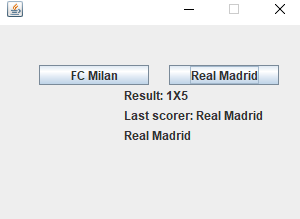
“GUI.java”

**public class** GUI **extends** JFrame **implements** ActionListener {  
 **int team1** = 0;  
 **int team2** = 0;  
 String **team** = **"N/A"**;  
 JPanel **panel** = **new** JPanel(**null**);  
 JButton **Milan** = **new** JButton(**"FC Milan"**);  
 JButton **Real** = **new** JButton(**"Real Madrid"**);  
 JLabel **result** = **new** JLabel(**"Result: "** + **team1** + **"X"** + **team2**);  
 JLabel **last\_scorer** = **new** JLabel(**"Last scorer: "** + **team**);  
 JLabel **winner** = **new** JLabel(**"DRAW"**);  
  
 **public** GUI(){  
 **this**.setContentPane(**panel**);  
 **Milan**.setBounds(40,40,110,20);  
 **Milan**.addActionListener(**this**);  
 **Real**.setBounds(170,40,110,20);  
 **Real**.addActionListener(**this**);  
 **result**.setBounds(125,60,80,20);  
 **last\_scorer**.setBounds(125,80,140,20);  
 **winner**.setBounds(125,100,70,20);  
 **panel**.add(**Milan**);  
 **panel**.add(**Real**);  
 **panel**.add(**result**);  
 **panel**.add(**last\_scorer**);  
 **panel**.add(**winner**);  
 }  
  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 String command = e.getActionCommand();  
 **if**(command.equals(**"FC Milan"**)){  
 **team** = **"FC Milan"**;  
 **team1**++;  
 **if**(**team1**>**team2**){  
 **winner**.setText(**"FC Milan"**);  
 }  
 **else if**(**team1**==**team2**){  
 **winner**.setText(**"DRAW"**);  
 }  
 **result**.setText(**"Result: "** + **team1** + **"X"** + **team2**);  
 **last\_scorer**.setText(**"Last scorer: "** + **team**);  
 }  
 **else if**(command.equals(**"Real Madrid"**)){  
 **team** = **"Real Madrid"**;  
 **team2**++;  
 **if**(**team2**>**team1**){  
 **winner**.setText(**"Real Madrid"**);  
 }  
 **else if**(**team1**==**team2**){  
 **winner**.setText(**"DRAW"**);  
 }  
 **result**.setText(**"Result: "** + **team1** + **"X"** + **team2**);  
 **last\_scorer**.setText(**"Last scorer: "** + **team**);  
 }  
 }  
}

# UML диаграмма классов:



# Результат работы программы:



**Заключение**

В ходе практической работы я изучил событийное программирование и программирование графического интерфейса пользователя.

# Практическая работа №5

# Тема: «Рекурсия и её использование.»

# Цель практической работы:

Цель данной практической работы – изучение на практике применения рекурсивных алгоритмов и разработка рекурсивных функций.

# Задание:

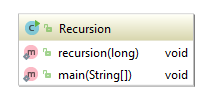
Разработать программное обеспечение, которое позволяет выводить на экран цифры произвольного числа в обратном порядке, используя рекурсивные алгоритмы.

# Код программы:

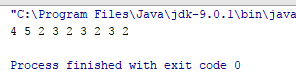
“Recursion.java”

**public class** Recursion {  
 **public static void** recursion(**long** n){  
 **if**(n==0){  
 System.***out***.println(**" "**);  
 }  
 **else**{  
 System.***out***.print(n%10 + **" "**);  
 n=n/10;  
 *recursion*(n);  
 }  
 }  
  
 **public static void** main(String[] args) {  
 **long** N = 232323254;  
 *recursion*(N);  
 }  
}

# UML диаграмма класса:



# Результат работы программы:



**Заключение**

В ходе практической работы я научился применять рекурсивные алгоритмы при разработке программных продуктов на Java.

# Практическая работа №6

# Тема: «Техники сортировки.»

# Цель практической работы:

Цель данной практической работы – освоить на практике различные методы сортировки массивов.

# Задание:

Реализовать программу на Java, включающую в себя класс SortingStudentByGPA, который реализует интерфейс Comparator таким образом, что она сортирует студентов по их итоговому баллу в порядке убывания.

# Код программы:

“SortingStudentsByGPA.java”

**import** java.util.Comparator;  
  
**public class** SortingStudentsByGPA **implements** Comparator {  
 @Override  
 **public int** compare(Object o1, Object o2) {  
 **if**(((Student)o1).getGPA()>((Student)o2).getGPA()){  
 **return** -1;  
 }  
 **else return** 1;  
 }  
}

“Student.java”

**public class** Student {  
 String **name**;  
 String **lastname**;  
 **int GPA**;  
  
 **public** Student(String name, String lastname, **int** GPA) {  
 **this**.**name** = name;  
 **this**.**lastname** = lastname;  
 **this**.**GPA** = GPA;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public** String getLastname() {  
 **return lastname**;  
 }  
  
 **public int** getGPA() {  
 **return GPA**;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Student{"** +  
 **"name='"** + **name** + **'\''** +  
 **", lastname='"** + **lastname** + **'\''** +  
 **", GPA="** + **GPA** +  
 **'}'**;  
 }  
}

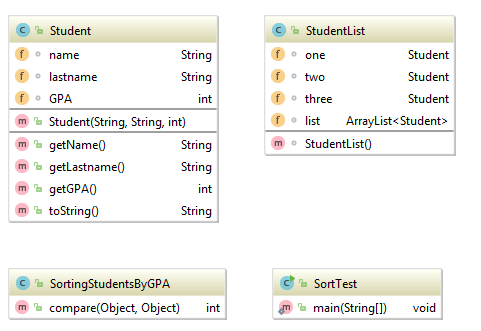
“SortTest.java”

**import** java.util.Collections;  
  
**public class** SortTest {  
  
 **public static void** main(String[] args) {  
 StudentList example = **new** StudentList();  
 **for**(Student student:example.**list**){  
 System.***out***.println(student.toString());  
 }  
 Collections.*sort*(example.**list**, **new** SortingStudentsByGPA());  
 System.***out***.println(**"После сортировки:"**);  
 **for**(Student student:example.**list**){  
 System.***out***.println(student.toString());  
 }  
 }  
}

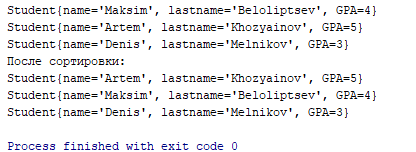
“StudentList.java”

**import** java.util.ArrayList;  
  
**public class** StudentList {  
 Student **one** = **new** Student(**"Maksim"**, **"Beloliptsev"**, 4);  
 Student **two** = **new** Student(**"Artem"**,**"Khozyainov"**, 5);  
 Student **three** = **new** Student(**"Denis"**, **"Melnikov"**, 3);  
 ArrayList<Student> **list** = **new** ArrayList<>();  
 StudentList(){  
 **list**.add(**one**);  
 **list**.add(**two**);  
 **list**.add(**three**);  
 }  
}

# UML диаграмма классов:



# Результат работы программы:



**Заключение**

В ходе данной практической работы я научился применять различные алгоритмы сортировки массивов при разработке программ на Java.

# Практическая работа №7

# Тема: «Коллекции, очереди, списки в Java.»

# Цель практической работы:

Целью данной практической работы является изучение разработки программ с использованием динамических структур данных на примере абстрактных типов данных.

# Задание:

Написать карточную игру «Пьяницa»

# Код программы:

“Main.java”

**public class** Main {  
  
  
 **public static void** main(String[] args) {  
 Game game = **new** Game();  
 System.***out***.println(game.toString());  
 game.gameplay();  
 System.***out***.println(game.toString());  
 **if**(game.**playerOneWins**)  
 System.***out***.println(**"first "** + game.**n**);  
 **else if**(game.**playerTwoWins**)  
 System.***out***.println(**"second "** + game.**n**);  
 **else** System.***out***.println(**"botva"**);  
 }  
}

“Deck.java”

**import** java.util.ArrayList;  
**import** java.util.Comparator;  
  
**public class** Deck {  
 ArrayList<Card> **list** = **new** ArrayList<>();  
 Deck(){  
 **int** i=0;  
 **while** (i<10){  
 Card card = **new** Card(i);  
 **list**.add(card);  
 i++;  
 }  
 }  
 **public void** ShuffleDeck(){  
 **list**.sort(**new** Comparator<Card>() {  
 @Override  
 **public int** compare(Card o1, Card o2) {  
 **int** a = (**int**)(Math.*random*()\*10);  
 **if**(a<5){  
 **return** 1;  
 }  
 **else return** -1;  
 }  
 });  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Deck{"** +  
 **"list="** + **list** +  
 **'}'**;  
 }  
}

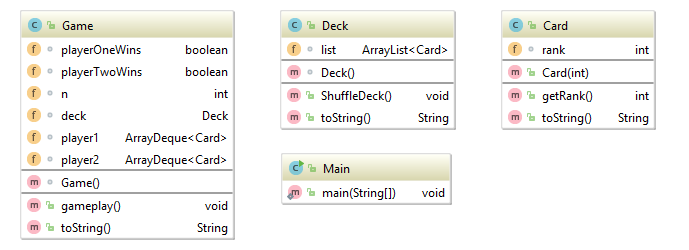
“Card.java”

**public class** Card {  
 **int rank**;  
 **public** Card(**int** rang) {  
 **this**.**rank** = rang;  
 }  
 **public int** getRank() {  
 **return rank**;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Card{"** +  
 **"rank="** + **rank** +  
 **'}'**;  
 }  
}

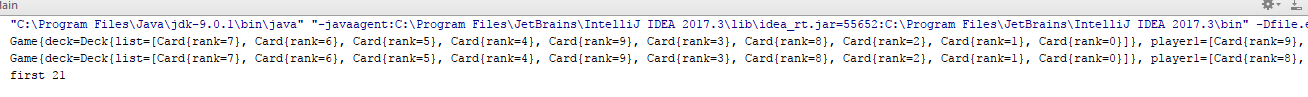
“Game.java”

**import** java.util.ArrayDeque;  
**import** java.util.Stack;  
  
**public class** Game {  
 **boolean playerOneWins** = **false**;  
 **boolean playerTwoWins** = **false**;  
 **int n** = 0;  
 Deck **deck** = **new** Deck();  
 ArrayDeque<Card> **player1** = **new** ArrayDeque<>();  
 ArrayDeque<Card> **player2** = **new** ArrayDeque<>();  
 Game(){  
 **int** i = 0;  
 **deck**.ShuffleDeck();  
 **while**(i<5){  
 **player1**.push(**deck**.**list**.get(i));  
 **player2**.push(**deck**.**list**.get(i+5));  
 i++;  
 }  
  
 }  
 **public void** gameplay() {  
 **while** (!(**playerOneWins** || **playerTwoWins** || (**n** > 106))) {  
 **n**++;  
 **if** (**player1**.getFirst().getRank() > **player2**.getFirst().getRank()) {  
 **player1**.addLast(**player1**.getFirst());  
 **player1**.removeFirst();  
 **player1**.addLast(**player2**.getFirst());  
 **player2**.removeFirst();  
 **if** (**player2**.isEmpty())  
 **playerOneWins** = **true**;  
 }  
 **else** {  
 **player2**.addLast(**player2**.getFirst());  
 **player2**.removeFirst();  
 **player2**.addLast(**player1**.getFirst());  
 **player1**.removeFirst();  
 **if** (**player1**.isEmpty())  
 **playerTwoWins** = **true**;  
 }  
 }  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Game{"** +  
 **"deck="** + **deck** +  
 **", player1="** + **player1** +  
 **", player2="** + **player2** +  
 **'}'**;  
 }  
}

# UML диаграмма классов:



# Результат работы программы:



**Заключение**

В ходе практической работы я изучил работу с динамическими структурами данных на примере разработки карточной игры «Пьяница»