|  |  |
| --- | --- |
|  | |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  "МИРЭА - Российский технологический университет"  **РТУ** **МИРЭА** | |
| Институт информационных технологий (ИИТ)  Кафедра информационного и прикладного программного обеспечения (ИППО) | |
|  | |

|  |  |
| --- | --- |
| **Отчет по практическим работам** | |
| **По дисциплине «Объектно-ориентированное программирование»** | |
| Выполнил студент группы ИКБО-13-17 | Хитров Н.О. |
| |  |  |  | | --- | --- | --- | | Отчет выполнен | «\_\_»\_\_\_\_\_\_\_201\_\_ г. |  | |  |  |  | | «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. |  |   Принял \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Хлебникова В. Л. |

Москва 2018

# Таблица успеваемости

**Лабораторные занятия**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №/р. | страница в отчете. | Студент выполнил:  (подпись) | Преподаватель принял:  (подпись) | Балл  (максимально возможный) | Балл (Фактический) |
| 1. |  | *.* | *.* | 2 | … |
| 2. |  |  |  | 2 |  |
| 3. |  |  |  | 4 |  |
| 4. |  |  |  | 6 |  |
| 5. |  |  |  | 6 |  |
| 6. |  |  |  | 8 |  |
| 7. |  |  |  | 8 |  |
|  |  |  | Итоговый балл: | 36 |  |

Оглавление

[Таблица успеваемости 2](#_Toc531627255)

[Практическая работа №1 4](#_Toc531627256)

[Практическая работа №2 6](#_Toc531627257)

[Практическая работа №3 8](#_Toc531627258)

[Практическая работа №4 11](#_Toc531627259)

[Практическая работа №5 14](#_Toc531627260)

[Практическая работа №6 15](#_Toc531627261)

[Практическая работа №7 17](#_Toc531627262)

# Практическая работа №1

**Цель практической работы**: освоить на практике работу с классами на Java.

**Задание:**

Реализуйте простейший класс «Книга»

**Ход выполнения работы:**

Программа содержит класс Book, который содержит в себе комплект конструкторов, а также сеттеров и геттеров.

|  |
| --- |
| package ru.mirea.khitrov.practice.practice1.task2;  public class Book {  private String title, author;  private int pagesCount;  public Book(String title, String author, int pagesCount) {  this.title = title;  this.author = author;  this.pagesCount = pagesCount;  }  public String getTitle() {  return title;  }  public void setTitle(String title) {  this.title = title;  }  public String getAuthor() {  return author;  }  public void setAuthor(String author) {  this.author = author;  }  public int getPagesCount() {  return pagesCount;  }  public void setPagesCount(int pagesCount) {  this.pagesCount = pagesCount;  }  @Override  public String toString() {  return "Book{" +  "title='" + title + '\n'' +  ", author='" + author + '\n'' +  ", pages=" + pagesCount +  '}';  }  } |

Обращение к этому классу происходит из Main класса:

|  |
| --- |
| package ru.mirea.khitrov.practice.practice1.task2;  public class Main {  public static void main(String[] args) {  Book b = new Book("Java Cookbook", "Ian F. Darwin", 864);  System.out.println(b);  }  } |

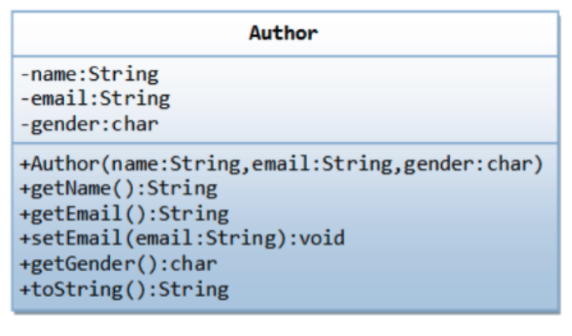
Вывод данной программы выглядит следующим образом:

|  |
| --- |
| Book{title='Java Cookbook', author='Ian F. Darwin', pages=864} |

**Вывод:** в результате выполнения данной практической работы были получены навыки по работе с классами в Java.

# Практическая работа №2

**Цель практической работы**: Изучить основные концепции работы с UML-диаграммами классов.

**Задание**: По диаграмме класса UML описывающей сущность Автор написать программу, которая состоит из класса Author и основного класса программы. Класс Author должен содержать реализацию методов, представленных на диаграмме.

Риснок 1 – UML-диграмма класса.

**Ход выполнения работы:**

Для выполнения данной практической работы было создан класс Author, в котором были реализованы методы, представленные в диаграмме.

|  |
| --- |
| package ru.mirea.khitrov.practice.practice2.task1;  public class Author {  private String name, 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 char getGender() {  return gender;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  @Override  public String toString() {  return name + " (" + gender + ") at " + email;  }  } |

Создание объекта класса происходит в основном классе программы:

|  |
| --- |
| package ru.mirea.khitrov.practice.practice2.task1;  public class TestAuthor {  public static void main(String[] args) {  Author a = new Author("Rob Pike", "pike.rob@gmail.com", 'm');  System.out.println(a);  }  } |

Вывод работы программы выглядит следующим образом:

|  |
| --- |
| Rob Pike (m) at pike.rob@gmail.com |

**Вывод:** в ходе выполнения данной практической работы были изучены основные концепции работы с UML-диаграммами классов.

# Практическая работа №3

**Цель практической работы**: Освоить на практике работу с абстрактными классами.

З**адание**: Вам нужно написать два класса MovablePoint и MovableCircle - которые реализуют интерфейс Movable . Напишите новый класс MovableRectangle (движущийся прямоугольник). Его можно представить как две движущиеся точки MovablePoints (представляющих верхняя левая и нижняя правая точки) и реализующие интерфейс Movable. Убедитесь, что две точки имеет одну и ту же скорость (нужен метод это проверяющий).

**Ход выполнения работы:**

В ходе выполнения данной лабораторной было создано три класса и один интерфейс.

Интерфейс Moveable, c первоначальным комплектом методов.

|  |
| --- |
| package ru.mirea.khitrov.practice.practice3.task2;  public interface Movable {  void moveUp();  void moveDown();  void moveLeft();  void moveRight();  } |

Также, были созданы три класса.

Класс MovableCircle:

|  |
| --- |
| package ru.mirea.khitrov.practice.practice3.task2;  public class MovableCircle implements Movable {  private int radius;  private MovablePoint center;  public MovableCircle(int x, int y, int xSpeed, int ySpeed, int radius) {  center = new MovablePoint(x, y, xSpeed, ySpeed);  this.radius = radius;  }  @Override  public void moveUp() {  center.moveUp();  }  @Override  public void moveDown() {  center.moveDown();  }  @Override  public void moveLeft() {  center.moveLeft();  }  @Override  public void moveRight() {  center.moveRight();  }  @Override  public String toString() {  return "circle(" +  "r=" + radius +  ", c=" + center.toString() +  ')';  }  } |

Класс MovablePoint

|  |
| --- |
| package ru.mirea.khitrov.practice.practice3.task2;  public class MovablePoint implements Movable {  int x, y, xSpeed, 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 void moveUp() {  y += ySpeed;  }  @Override  public void moveDown() {  y -= ySpeed;  }  @Override  public void moveLeft() {  x -= xSpeed;  }  @Override  public void moveRight() {  x += xSpeed;  }  @Override  public String toString() {  return "(" +  "" + x +  ", " + y +  ')';  }  } |

Класс MovableRectangle

|  |
| --- |
| package ru.mirea.khitrov.practice.practice3.task2;  public class MovableRectangle implements Movable {  private MovablePoint topLeft, bottomRight;  public MovableRectangle(int x1, int y1, int x2, int y2, int xSpeed, int ySpeed) {  topLeft = new MovablePoint(x1, y1, xSpeed, ySpeed);  bottomRight = new MovablePoint(x2, y2, xSpeed, ySpeed);  }  @Override  public void moveUp() {  topLeft.moveUp();  bottomRight.moveUp();  }  @Override  public void moveDown() {  topLeft.moveDown();  bottomRight.moveDown();  }  @Override  public void moveLeft() {  topLeft.moveLeft();  bottomRight.moveLeft();  }  @Override  public void moveRight() {  topLeft.moveRight();  bottomRight.moveRight();  }  @Override  public String toString() {  return "MovableRectangle{" +  "p1=" + topLeft.toString() +  ", p2=" + bottomRight.toString() +  '}';  }  } |

Вызов методов данной семьи классов происходит из функции main основного класса программы:

|  |
| --- |
| package ru.mirea.khitrov.practice.practice3.task2;  public class MovableTest {  public static void main(String[] args) {  Movable circle = new MovableCircle(0, 0, 1, 1, 2);  circle.moveLeft();  System.out.println(circle.toString());  circle.moveDown();  System.out.println(circle.toString());  Movable rectangle = new MovableRectangle(0, 4, 4, 0, 2, 2);  rectangle.moveRight();  System.out.println(rectangle.toString());  rectangle.moveUp();  System.out.println(rectangle.toString());  }  } |

**Вывод:** в ходе выполнения данной практической работы были получены навыки работы с абстрактными классами.

# Практическая работа №4

**Цель данной практической работы**: Введение в событийное программирование

**Задание**: Создайте JFrame приложение у которо 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”;

Теперь всякий раз, когда вы нажимаете на кнопку AC Milan, результат будет увеличиваться для Милана, чтобы стать сначала 1 X 0, затем 2 X 0. Last Scorer означает последнюю забившую команду. В этом случае: AC Milan. И победителем становится команда, которая имеет больше кликов кнопку на соответствующую, чем другая.

**Ход выполнения работы:**

В ходе выполнения работы был создан класс основного окна:

|  |
| --- |
| package ru.mirea.khitrov.practice.practice4.task1;  import javax.swing.\*;  import java.awt.\*;  public class GuiExample extends JFrame {  int milanScore = 0, madridScore = 0;  String milanName = "AC Milan", madridName = "Real Madrid";  JLabel resultLabel = new JLabel("Result: 0 X 0"),  lastScorerLabel = new JLabel("Last Scorer: N/A"),  winnerLabel = new JLabel("Winner: DRAW");  Button milanButton = new Button(milanName),  madridButton = new Button(madridName);  public static void main(String[] args) {  new GuiExample();  }  public GuiExample() {  super("Gui Example");  setSize(500, 200);  setLayout(new GridLayout(3, 2));  setResizable(false);  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  milanScore = madridScore = 0;  milanName = "AC Milan";  madridName = "Real Madrid";  resultLabel = new JLabel("Result: 0 X 0");  lastScorerLabel = new JLabel("Last Scorer: N/A");  winnerLabel = new JLabel("Winner: DRAW");  milanButton = new Button(milanName);  madridButton = new Button(madridName);  milanButton.addActionListener(e -> {  milanScore++;  updateLabels(milanName);  });  madridButton.addActionListener(e -> {  madridScore++;  updateLabels(madridName);  });  add(milanButton);  add(madridButton);  add(resultLabel);  add(lastScorerLabel);  add(winnerLabel);  setVisible(true);  }  public void updateLabels(String lastScorer) {  resultLabel.setText(String.format("Result: %d X %d", milanScore, madridScore));  lastScorerLabel.setText(String.format("Last Scorer: %s", lastScorer));  String winnerName = milanScore < madridScore ? madridName : milanScore > madridScore ? milanName : "DRAW";  winnerLabel.setText("Winner: " + winnerName);  }  } |

Взаимодействие с классом происходит внутри метода main основного класса программы практической работы:

Вывод работы данной программы представляет собой окно с GUI.

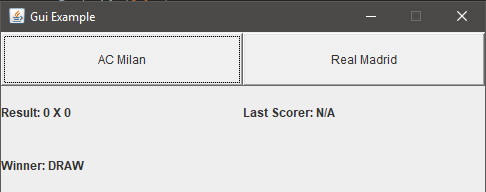


Рисунок 2 – Окно программы до взаимодействия с пользователем.

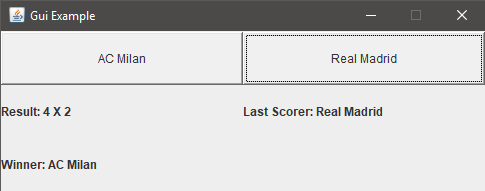


Рисунок 3– Окно программы после взаимодействия с пользователем

**Вывод:** в ходе выполнения данной практической работы были освоены начальные принципы событийного программирования.

# Практическая работа №5

**Цель данной практическая работы:** Изучение работы с рекурсией.

**Задание:** Дано натуральное число n. Выведите все числа от 1 до n.

**Ход выполнения работы:**

Для выполнения данной практической работы был использован только один класс.

|  |
| --- |
| package ru.mirea.khitrov.practice.practice5.task2;  public class Main {  public static void main(String[] args) {  printToN(10);  }  public static void printToN(int n) {  printToN(n, n);  }  public static void printToN(int startN, int currentN) {  System.out.println(startN - currentN + 1);  if (currentN - 1 == 0) {  return;  }  printToN(startN, currentN-1);  }  } |

Вывод работы данной программы выглядит следующим образом:

|  |
| --- |
| 1  2  3  4  5  6  7  8  9  10 |

**Вывод:** в ходе выполнения данной работы были получены навыки работы с рекурсией.

# Практическая работа №6

**Цель данной практическая работы:** Освоить на практике сортировки различными методами.

**Задание**: Написать тестовый класс, который создает массив класса Student и сортирует массив iDNumber.

**Ход выполнения работы:**

Для выполнения данной работы был создан класс Student:

|  |
| --- |
| package ru.mirea.khitrov.practice.practice6.task3;  import java.util.ArrayList;  import java.util.Comparator;  import java.util.Random;  public class Student {  private static int studentsCount = 0;  private int id;  private String name;  public Student() {  id = Student.studentsCount++;  name = "Student" + id;  }  public Student(int id, String name) {  id = Student.studentsCount++;  this.name = name;  }  public int getId() {  return id;  }  public String getName() {  return name;  }  static public int compare(Student first, Student second) {  if (first.getId() < second.getId()) {  return 1;  } else if (first.getId() == second.getId()) {  return 0;  } else {  return -1;  }  }  @Override  public String toString() {  return "Student: #" + id + ", name=" + name;  }  } |

Класс Student вызывается командами основного класса.

|  |
| --- |
| package ru.mirea.khitrov.practice.practice6.task3;  import java.util.\*;  public class Main {  public static void main(String[] args) {  ArrayList<Student> students = new ArrayList<>();  for (int i = 0; i < 5; i++) {  students.add(new Student());  }  Collections.reverse(students);  System.out.println("Generated:");  for (int i = 0; i < 5; i++) {  System.out.println(students.get(i));  }  System.out.println("===============================\nSorted:");  students.sort(Comparator.comparingInt(Student::getId));  for (Student s : students) {  System.out.println(s);  }  }  } |

Результатом работы данной программы будет следующий вывод:

|  |
| --- |
| Generated:  Student: #4, name=Student4  Student: #3, name=Student3  Student: #2, name=Student2  Student: #1, name=Student1  Student: #0, name=Student0  ===============================  Sorted:  Student: #0, name=Student0  Student: #1, name=Student1  Student: #2, name=Student2  Student: #3, name=Student3  Student: #4, name=Student4 |

**Вывод:** в ходе выполнения данной практической работы были освоены на практике сортировки различными методами.

# Практическая работа №7

**Цель данной практическая работы:** Изучение работы с очередями.

**Задание**: Смоделировать игру. Правила игры: В игре в пьяницу карточная колода раздается поровну двум игрокам. Далее они вскрывают по одной верхней карте, и тот, чья карта старше, забирает себе обе вскрытые карты, которые кладутся под низ его колоды. Тот, кто остается без карт - проигрывает. Для простоты будем считать, что все карты различны по номиналу, а также, что самая младшая карта побеждает самую старшую карту ("шестерка берет туза"). Игрок, который забирает себе карты, сначала кладет под низ своей колоды карту первого игрока, затем карту второго игрока (то есть карта второго игрока оказывается внизу колоды). В игре участвует 10 карт, имеющих значения от 0 до 9, большая карта побеждает меньшую, карта со значением 0 побеждает карту 9.

**Ход выполнения работы:**

Для выполнения задания были спроектированы 2 класса: Card и Player. Класс Card представляет собой сущность карты и логику для проверки старшинства карты:

|  |
| --- |
| package ru.mirea.khitrov.practice.practice7.task1;  public class Card {  private int value;  public Card(int value) {  this.value = value;  }  public int getValue() {  return value;  }  public boolean canHit(Card c) {  if (value == 0 && c.value == 9) {  return true;  }  return value > c.value;  }  } |

Класс Player представляет собой сущность игрока:

|  |
| --- |
| package ru.mirea.khitrov.practice.practice7.task1;  import java.util.ArrayDeque;  import java.util.ArrayList;  public class Player {  private ArrayDeque<Card> cards;  private boolean lostGame = false;  public Player(String cardsLine) {  ArrayList<Card> cards = new ArrayList<>();  for (String s: cardsLine.split(" ")) {  cards.add(new Card(Integer.parseInt(s)));  }  this.cards = new ArrayDeque<>();  this.cards.addAll(cards);  }  public Card showCard() {  if (cards.size() == 1) {  lostGame = true;  }  return cards.pop();  }  public void getNewCards(Card c1, Card c2) {  cards.add(c1);  cards.add(c2);  }  public boolean lostGame() {  return this.lostGame;  }  } |

Процесс игры начинается в основном файле программы:

|  |
| --- |
| package ru.mirea.khitrov.practice.practice7.task1;  import java.io.BufferedReader;  import java.io.IOException;  import java.io.InputStreamReader;  public class Main {  public static void main(String[] args) throws IOException {  BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));  Player player1 = new Player(reader.readLine()), player2 = new Player(reader.readLine());  int count = 0;  while (!player1.lostGame() && !player2.lostGame()) {  Card c1 = player1.showCard(), c2 = player2.showCard();  if (c1.canHit(c2)) {  player1.getNewCards(c1, c2);  } else {  player2.getNewCards(c1, c2);  }  count++;  if (count == 106) {  System.out.println("botva");  return;  }  }  if (!player1.lostGame()) {  System.out.println("first " + count);  } else {  System.out.println("second " + count);  }  }  } |

**Вывод:** в ходе выполнения данной практической работы были получены навыки работы с очередями в Java.

**Библиографический список**

1. Роберт Лафоре, Структуры данных и алгоритмы в Java
2. [В. Белов](http://www.ozon.ru/person/4530084/), [В. Чистякова](http://www.ozon.ru/person/4530088/) Алгоритмы и структуры данных. Учебник[Инфра-м](http://www.chaconne.ru/izdat.php?id=14253) 2016  240 стр. ISBN: 5906818256
3. Богачев К.Ю. Основы параллельного программирования– М.: Бином. Лаборатория знаний, 2003. -.342 с.
4. Зыль С.Н. Операционная система реального времени QNX Neutrino: от теории кпрактике – Изд. 2-е - СПб.: БХВ-Петербург, 2004. – 192 с.