|  |
| --- |
| 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 №1-24**

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

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил:  Студент группы ИКБО-16-22 | « » 2023 г. | (подпись) | Шкарупа Д. Е. |
|  |  |  |  |
| Принял:  Старший преподаватель кафедры ИиППО ИИТ | « » 2023 г. | (подпись) | Матчин В.Т. |

Москва 2023 г.

Оглавление

# Практическая работа № 1. Знакомство со средой разработки. Синтаксис и основные управляющие конструкции языка Джава

Цель: введение в разработку программ на языке программирования Джава.

1. Задание 1

Написать программу, в результате которой массив чисел создается с помощью инициализации (как в Си) вводится и считается в цикле сумма элементов целочисленного массива, а также среднее арифметическое его элементов результат выводится на экран. Использовать цикл for.

Листинг 1 – решение задания №1

1. Scanner sc = **new** Scanner(System.in);
2. System.out.println("Задание 1");
3. **int** [] arr = **new** **int** [10];
4. **int** summ = 0;
5. **for** (**int** i = 0; i < arr.length; i++)
6. {
7. System.out.print("Введите " + (i + 1) + " элемент массива: ");
8. arr[i] = sc.nextInt();
9. summ += arr[i];
11. }
12. System.out.println("Сумма элементов массива: " + summ);
13. System.out.println("Среднее арифметическое элементов массива " + (**float**)summ / arr.length);
14. Задание 2

Написать программу, в результате которой массив чисел вводится пользователем с клавиатуры считается сумма элементов целочисленного массива с помощью циклов do while, while, также необходимо найти максимальный и минимальный элемент в массиве, результат выводится на экран.

Листинг 2 – решение задания №2

1. System.out.println("Задание 2");
2. **int** [] arr2 = **new** **int**[5];
3. **int** summ2 = 0, min\_el = 1000000000, max\_el = 0;
4. **int** i = 0;
5. **while**(i < arr2.length)
6. {
7. System.out.print("Введите " + (i + 1) + " элемент массива: ");
8. arr2[i] = sc.nextInt();
9. summ2 += arr2[i];
10. **if** (arr2[i] < min\_el){ min\_el = arr2[i];}
11. **if** (arr2[i] > max\_el){ max\_el = arr2[i];}
12. i++;
13. }
14. System.out.println("Сумма элементов массива " + summ2);
15. System.out.println("Максимальный элемент массива " + max\_el);
16. System.out.println("Минимальный элемент массива " + min\_el);
18. System.out.println("Задание 2.2");
19. **int** [] arr3 = **new** **int**[5];
20. **int** summ3 = 0, min\_el\_ = 100000000, max\_el\_ = 0;
21. i = 0;
22. **do**
23. {
24. System.out.print("Введите " + (i + 1) + " элемент массива: ");
25. arr3[i] = sc.nextInt();
26. summ3 += arr3[i];
27. **if** (arr3[i] < min\_el\_){ min\_el\_ = arr3[i];}
28. **if** (arr3[i] > max\_el\_){ max\_el\_ = arr3[i];}
29. i++;
30. } **while**(i < arr3.length) ;
32. System.out.println("Сумма элементов массива " + summ3);
33. System.out.println("Максимальный элемент массива " + max\_el\_);
34. System.out.println("Минимальный элемент массива " + min\_el\_);
35. Задание 3

Написать программу, в результате которой выводятся на экран аргументы командной строки в цикле for.

Листинг 3 – решение задания №3

1. System.out.println("Форматированный вывод: ");
2. **for** (i = 0; i < 10; i++)
3. {
4. System.out.print((i + 1) + " ");
5. }
6. Задание 4

Написать программу, в результате работы которой выводятся на экран первые 10 чисел гармонического ряда (форматировать вывод).

Листинг 4 – решение задания №4

1. System.out.println("\nЗадание 4");
2. System.out.println("\nГармонический ряд");
3. **for** (i = 0; i < 10; i++)
4. {
5. **if** (i == 0) {
6. System.out.print("1" + "  ");
7. }
8. System.out.print("1" + "/" + (i+2) + "  ");
9. }
10. Задание 5

Написать программу, которая с помощью метода класса, вычисляет факториал числа (использовать управляющую конструкцию цикла), проверить работу метода.

Листинг 5 – решение задания №5

1. **public** **class** math\_func {
2. **public** **int** Factorial(**int** n)
3. {
4. **int** answer = 1;
5. **for** (**int** i = 1; i <= n; i++) {answer \*= i;}
6. **return** answer;
7. }
8. }
9. System.out.print("Введите число: ");
10. **int** n = sc.nextInt();
11. math\_func fact = **new** math\_func();
12. System.out.println("Факториал числа: " + fact.Factorial(n));

# Практическая работа № 2. Объектно-ориентированное программирование в Джава. Классы в Джава

Цель: изучить основные концепции объектно-ориентированного программирования, изучить понятие класса и научиться создавать классы.

1. Задание 1

По диаграмме класса UML описывающей сущность Автор. Необходимо написать программу, которая состоит из двух классов Author и TestAuthor. Класс Author должен содержать реализацию методов, представленных на диаграмме класса на рисунке 1.

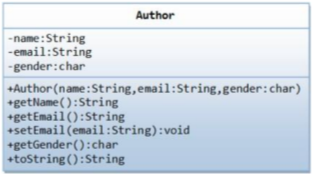


Рисунок 1 – Диаграмма класса Author

Листинг 6 – решение задания №1 файл Author.java

1. **public** **class** Author {
2. **private** String name;
3. **private** String email;
4. **private** **char** gender;
6. **public** Author(String name, String email, **char** gender)
7. {
8. **this**.name = name;
9. **this**.email = email;
10. **this**.gender = gender;
11. }
13. **public** String getName(){**return** name;}
14. **public** String getEmail(){**return** email;}
15. **public** **void** setEmail(String new\_email){email = new\_email;}
16. **public** **char** getGender(){**return** gender;}
18. @Override
19. **public** String toString() {
20. **return** "Author{" +
21. "name='" + name + '\'' +
22. ", email='" + email + '\'' +
23. ", gender=" + gender +
24. '}';
25. }
26. }

Листинг 7 – решение задания №1 файл TestAuthor.java

1. **import** org.w3c.dom.ls.LSOutput;
3. **import** java.util.Scanner;
5. **public** **class** TestAuthor {
6. **public** **static** **void** main(String[] args) {

9. Scanner sc = **new** Scanner(System.in);
10. System.out.print("Введите имя пользователя: ");
11. String user\_name = sc.nextLine();
12. System.out.print("Введите email пользователя: ");
13. String user\_email = sc.nextLine();
14. System.out.print("Введите пол пользователя: ");
15. **char** user\_gender = sc.next().charAt(0);
16. String n = sc.nextLine();
17. Author author = **new** Author(user\_name, user\_email, user\_gender);
18. System.out.print("\nРезультат работы метода getName: " + author.getName());
19. System.out.print("\nРезультат работы метода getEmail: " + author.getEmail());
20. System.out.print("\nРезультат работы метода getGender: " + author.getGender());
22. System.out.print("\nВызов метода setEmail: введите новый email пользователя -> ");
23. user\_email = sc.nextLine();
24. author.setEmail(user\_email);
25. System.out.print("Результат работы метода getEmail: " + author.getEmail());
26. System.out.print("\nРезультат работы метода toString: " + author.toString());
27. }
28. }
29. Задание 2

По UML диаграмме класса, представленной на рис. 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.

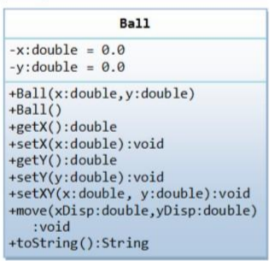


Рисунок 2 – Диаграмма класса Ball.

Листинг 8 – решение задания №2 файл Ball.java

1. **public** **class** Ball {
2. **private** **double** x = 0.0;
3. **private** **double** y = 0.0;
5. **public** Ball(**double** x, **double** y)
6. {
7. **this**.x = x;
8. **this**.y = y;
9. }
10. **public** Ball(){}
12. **public** **double** getX(){**return** x;}
13. **public** **double** getY(){**return** y;}
15. **public** **void** setX(**double** new\_x){x = new\_x;}
16. **public** **void** setY(**double** new\_y){y = new\_y;}
18. **public** **void** setXY(**double** new\_x, **double** new\_y ){x = new\_x; y = new\_y;}
20. @Override
21. **public** String toString() {
22. **return** "Ball{" +
23. "x=" + x +
24. ", y=" + y +
25. '}';
26. }
28. **public** **void** move(**double** xDisp, **double** yDisp)
29. {
30. x += xDisp;
31. y += yDisp;
32. }
33. }

Листинг 9 – решение задания №2 файл TestBall.java

1. **import** java.util.Scanner;
2. **public** **class** TestBall {
3. **public** **static** **void** main(String[] args) {
4. Scanner sc = **new** Scanner(System.in);
6. System.out.println("Введите координаты мяча:");
7. System.out.print("x: ");
8. **double** x\_ball = sc.nextDouble();
9. System.out.print("y: ");
10. **double** y\_ball = sc.nextDouble();
12. Ball ball = **new** Ball(x\_ball, y\_ball);
13. System.out.print("Результат работы метода toString -> " + ball.toString());
14. System.out.print("\nРезультат работы метода getX " + ball.getX());
15. System.out.print("\nРезультат работы метода getY " + ball.getY());
16. System.out.print("\nВызов метода setX: введите новую координату x ->  ");
17. ball.setX(sc.nextDouble());
18. System.out.print("Вызов метода setY: введите новую координату y ->  ");
19. ball.setY(sc.nextDouble());
20. System.out.print("Результат работы метода toString -> " + ball.toString());
21. System.out.println("\nВызов метода setXY:");
22. System.out.print("Введите координату x: ");
23. **double** new\_x = sc.nextDouble();
24. System.out.print("Введите координату y: ");
25. **double** new\_y = sc.nextDouble();
26. ball.setXY(new\_x, new\_y);
27. System.out.print("Результат работы метода toString -> " + ball.toString());
28. System.out.println("\nВызов метода move");
29. System.out.print("Введите смешение по координате x: ");
30. **double** x\_disp = sc.nextDouble();
31. System.out.print("Введите смешение по координате y: ");
32. **double** y\_disp = sc.nextDouble();
33. ball.move(x\_disp, y\_disp);
34. System.out.print("Результат работы метода toString -> " + ball.toString());
36. }
37. }
38. Задание 3

Создать класс точка Point, описывающий точку на плоскости. Создать Circle класс, в котором одно поле представляет точку – центр окружности, и добавить другие свойства, позволяющие задать точку на плоскости. Создать третий класс Tester который использует для хранения объектов массив объектов Circle и второе поле количество элементов в массиве.

Листинг 10 – решение задания №3 файл Point.java

1. **public** **class** Point {
2. **private** **double** x;
3. **private** **double** y;
5. **public** Point(**double** x, **double** y){
6. **this**.x = x;
7. **this**.y = y;
8. }
10. @Override
11. **public** String toString() {
12. **return** "Point{" +
13. "x=" + x +
14. ", y=" + y +
15. '}';
16. }
17. }

Листинг 11 – решение задания №3 файл Circle.java

1. **public** **class** Circle {
2. **private** Point point;
4. **public** Circle(**double** x, **double** y){
5. point = **new** Point(x, y);
6. }
8. @Override
9. **public** String toString() {
10. **return** "Circle{" +
11. "point=" + point +
12. '}';
13. }
14. }

Листинг 12 – решение задания №3 файл Tester.java

1. **import** java.util.Scanner;
2. **public** **class** Tester {
3. **public** **static** **void** main(String[] args) {
4. Scanner sc = **new** Scanner(System.in);
5. System.out.print("Введите количество желаемых объектов: ");
6. **int** cnt\_object = sc.nextInt();
7. Circle [] arr\_object = **new** Circle[cnt\_object];
8. **for**(**int** i = 0; i < cnt\_object; i++)
9. {
10. System.out.print("Введите координаты точки: ");
11. arr\_object[i] = **new** Circle(sc.nextDouble(), sc.nextDouble());
13. }
14. **for**(**int** i = 0; i < cnt\_object; i++)
15. {
16. System.out.println(arr\_object[i].toString());
17. }
18. System.out.print("Количество элементов массива: " + cnt\_object);
19. }
21. }
22. Задание 4

Напишите программу, которая меняет местами элементы одномерного массива из String в обратном порядке. Не используйте дополнительный массив для хранения результатов.

Листинг 13 – решение задания №4 файл Change.java

1. **import** java.util.Scanner;
3. **public** **class** Change {
4. **public** **static** **int** [] change(**int** [] array){
5. **int** elem;
6. **for** (**int** i = 0; i < array.length / 2; i++){
7. elem = array[i];
8. array[i] = array[array.length - 1 - i];
9. array[array.length - 1 - i] = elem;
10. }
11. **return** array;
12. }
13. **public** **static** **void** main(String[] args) {
14. Scanner scanner = **new** Scanner(System.in);
15. System.out.println("Введите количество элементов целочисленного массива: ");
16. **int** quantity = scanner.nextInt();
17. **int** array [] = **new** **int**[quantity];
18. System.out.println("Заполните массив числами: ");
19. **for** (**int** i = 0; i < quantity; i++){
20. array[i] = scanner.nextInt();
21. }
22. array = Change.change(array);
23. **for** (**int** i = 0; i < array.length; i++){
24. System.out.print(array[i] + " ");
25. }
26. }
27. }
28. Задание 5

Напишите программу HowMany.java, которая определит, сколько слов Вы ввели с консоли

Листинг 14 – решение задания №5 файл HowMany

1. **import** java.util.Scanner;
2. **public** **class** HowMany {
3. **public** **static** **void** main(String[] args) {
5. Scanner sc = **new** Scanner(System.in);
6. System.out.println("Введите слова одной строкой через пробел");
7. String input = sc.nextLine();
9. **int** count = 0;
11. **if**(input.length() != 0){
12. count++;
13. **for** (**int** i = 0; i < input.length(); i++) {
14. **if**(input.charAt(i) == ' '){
15. count++;
16. }
17. }
18. }
19. System.out.println("Вы ввели "+count+"слов");
20. }
21. }

# Практическая работа № 3. Классы Math и Random. Классы оболочки

Цель: изучить работу с классами Math и Random основные концепции объектно-ориентированного программирования, научиться программировать математические вычисления с использованием этих классов, а также познакомиться с классами оболочками и их использованием в Джава программах и научиться форматировать вывод строк.

1. Задание 1

Создайте массив из 4 случайных целых чисел из отрезка [10;99], выведите его на экран в строку, далее определите и выведите на экран сообщение о том, является ли массив строго возрастающей последовательностью.

Листинг 15 – решение задания №1

1. **import** java.util.Random;
2. **public** **class** random\_array {
3. **public** **static** **void** main(String[] args) {
4. Random rand = **new** Random();
5. **int**[] num = **new** **int**[4];
6. **for** (**int** i = 0; i < 4; i++) {
7. num[i] = rand.nextInt(89) + 10;
8. System.out.println(num[i]);
9. }
10. **for**(**int** i = 0; i < 3; i++){
11. **if** (num[i] >= num[i+1]){
12. System.out.println("Массив не является строго возрастающей последовательностью.");
13. **return**;
14. }
15. }
16. System.out.println("Массив является строго возврастающей последовательностью.");
17. }
18. }
19. Задание 2

Преобразовать значение типа String к типу double. Используем метод Double.parseDouble()

Листинг 16 – решение задания №2

1. **import** java.util.Scanner;
2. **public** **class** nomer\_2 {
3. **public** **static** **void** main(String[] args) {
4. Scanner sc = **new** Scanner(System.in);
5. System.out.print("Введите число: ");
6. String line = sc.nextLine();
7. Double val = Double.parseDouble(line);
8. **double** answer = val;
9. System.out.println(answer);
11. String d = Double.toString(3.14);
12. System.out.println("d = " + d);
13. }
14. }
15. Задание 3

Пользователь должен ввести с клавиатуры размер массива - натуральное число больше, так чтобы введенное пользователем число сохранялось в переменную n. Если пользователь ввел не подходящее число, то программа должна просить пользователя повторить ввод. Создать массив из n случайных целых чисел из отрезка [0; n] и вывести его на экран. Создать второй массив только из четных элементов первого массива, если они там есть, и вывести его на экран.

Листинг 17 – решение задания №3

1. **import** java.util.ArrayList;
2. **import** java.util.Scanner;
3. **import** java.util.Random;
4. **public** **class** nomer\_5 {
5. **public** **static** **void** main(String[] args) {
6. **int** N = 0;
7. Scanner sc = **new** Scanner(System.in);
8. Random rand = **new** Random();
10. **while** (**true**){
11. System.out.print("Введите значение n больше нуля: ");
12. N = sc.nextInt();
13. **if** (N > 0){**break**;}
14. **else**{System.out.println("Невернный ввод. Попробуйте ещё раз...");}
15. }
17. **int** [] array = **new** **int**[N];
18. ArrayList<integer> array\_2 = **new** ArrayList<integer>();
19. System.out.print("Исходный массив: ");
20. **for**(**int** i = 0; i < N; i++) {
21. array[i] = rand.nextInt(N);
22. System.out.print(array[i] + "  ");
23. **if** (array[i] % 2 == 0){array\_2.add(array[i]);}
24. }
25. System.out.print("\n");
26. System.out.print("Массив с четными числами: ");
27. **for**(**int** i = 0; i < array\_2.size();i++){
28. System.out.print(array\_2.get(i) + "  ");
29. }
30. }
31. }
32. Задание 4

Разработать класс Отчет о сотрудниках

1) Создать класс Employee, у которого есть переменные класса - fullname, salary.

2) Создать массив, содержащий несколько объектов этого типа.

3) Создать класс Report со статическим методом generateReport, в котором выводится информация о зарплате всех сотрудников.

4) Используйте форматирование строк. Пусть salary будет выровнено по правому краю, десятичное значение имеет 2 знака после запятой

Листинг 18 – решение задания №4 файл Employee.java

1. **public** **class** Employee {
2. **private** **double** salary;
3. **private** String fullname;
4. **public** Employee(String fullname, **double** salary)
5. {
6. **this**.fullname = fullname;
7. **this**.salary = salary;
8. }
9. **public** String getFullname() {
10. **return** fullname;
11. }
13. **public** **double** getSalary() {
14. **return** salary;
15. }
16. }

Листинг 19 – решение задания №4 файл report.java

1. **public** **class** report {
2. Employee employee;
3. **public** report(String name, **double** salary){
4. employee = **new** Employee(name, salary);
5. }
6. **public** **static** **void** generateReport(Employee[] employees) {
7. **for** (Employee employee : employees) {
8. System.out.printf("%-30s %10.2f\n", employee.getFullname(), employee.getSalary());
9. }
10. }
11. }

Листинг 20 – решение задания №4 файл Tester\_class.java

1. **import** java.util.Scanner;
2. **public** **class** Tester\_class {
4. **public** **static** **void** main(String[] args) {
5. Employee[] employees = {
6. **new** Employee("Иванов Иван Иванович", 50000.0),
7. **new** Employee("Петров Петр Петрович", 70000.50),
8. **new** Employee("Иванова Анна Викторовна", 60900.75)
9. };
11. report.generateReport(employees);
12. }
14. }
15. Задание 5

Заполнить таблицу Методы классов оболочек - на пересечении указать x, если данный метод существует у соответствующего класса оболочки.

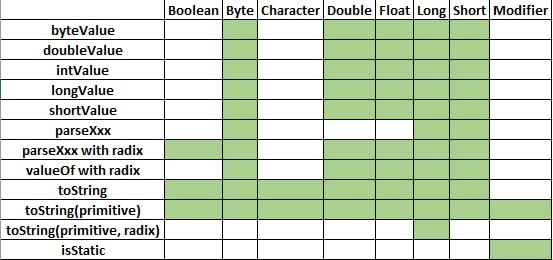


Рисунок 3 – Методы классов оболочек

# Практическая работа № 4. Перечисления и их использование в Джава программах

Цель: познакомиться с новым ссылочным типом данных перечислением, научиться разрабатывать перечисления и использовать их в своих программах.

1. Задание 1

Создать мини приложение - интернет-магазин. Должны быть реализованы следующие возможности:

* 1. Аутентификация пользователя. Пользователь вводит логин и пароль с клавиатуры.
  2. Просмотр списка каталогов товаров.
  3. Просмотр списка товаров определенного каталога.
  4. Выбор товара в корзину.
  5. Покупка товаров, находящихся в корзине.

Листинг 21 – решение задания №1 файл Product.java

1. **public** **class** Product {
2. **private** String productId;
3. **private** String name;
4. **private** **double** price;
5. **private** Category category;
7. **public** Product(String productId, String name, **double** price, Category category) {
8. **this**.productId = productId;
9. **this**.name = name;
10. **this**.price = price;
11. **this**.category = category;
12. }
14. **public** String getName() {
15. **return** name;
16. }
18. **public** **double** getPrice() {
19. **return** price;
20. }
22. **public** Category getCategory() {
23. **return** category;
24. }
26. **public** String getProductId() {
27. **return** productId;
28. }
30. @Override
31. **public** String toString() {
32. **return** "Product ID: " + productId + ", Name: " + name + ", Price: $" + price + ", Category: " + category;
33. }
34. }

Листинг 22 – решение задания №1 файл Category.java

1. **public** **enum** Category {
2. ELECTRONICS,
3. CLOTHING,
4. BOOKS,
5. BEAUTY,
6. HOME\_APPLIANCES
7. }

Листинг 23 – решение задания №1 файл User.java

1. **import** java.util.ArrayList;
2. **import** java.util.List;
3. **public** **class** User {
4. **private** String username;
5. **private** String password;
6. **private** List<product> cart;
8. **public** User(String username, String password) {
9. **this**.username = username;
10. **this**.password = password;
11. **this**.cart = **new** ArrayList<>();
12. }
13. **public** String getPassword() {
14. **return** password;
15. }
17. **public** String getUsername() {
18. **return** username;
19. }
21. **public** **void** setUsername(String username) {
22. **this**.username = username;
23. }
25. **public** **void** setPassword(String password) {
26. **this**.password = password;
27. }
29. **public** **void** setCart(List<product> cart) {
30. **this**.cart = cart;
31. }
33. **public** **void** addToCart(Product product) {
34. cart.add(product);
35. }
37. **public** **void** clearCart() {
38. cart.clear();
39. }
41. **public** List<product> getCart() {
42. **return** cart;
43. }
44. }

Листинг 24 – решение задания №1 файл Main.java

1. **public** **class** Main {
2. **public** **static** **void** main(String[] args) {
3. Store store = **new** Store();
5. // Добавление товаров
6. store.addProduct("P1", "Ноутбук", 1000.0, Category.ELECTRONICS);
7. store.addProduct("P2", "Футболка", 20.0, Category.CLOTHING);
8. store.addProduct("P3", "Книга", 15.0, Category.BOOKS);
10. // Регистрация пользователей
11. store.registerUser("user1", "password1");
12. store.registerUser("user2", "password2");
14. // Вход пользователя
15. **if** (store.login("user1", "password1")) {
16. User currentUser = store.getCurrentUser();
18. // Просмотр категорий и товаров
19. store.viewCategories();
20. store.viewProductsByCategory(Category.CLOTHING);
22. // Добавление товаров в корзину
23. store.addToCart("P2");
24. store.addToCart("P3");
26. // Оформление заказа
27. store.checkout();
28. } **else** {
29. System.out.println("Ошибка входа. Пользователь не найден.");
30. }
32. // Выход пользователя
33. store.logout();
34. }
35. }

# Практическая работа № 4.1. Наследование в Джава. Абстрактные классы.

Цель: познакомиться на практике с реализацией принципа ООП Наследование в Джава и освоить на практике работу с наследованием от абстрактных классов.

1. Задание 1

Создать абстрактный класс, описывающий Транспортное средство и подклассы Автомобиль, Самолет. Подсчитать время и стоимость перевозки пассажиров и грузов каждым транспортным средством.

Листинг 25 – решение задания №1 файл Transport.java

1. **abstract** **class** Transport {
2. **private** String name;
4. **public** Transport(String name) {
5. **this**.name = name;
6. }
8. **public** String getName() {
9. **return** name;
10. }
12. // Метод для подсчета времени перевозки пассажиров
13. **public** **abstract** **double** calculatePassengerTime(**int** passengers);
15. // Метод для подсчета стоимости перевозки пассажиров
16. **public** **abstract** **double** calculatePassengerCost(**int** passengers);
18. // Метод для подсчета времени перевозки грузов
19. **public** **abstract** **double** calculateCargoTime(**double** cargoWeight);
21. // Метод для подсчета стоимости перевозки грузов
22. **public** **abstract** **double** calculateCargoCost(**double** cargoWeight);
23. }

26. // Выход пользователя
27. store.logout();
28. }
29. }

Листинг 26 – решение задания №1 файл Car.java

1. **class** Car **extends** Transport {
2. **public** Car() {
3. **super**("Автомобиль");
4. }
6. @Override
7. **public** **double** calculatePassengerTime(**int** passengers) {
8. // Рассчет времени для автомобиля (просто для примера)
9. **return** passengers \* 0.1;
10. }
12. @Override
13. **public** **double** calculatePassengerCost(**int** passengers) {
14. // Рассчет стоимости для автомобиля (просто для примера)
15. **return** passengers \* 5.0;
16. }
18. @Override
19. **public** **double** calculateCargoTime(**double** cargoWeight) {
20. // Рассчет времени для перевозки грузов автомобилем (просто для примера)
21. **return** cargoWeight \* 0.05;
22. }
24. @Override
25. **public** **double** calculateCargoCost(**double** cargoWeight) {
26. // Рассчет стоимости для перевозки грузов автомобилем (просто для примера)
27. **return** cargoWeight \* 10.0;
28. }
29. }
30. }

Листинг 27 – решение задания №1 файл Airplane.java

1. **class** Airplane **extends** Transport {
2. **public** Airplane() {
3. **super**("Самолет");
4. }
6. @Override
7. **public** **double** calculatePassengerTime(**int** passengers) {
8. // Рассчет времени для самолета (просто для примера)
9. **return** passengers \* 0.05;
10. }
12. @Override
13. **public** **double** calculatePassengerCost(**int** passengers) {
14. // Рассчет стоимости для самолета (просто для примера)
15. **return** passengers \* 50.0;
16. }
18. @Override
19. **public** **double** calculateCargoTime(**double** cargoWeight) {
20. // Рассчет времени для перевозки грузов самолетом (просто для примера)
21. **return** cargoWeight \* 0.02;
22. }
24. @Override
25. **public** **double** calculateCargoCost(**double** cargoWeight) {
26. // Рассчет стоимости для перевозки грузов самолетом (просто для примера)
27. **return** cargoWeight \* 100.0;
28. }
29. }

Листинг 28 – решение задания №1 файл Main\_1.java

1. **public** **class** Main\_1 {
2. **public** **static** **void** main(String[] args) {
3. Transport car = **new** Car();
4. Transport airplane = **new** Airplane();
6. **int** passengers = 100;
7. **double** cargoWeight = 500.0;
9. // Пример расчета времени и стоимости для перевозки пассажиров и грузов
10. System.out.println("Перевозка " + passengers + " пассажиров на " + car.getName() + ":");
11. System.out.println("Время: " + car.calculatePassengerTime(passengers) + " часа");
12. System.out.println("Стоимость: $" + car.calculatePassengerCost(passengers));
14. System.out.println("\nПеревозка " + cargoWeight + " кг грузов на " + airplane.getName() + ":");
15. System.out.println("Время: " + airplane.calculateCargoTime(cargoWeight) + " часа");
16. System.out.println("Стоимость: $" + airplane.calculateCargoCost(cargoWeight));
17. }
18. }

# Практическая работа № 5. Создание программ с графическим интерфейсом пользователя на языке Джава

Цель: научится разрабатывать программы на языке Джава с использованием графического интерфейса пользователя.

1. Задание 1

Напишите интерактивную программу с использованием GUI имитирует таблицу результатов матчей между командами Милан и Мадрид.

Создайте пользовательское JFrame приложение, у которого есть следующие компоненты GUI:

* + одна кнопка JButton подписана “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.

Листинг 29 – решение задания №1

1. **import** javax.swing.\*;
2. **import** java.awt.\*;
3. **import** java.awt.event.ActionEvent;
4. **import** java.awt.event.ActionListener;
6. **public** **class** MatchResultTable **extends** JFrame {
7. **private** JButton milanButton;
8. **private** JButton madridButton;
9. **private** JLabel resultLabel;
10. **private** JLabel lastScorerLabel;
11. **private** JLabel winnerLabel;
12. **private** **int** milanScore;
13. **private** **int** madridScore;
15. **public** MatchResultTable() {
16. **super**("Match Result Table");
18. milanButton = **new** JButton("AC Milan");
19. madridButton = **new** JButton("Real Madrid");
20. resultLabel = **new** JLabel("Result: 0 X 0");
21. lastScorerLabel = **new** JLabel("Last Scorer: N/A");
22. winnerLabel = **new** JLabel("Winner: DRAW");
23. milanScore = 0;
24. madridScore = 0;
26. milanButton.addActionListener(**new** ActionListener() {
27. @Override
28. **public** **void** actionPerformed(ActionEvent e) {
29. milanScore++;
30. resultLabel.setText("Result: " + milanScore + " X " + madridScore);
31. lastScorerLabel.setText("Last Scorer: AC Milan");
33. **if** (milanScore > madridScore) {
34. winnerLabel.setText("Winner: AC Milan");
35. } **else** **if** (milanScore < madridScore) {
36. winnerLabel.setText("Winner: Real Madrid");
37. } **else** {
38. winnerLabel.setText("Winner: DRAW");
39. }
40. }
41. });
43. madridButton.addActionListener(**new** ActionListener() {
44. @Override
45. **public** **void** actionPerformed(ActionEvent e) {
46. madridScore++;
47. resultLabel.setText("Result: " + milanScore + " X " + madridScore);
48. lastScorerLabel.setText("Last Scorer: Real Madrid");
50. **if** (milanScore > madridScore) {
51. winnerLabel.setText("Winner: AC Milan");
52. } **else** **if** (milanScore < madridScore) {
53. winnerLabel.setText("Winner: Real Madrid");
54. } **else** {
55. winnerLabel.setText("Winner: DRAW");
56. }
57. }
58. });
60. setLayout(**new** GridLayout(5, 1));
61. add(milanButton);
62. add(madridButton);
63. add(resultLabel);
64. add(lastScorerLabel);
65. add(winnerLabel);
67. setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
68. pack();
69. setVisible(**true**);
70. }
72. **public** **static** **void** main(String[] args) {
73. SwingUtilities.invokeLater(**new** Runnable() {
74. @Override
75. **public** **void** run() {
76. **new** MatchResultTable();
77. }
78. });
79. }
80. }

# Практическая работа № 6. Интерфейсы в Java

Цель: научится разрабатывать практике пользовательские интерфейсы, и применять их в программах на языке Джава.

1. Задание 1

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, см рис. 6.4. Убедитесь, что две точки имеет одну и ту же скорость.

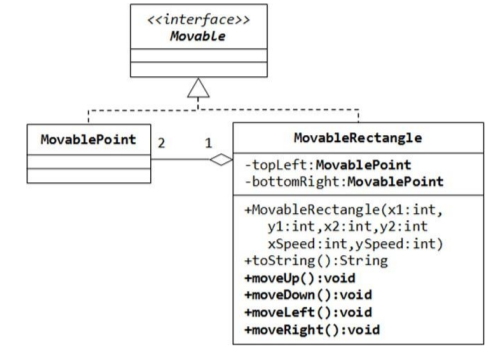


Рисунок 4 – Диаграмма класса MovableRectangle

Листинг 30 – решение задания №1 файл MovableRectangle.java

1. **public** **class** MovableRectangle **implements** Movable{
2. **private** MovablePoint topLeft;
3. **private** MovablePoint bottomRight;
5. **public** MovableRectangle(**int** x1, **int** x2, **int** y1, **int** y2, **int** xSpeed, **int** ySpeed){
6. **this**.topLeft = **new** MovablePoint(x1, y1);
7. **this**.bottomRight = **new** MovablePoint(x2, y2);
8. **if** (!(xSpeed == ySpeed)){
9. System.out.println("У точек не совпадает скорость движения. Скорость второй точки будет изменена.");
10. }
11. }
13. @Override
14. **public** **void** moveUp(**int** ySpeed){
15. **this**.topLeft.moveUp(ySpeed);
16. }
17. @Override
18. **public** **void** moveDown(**int** ySpeed){
19. **this**.topLeft.moveDown(ySpeed);
20. }
21. @Override
22. **public** **void** moveLeft(**int** xSpeed){
23. **this**.topLeft.moveLeft(xSpeed);
24. }
25. @Override
26. **public** **void** moveRight(**int** xSpeed){
27. **this**.topLeft.moveRight(xSpeed);
28. }
30. @Override
31. **public** String toString() {
32. **return** "MovableRectangle{" +
33. "topLeft=" + topLeft +
34. ", bottomRight=" + bottomRight +
35. '}';
36. }
37. }

Листинг 31 – решение задания №1 файл MovablePoint.java

1. **public** **class** MovablePoint **implements** Movable{
2. **private** **int** x;
3. **private** **int** y;
4. **public** MovablePoint(**int** x, **int** y){
5. **this**.x = x;
6. **this**.y = y;
7. }
9. @Override
10. **public** **void** moveUp(**int** ySpeed){
11. **this**.y += ySpeed;
12. }
13. @Override
14. **public** **void** moveDown(**int** ySpeed){
15. **this**.y -= ySpeed;
16. }
17. @Override
18. **public** **void** moveLeft(**int** xSpeed){
19. **this**.x -= xSpeed;
20. }
21. @Override
22. **public** **void** moveRight(**int** xSpeed){
23. **this**.x += xSpeed;
24. }
26. @Override
27. **public** String toString() {
28. **return** "MovablePoint{" +
29. "x=" + x +
30. ", y=" + y +
31. '}';
32. }
33. }

Листинг 32 – решение задания №1 файл Movable.java

1. **interface** Movable {
2. **void** moveUp(**int** ySpeed);
3. **void** moveDown(**int** ySpeed);
4. **void** moveLeft(**int** xSpeed);
5. **void** moveRight(**int** xSpeed);
6. }

Листинг 33 – решение задания №1 файл Main.java

1. **public** **class** Main {
2. **public** **static** **void** main(String args[]){
3. **int** xSpeed = 2;
4. **int** ySpeed = 2;
5. MovableRectangle rect = **new** MovableRectangle(0,0,0,0,2,2);
6. System.out.println(rect.toString());
7. rect.moveLeft(xSpeed);
8. rect.moveUp(ySpeed);
9. System.out.println(rect.toString());
10. }
11. }
12. Задание 2

Напишите программу для перевода температуры по Цельсию в температуру по Кельвину и Фаренгейту. Для этого добавьте интерфейс Convertable у которого есть метод convert для конвертации из одной системы измерения в другую.

Листинг 34 – решение задания №2 файл Convertible.java

1. **public** **interface** Convertible {
2. **double** convert(**double** temperature);
3. }

Листинг 35 – решение задания №2 файл CelsiusToFahrenheitConverter.java

1. **public** **class** CelsiusToFahrenheitConverter **implements** Convertible {
2. @Override
3. **public** **double** convert(**double** temperature) {
4. **return** (temperature \* 9 / 5) + 32;
5. }
6. }

Листинг 36 – решение задания №2 файл CelsiusToKelvinConverter.java

1. **public** **class** CelsiusToKelvinConverter **implements** Convertible {
2. @Override
3. **public** **double** convert(**double** temperature) {
4. **return** temperature + 273.15;
5. }
6. }

Листинг 37 – решение задания №2 файл TemperatureConverter.java

1. **import** java.util.Scanner;
2. **public** **class** TemperatureConverter {
3. **public** **static** **void** main(String[] args) {
4. Scanner sc = **new** Scanner(System.in);
5. Convertible celsiusToKelvinConverter = **new** CelsiusToKelvinConverter();
6. Convertible celsiusToFahrenheitConverter = **new** CelsiusToFahrenheitConverter();
7. System.out.print("Enter the temperature: ");
8. **double** celsiusTemperature = sc.nextDouble();
10. **double** kelvinTemperature = celsiusToKelvinConverter.convert(celsiusTemperature);
11. System.out.println(celsiusTemperature + " °C = " + kelvinTemperature + " K");
13. **double** fahrenheitTemperature = celsiusToFahrenheitConverter.convert(celsiusTemperature);
14. System.out.println(celsiusTemperature + " °C = " + fahrenheitTemperature + " °F");
15. }
16. }

# Практическая работа № 7. Реализация интерфейсов

Цель: научится разрабатывать практике пользовательские интерфейсы, и применять их в программах на языке Джава.

1. Задание 1

Напишите по диаграмме класс MovableRectangle (движущийся прямоугольник), реализующий интерфейс Movable, класс прямоугольник, который можно представить как две движущиеся точки MovablePoint (верхняя левая и нижняя правая точки – topLeft и bottomRight), также реализующие интерфейс Movable.

Листинг 38 – решение задания №1 файл MovablePoint.java

1. **class** MovablePoint **implements** Movable {
2. **int** x, y, xSpeed, ySpeed;
4. **public** MovablePoint(**int** x, **int** y, **int** xSpeed, **int** ySpeed) {
5. **this**.x = x;
6. **this**.y = y;
7. **this**.xSpeed = xSpeed;
8. **this**.ySpeed = ySpeed;
9. }
11. **public** String toString() {
12. **return** "(" + x + "," + y + ")";
13. }
15. **public** **void** moveUp() {
16. y -= ySpeed;
17. }
19. **public** **void** moveDown() {
20. y += ySpeed;
21. }
23. **public** **void** moveLeft() {
24. x -= xSpeed;
25. }
27. **public** **void** moveRight() {
28. x += xSpeed;
29. }
30. }

Листинг 39 – решение задания №1 файл MovableRectangle.java

1. **public** **class** MovableRectangle **implements** Movable{
2. MovablePoint topLeft, bottomRight;
4. **public** MovableRectangle(**int** x1, **int** y1, **int** x2, **int** y2, **int** xSpeed, **int** ySpeed) {
5. topLeft = **new** MovablePoint(x1, y1, xSpeed, ySpeed);
6. bottomRight = **new** MovablePoint(x2, y2, xSpeed, ySpeed);
7. }
9. **public** **boolean** SpeedTest() {
10. **return** topLeft.xSpeed == bottomRight.xSpeed && topLeft.ySpeed == bottomRight.ySpeed;
11. }
13. **public** **void** moveUp() {
14. topLeft.moveUp();
15. bottomRight.moveUp();
16. }
18. **public** **void** moveDown() {
19. topLeft.moveDown();
20. bottomRight.moveDown();
21. }
23. **public** **void** moveLeft() {
24. topLeft.moveLeft();
25. bottomRight.moveLeft();
26. }
28. **public** **void** moveRight() {
29. topLeft.moveRight();
30. bottomRight.moveRight();
31. }
32. }

Листинг 40 – решение задания №1 файл Movable.java

1. **interface** Movable {
2. **void** moveUp();
4. **void** moveDown();
6. **void** moveLeft();
8. **void** moveRight();
9. }

Листинг 41 – решение задания №1 файл MovableRectangleTest.java

1. **public** **class** MovableRectangleTest {
2. **public** **static** **void** main(String[] args) {
3. MovableRectangle rectangle = **new** MovableRectangle(0, 0, 10, 10, 2, 3);
5. System.out.println("Initial position of the rectangle:");
6. System.out.println("Top left: " + rectangle.topLeft);
7. System.out.println("Bottom right: " + rectangle.bottomRight);
9. rectangle.moveUp();
11. System.out.println("Position of the rectangle after moving up:");
12. System.out.println("Top left: " + rectangle.topLeft);
13. System.out.println("Bottom right: " + rectangle.bottomRight);
15. System.out.println("Do the two points have the same speed? " + rectangle.SpeedTest());
16. }
17. }

# Практическая работа № 8. Рекурсия Программирование рекурсии в Java. Решение задач на рекурсию

Цель: разработка и программирование рекурсивных алгоритмов на языке Java.

1. Задание 1

Дано натуральное число N. Вычислите сумму его цифр. При решении этой задачи нельзя использовать строки, списки, массивы (ну и циклы, разумеется).

Листинг 42 – решение задания №1

1. **import** java.util.Scanner;
3. **public** **class** number\_5 {
4. **public** **static** **void** main(String[] args) {
5. Scanner sc = **new** Scanner(System.in);
6. System.out.print("Введите число: ");
7. **int** N = sc.nextInt();
8. **int** sum = sumDigits(N);
9. System.out.println("Сумма цифр числа " + N + " равна " + sum);;
10. }
11. **public** **static** **int** sumDigits(**int** number) {
12. **if** (number == 0) {
13. **return** 0;}
14. **return** number % 10 + sumDigits(number / 10);
15. }
16. }
17. Задание 2

Дано натуральное число N. Выведите все его цифры по одной, в обычном порядке, разделяя их пробелами или новыми строками.

Листинг 43 – решение задания №2

1. **import** java.util.Scanner;
2. **public** **class** number\_14 {
3. **public** **static** **void** main(String[] args) {
4. Scanner sc = **new** Scanner(System.in);
5. System.out.print("Введите число: ");
6. **int** N = sc.nextInt();
7. print\_digitals(N);
8. }
9. **public** **static** **void** print\_digitals(**int** number){
10. **if** (number > 9){
11. print\_digitals(number/10);
12. }
13. System.out.print(number % 10 + " ");
14. }
15. }

# Практическая работа № 9. Использование полиморфизма при программировании при реализации алгоритмов сортировок и поиска

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

1. Задание 1

Написать тестовый класс, который создает массив класса Student и сортирует массив iDNumber и сортирует его вставками.

Листинг 44 – решение задания №1 файл InsertionSort.java

1. **import** java.util.ArrayList;
2. **import** java.util.List;
4. **public** **class** InsertionSort {
5. **public** **static** <t **extends**="" comparable<t="">> **void** sort(List<t> list) {
6. **int** n = list.size();
8. **for** (**int** i = 1; i < n; i++) {
9. T key = list.get(i);
10. **int** j = i - 1;
12. // Сдвигаем элементы, которые больше key, вправо
13. **while** (j >= 0 && list.get(j).compareTo(key) > 0) {
14. list.set(j + 1, list.get(j));
15. j = j - 1;
16. }
17. list.set(j + 1, key);
18. }
19. }
20. }

Листинг 45 – решение задания №1 файл Student.java

1. **public** **class** Student **implements** Comparable<student> {
2. **private** String name;
3. **private** Integer idNumber;
4. **private** Integer GPA; //  Grade Point Average — это усредненный балл оценок
6. **public** Student(String name, Integer idNumber, Integer GPA) {
7. **this**.name = name;
8. **this**.idNumber = idNumber;
9. **this**.GPA = GPA;
10. }
12. **public** String getName() {
13. **return** name;
14. }
16. **public** Integer getIdNumber() {
17. **return** idNumber;
18. }
20. **public** Integer getGPA() {
21. **return** GPA;
22. }
24. @Override
25. **public** **int** compareTo(Student o) {
26. **return** (**this**.idNumber - o.idNumber);
27. }
29. @Override
30. **public** String toString() {
31. **return** "Student{" +
32. "name='" + name + '\'' +
33. ", ID=" + idNumber +
34. ", GPA=" + GPA +
35. '}';
36. }
37. }

Листинг 46 – решение задания №1 файл Test.java

1. **import** java.util.ArrayList;
2. **import** java.util.List;
4. **public** **class** Test {
5. **public** **static** **void** main(String[] args) {
6. List<student> students = **new** ArrayList<>();
7. students.add(**new** Student("Татьяна", 4, 1));
8. students.add(**new** Student("Ульяна", 3, 2));
9. students.add(**new** Student("Максим", 2, 3));
10. students.add(**new** Student("Яна", 1, 4));
12. InsertionSort.sort(students);
13. System.out.println(students);
14. }
15. }
16. Задание 2

Напишите класс SortingStudentsByGPA который реализует интерфейс Comparator таким образом, чтобы сортировать список студентов по их итоговым баллам в порядке убывания с использованием алгоритма быстрой сортировки.

Листинг 47 – решение задания №2 файла QuickSort.java

1. **import** java.util.ArrayList;
2. **import** java.util.Comparator;
3. **import** java.util.List;
5. **public** **class** QuickSort {
6. **public** **static** <t> **void** quickSort(List<t> list, Comparator<t> comparator) {
7. quickSortHelper(list, comparator, 0, list.size() - 1);
8. }
10. **private** **static** <t> **void** quickSortHelper(List<t> list, Comparator<t> comparator, **int** low, **int** high) {
11. **if** (low < high) {
12. **int** pivotIndex = partition(list, comparator, low, high);
13. quickSortHelper(list, comparator, low, pivotIndex - 1);
14. quickSortHelper(list, comparator, pivotIndex + 1, high);
15. }
16. }
18. **private** **static** <t> **int** partition(List<t> list, Comparator<t> comparator, **int** low, **int** high) {
19. T pivot = list.get(high);
20. **int** i = low - 1;
22. **for** (**int** j = low; j < high; j++) {
23. **if** (comparator.compare(list.get(j), pivot) <= 0) {
24. i++;
25. swap(list, i, j);
26. }
27. }
29. swap(list, i + 1, high);
30. **return** i + 1;
31. }
33. **private** **static** <t>  **void** swap(List<t> list, **int** i, **int** j) {
34. T temp = list.get(i);
35. list.set(i, list.get(j));
36. list.set(j, temp);
37. }
38. }

Листинг 48 – решение задания №2 файла QuickSort1.java

1. **import** java.util.ArrayList;
2. **import** java.util.Comparator;
3. **import** java.util.List;
5. **public** **class** QuickSort1 {
6. **public** **static** <t> **void** quickSort(List<t> list, Comparator<t> comparator) {
7. **if** (list.size() < 2)
8. **return**; // базовый случай (массивы из 0 или 1 элемента)
9. **else** {
10. T pivot = list.get(0);
11. List<t> less = **new** ArrayList<>();
12. List<t> greater = **new** ArrayList<>();
13. **for** (**int** i = 1; i < list.size(); i++) {
14. **if** (comparator.compare(list.get(i), pivot) > 0)
15. greater.add(list.get(i));
16. **else** **if** (comparator.compare(list.get(i), pivot) <= 0)
17. less.add(list.get(i));
18. }
19. quickSort(less, comparator);
20. quickSort(greater, comparator);
22. list.clear(); // Очищаем исходный список
24. list.addAll(less); // Добавляем элементы из less
25. list.add(pivot); //  добавляем pivot
26. list.addAll(greater); // Добавляем элементы из greater
27. }
28. }
29. }

Листинг 49 – решение задания №2 файла SortingStudentsByGPA.java

1. **import** java.util.Comparator;
2. **import** java.util.List;
4. **public** **class** SortingStudentsByGPA **implements** Comparator<student> {
5. @Override
6. **public** **int** compare(Student student1, Student student2) {
7. **return** Integer.compare(student2.getGPA(), student1.getGPA());
8. }
9. **public** **static** **void** sort(List<student> students) {
10. QuickSort.quickSort(students, **new** SortingStudentsByGPA());
11. }
12. }

Листинг 50 – решение задания №2 файла Test.java

1. **import** java.util.ArrayList;
2. **import** java.util.List;
4. **public** **class** Test {
5. **public** **static** **void** main(String[] args) {
6. List<student> students = **new** ArrayList<>();
7. students.add(**new** Student("Татьяна", 4, 1));
8. students.add(**new** Student("Ульяна", 3, 2));
9. students.add(**new** Student("Максим", 2, 3));
10. students.add(**new** Student("Яна", 1, 4));
12. System.out.println(students);
13. SortingStudentsByGPA.sort(students);
14. System.out.println(students);
15. }
16. }
17. Задание 3

Напишите программу, которая объединяет два списка данных о студентах в один отсортированный списках с использованием алгоритма сортировки слиянием.

Листинг 51 – решение задания №3 файла Tester.java

1. **import** practica\_9.task1.InsertionSort;
2. **import** practica\_9.task1.Student;
4. **import** java.util.ArrayList;
5. **import** java.util.List;
7. **public** **class** Tester {
8. **public** **static** **void** main(String[] args) {
9. List<student> students1 = **new** ArrayList<>();
10. students1.add(**new** Student("Татьяна", 4, 1));
11. students1.add(**new** Student("Ульяна", 3, 2));
12. students1.add(**new** Student("Максим", 2, 3));
13. students1.add(**new** Student("Яна", 1, 4));
14. List<student> students2 = **new** ArrayList<>();
15. students2.add(**new** Student("Марианна", 8, 5));
16. students2.add(**new** Student("Михаил", 5, 6));
17. students2.add(**new** Student("Светлана", 6, 7));
18. students2.add(**new** Student("Александра", 7, 8));
20. List<student> students = merge(students1, students2);
21. System.out.println(students);
22. }
24. **public** **static** List<student> merge(List<student> students1, List<student> students2) {
25. InsertionSort.sort(students1);
26. InsertionSort.sort(students2);
28. List<student> mergedList = **new** ArrayList<>();
29. **int** i = 0;
30. **int** j = 0;
32. **while** (i < students1.size() && j < students2.size()) {
34. **if** (students1.get(i).compareTo(students2.get(j)) < 0) {
35. mergedList.add(students1.get(i++));
36. } **else** {
37. mergedList.add(students1.get(i++));
38. }
39. }
41. **while** (i < students1.size()) {
42. mergedList.add(students1.get(i++));
43. }
45. **while** (j < students2.size()) {
46. mergedList.add(students2.get(j++));
47. }
49. **return** mergedList;
50. }
51. }

# Практическая работа № 10. Стандартные интерфейсы Джава. Интерфейс Comparator

Цель: цель данной практической работы - закрепить знания в области использования стандартных интерфейсов языка Джава, научиться применять интерфейсы для разработки практических программ на Джаве.

1. Задание 1

Создать свой класс Student со всеми переменными экземпляра, конструктором, включающим все переменные, предпочтительно использовать 79 геттеры и сеттеры для каждой переменной. Класс студент имеет свойства: Имя, Фамилия, Специальность, Курс, Группа

Листинг 52 – Решение задания №1 из файла Student.java

1. **public** **class** Student {
2. **private** String firstName;
3. **private** String secondName;
4. **private** String specialty;
5. **private** **int** course;
6. **private** String group;
7. **private** **int** GPA;
9. **public** Student(String firstName, String secondName, String specialty, **int** course, String group, **int** GPA) {
10. **this**.firstName = firstName;
11. **this**.secondName = secondName;
12. **this**.specialty = specialty;
13. **this**.course = course;
14. **this**.group = group;
15. **this**.GPA = GPA;
16. }
18. **public** String getFirstName() {
19. **return** firstName;
20. }
22. **public** **void** setFirstName(String firstName) {
23. **this**.firstName = firstName;
24. }
26. **public** String getSecondName() {
27. **return** secondName;
28. }
30. **public** **void** setSecondName(String secondName) {
31. **this**.secondName = secondName;
32. }
34. **public** String getSpecialty() {
35. **return** specialty;
36. }
38. **public** **void** setSpecialty(String specialty) {
39. **this**.specialty = specialty;
40. }
42. **public** **int** getCourse() {
43. **return** course;
44. }
46. **public** **void** setCourse(**int** course) {
47. **this**.course = course;
48. }
50. **public** String getGroup() {
51. **return** group;
52. }
54. **public** **void** setGroup(String group) {
55. **this**.group = group;
56. }
58. **public** **int** getGPA() {
59. **return** GPA;
60. }
62. **public** **void** setGPA(**int** GPA) {
63. **this**.GPA = GPA;
64. }
66. @Override
67. **public** String toString() {
68. **return** "Студент{" +
69. "имя='" + firstName + '\'' +
70. ", фамилия='" + secondName + '\'' +
71. ", специальность='" + specialty + '\'' +
72. ", курс=" + course +
73. ", группа='" + group + '\'' +
74. ", средний балл=" + GPA +
75. '}';
76. }
77. }
78. Задание 2

Напишите класс SortingStudentsByGPA (может у вас называться Tester или Main, так как содержит функцию main()) создайте поле как массив объектов Student с названием iDNumber, вы можете использовать как массив, так и и ArrayList или TreeSet для хранения данных о студентах Добавьте методы класса: 1) заполнения массива setArray() 2) метод для сортировки по среднему баллу студентов quicksort() который реализует интерфейс Comparator таким образом, чтобы он сортировал студентов с их итоговым баллом в порядке убывания. В качестве алгоритма сортировки использовать методы сортировок: слиянием и быструю сортировку (добавьте в класс еще один метод). 3)метод для вывода массива студентов outArray() 4)Добавьте в класс возможность сортировать список студентов по другому полю

Листинг 53 – Решение задания №2, файлы из папки comparators

1. **import** ru.mirea.lab10.Student;
3. **import** java.util.Comparator;
5. **public** **class** ComparatorByCourse **implements** Comparator<student> {
7. @Override
8. **public** **int** compare(Student s1, Student s2) {
9. **return** Integer.compare(s1.getCourse(), s2.getCourse());
10. }
11. }
12. **import** ru.mirea.lab10.Student;
13. **import** java.util.Comparator;
15. **public** **class** ComparatorByFirstName **implements** Comparator<student> {
16. @Override
17. **public** **int** compare(Student s1, Student s2) {
18. **return** s1.getFirstName().compareTo(s2.getFirstName());
19. }
20. }
21. **import** ru.mirea.lab10.Student;
23. **import** java.util.Comparator;
25. **public** **class** ComparatorByGPA **implements** Comparator<student> {
27. @Override
28. **public** **int** compare(Student s1, Student s2) {
29. **return** Integer.compare(s1.getGPA(), s2.getGPA());
30. }
31. }
32. **import** ru.mirea.lab10.Student;
33. **import** java.util.Comparator;
35. **public** **class** ComparatorByGroup **implements** Comparator<student> {
36. @Override
37. **public** **int** compare(Student s1, Student s2) {
38. **return** s1.getGroup().compareTo(s2.getGroup());
39. }
40. }
41. **import** ru.mirea.lab10.Student;
42. **import** java.util.Comparator;
44. **public** **class** ComparatorBySecondName **implements** Comparator<student> {
45. @Override
46. **public** **int** compare(Student s1, Student s2) {
47. **return** s1.getSecondName().compareTo(s2.getSecondName());
48. }
49. }
50. **import** ru.mirea.lab10.Student;
51. **import** java.util.Comparator;
53. **public** **class** ComparatorBySpeciality **implements** Comparator<student> {
54. @Override
55. **public** **int** compare(Student s1, Student s2) {
56. **return** s1.getSpecialty().compareTo(s2.getSpecialty());
57. }
58. }
59. Задание 3

Напишите программу, которая объединяет два списка данных о студентах в один отсортированный списках.

Листинг 54 – Решение задания №3 из файла SortingStudents.java

1. **import** ru.mirea.lab10.comparators.\*;
3. **import** java.util.ArrayList;
4. **import** java.util.Comparator;
5. **import** java.util.List;
6. **import** java.util.Scanner;
8. **public** **class** SortingStudents {
9. **public** **static** **void** main(String[] args) {
10. setArray();
11. outArray();
12. mergeSort();
13. outArray();
14. }
15. **private** **static** Comparator<student> comparator = **new** ComparatorByGPA();
16. **private** **static** ArrayList<student> iDNumber = **new** ArrayList<>();
18. **public** **static** **void** setArray() {
19. Scanner in = **new** Scanner(System.in);
20. System.out.print("Введите размер массива: ");
21. **int** n = in.nextInt();
22. in.nextLine();
23. String firstName, secondName, speciality, group;
24. **int** GPA, course;
25. **for** (**int** i = 0; i< n; i++) {
26. System.out.println("Заполните данные о студенте №" + (i+1));
27. System.out.print("Имя: ");
28. firstName = in.nextLine();
29. System.out.print("Фамилия: ");
30. secondName = in.nextLine();
31. System.out.print("Специальность: ");
32. speciality = in.nextLine();
33. System.out.print("Курс: ");
34. course = in.nextInt();
35. in.nextLine();
36. System.out.print("Группа: ");
37. group = in.nextLine();
38. System.out.print("Средний балл: ");
39. GPA = in.nextInt();
40. in.nextLine();
41. iDNumber.add(**new** Student(firstName, secondName, speciality, course, group, GPA));
42. }
43. }
45. **public** **static** **void** quickSort() {
46. Scanner in = **new** Scanner(System.in);
47. System.out.println("Введите поле, по которому будут сортироваться студенты: ");
48. String choice = in.nextLine();
49. **switch** (choice) {
50. **case** "средний балл":
51. comparator = **new** ComparatorByGPA();
52. **break**;
53. **case** "имя":
54. comparator = **new** ComparatorByFirstName();
55. **break**;
56. **case** "фамилия":
57. comparator = **new** ComparatorBySecondName();
58. **break**;
59. **case** "курс":
60. comparator = **new** ComparatorByCourse();
61. **break**;
62. **case** "группа":
63. comparator = **new** ComparatorByGroup();
64. **break**;
65. **case** "специальность":
66. comparator = **new** ComparatorBySpeciality();
67. **break**;
68. **default**:
69. **break**;
70. }
71. quickSortHelper(iDNumber, comparator, 0, iDNumber.size() - 1);
72. }
74. **private** **static** **void** quickSortHelper(List<student> list, Comparator<student> comparator, **int** low, **int** high) {
75. **if** (low < high) {
76. **int** pivotIndex = partition(list, comparator, low, high);
77. quickSortHelper(list, comparator, low, pivotIndex - 1);
78. quickSortHelper(list, comparator, pivotIndex + 1, high);
79. }
80. }
82. **private** **static** **int** partition(List<student> list, Comparator<student> comparator, **int** low, **int** high) {
83. Student pivot = list.get(high);
84. **int** i = low - 1;
86. **for** (**int** j = low; j < high; j++) {
87. **if** (comparator.compare(list.get(j), pivot) <= 0) {
88. i++;
89. swap(list, i, j);
90. }
91. }
93. swap(list, i + 1, high);
94. **return** i + 1;
95. }
97. **public** **static** **void** mergeSort() {
98. Scanner in = **new** Scanner(System.in);
99. System.out.println("Введите поле, по которому будут сортироваться студенты: ");
100. String choice = in.nextLine();
101. **switch** (choice) {
102. **case** "средний балл":
103. comparator = **new** ComparatorByGPA();
104. **break**;
105. **case** "имя":
106. comparator = **new** ComparatorByFirstName();
107. **break**;
108. **case** "фамилия":
109. comparator = **new** ComparatorBySecondName();
110. **break**;
111. **case** "курс":
112. comparator = **new** ComparatorByCourse();
113. **break**;
114. **case** "группа":
115. comparator = **new** ComparatorByGroup();
116. **break**;
117. **case** "специальность":
118. comparator = **new** ComparatorBySpeciality();
119. **break**;
120. **default**:
121. **break**;
122. }
123. mergeSortHelper(iDNumber, comparator);
124. }
125. **public** **static** **void** mergeSortHelper(List<student> list, Comparator<student> comparator) {
126. **if** (list.size() <= 1) {
127. **return**; // Базовый случай: если список содержит 1 элемент или пуст, то он уже отсортирован
128. }
130. **int** mid = list.size() / 2; // Найдем средний индекс
131. List<student> left = **new** ArrayList<>(list.subList(0, mid)); // Создаем копию левой части
132. List<student> right = **new** ArrayList<>(list.subList(mid, list.size())); // Создаем копию правой части
134. mergeSortHelper(left, comparator); // Рекурсивно сортируем левую часть
135. mergeSortHelper(right, comparator); // Рекурсивно сортируем правую часть
137. merge(list, left, right, comparator); // Объединяем отсортированные части
138. }
140. **public** **static** **void** merge(List<student> result, List<student> left, List<student> right, Comparator<student> comparator) {
141. **int** i = 0, j = 0, k = 0;
143. **while** (i < left.size() && j < right.size()) {
144. **if** (comparator.compare(left.get(i), right.get(j)) <= 0) {
145. result.set(k++, left.get(i++));
146. } **else** {
147. result.set(k++, right.get(j++));
148. }
149. }
151. **while** (i < left.size()) {
152. result.set(k++, left.get(i++));
153. }
155. **while** (j < right.size()) {
156. result.set(k++, right.get(j++));
157. }
158. }
160. **private** **static** **void** swap(List<student> list, **int** i, **int** j) {
161. Student temp = list.get(i);
162. list.set(i, list.get(j));
163. list.set(j, temp);
164. }
166. **public** **static** **void** outArray() {
167. System.out.println("СТУДЕНТЫ --> ");
168. **for** (Student student : iDNumber) {
169. System.out.println(student.toString());
170. }
171. }
172. }

# Практическая работа № 11. Работа с датой и временем

Цель данной практической работы – научиться работать с датами и временем, применять методы класса Date и Calenadar, других классов для обработки строк.

1. Задание 1

Написать программу, выводящую фамилию разработчика, дату и время получения задания, а также дату и время сдачи задания. Для получения последней даты и времени использовать класс Date из пакета java.util.\* (Объявление Dated=newDate() или метод System.currentTimeMillis().

Листинг 55 – Решение задания №1 из файла Main1.java

1. **import** java.util.Date;
3. **public** **class** Main1 {
4. **public** **static** **void** main(String[] args) {
5. String developerSurname = "Иванов";
6. Date startDate = **new** Date();
7. Date endDate = **new** Date();
9. System.out.println("Фамилия разработчика: " + developerSurname);
10. System.out.println("Дата получения задания: " + startDate);
11. System.out.println("Дата сдачи задания: " + endDate);
12. }
13. }
14. Задание 2

Приложение, сравнивающее текущую дату и дату, введенную пользователем c текущим системным временем

Листинг 56 – Решение задания №2 из файла Main2.java

1. **import** java.text.ParseException;
2. **import** java.text.SimpleDateFormat;
3. **import** java.util.Date;
4. **import** java.util.Scanner;
6. **public** **class** Main2 {
7. **public** **static** **void** main(String[] args) {
8. Date currentDate = **new** Date();
10. Scanner scanner = **new** Scanner(System.in);
11. System.out.print("Введите дату и время в формате ГГГГ-ММ-ДД ЧЧ:ММ: ");
12. String inputDateTime = scanner.nextLine();
14. SimpleDateFormat dateFormat = **new** SimpleDateFormat("yyyy-MM-dd HH:mm");
15. **try** {
16. Date userDateTime = dateFormat.parse(inputDateTime);
18. **if** (userDateTime.compareTo(currentDate) > 0) {
19. System.out.println("Введенная дата и время больше текущей даты и времени");
20. } **else** **if** (userDateTime.compareTo(currentDate) < 0) {
21. System.out.println("Введенная дата и время меньше текущей даты и времени");
22. } **else** {
23. System.out.println("Введенная дата и время равны текущей дате и времени");
24. }
25. } **catch** (ParseException e) {
26. System.out.println("Неправильный формат даты и времени");
27. }
28. }
29. }
30. Задание 3

Напишите пользовательский код, который формирует объекты Date и Calendar по следующим данным, вводимым пользователем:

<Год><Месяц><Число>

<Часы><Минуты>

Листинг 57 – Решение задания №3 из файла Main4.java

1. **import** java.util.Calendar;
2. **import** java.util.Date;
3. **import** java.util.Scanner;
5. **public** **class** Main4 {
6. **public** **static** **void** main(String[] args) {
7. Scanner scanner = **new** Scanner(System.in);
9. System.out.print("Введите год: ");
10. **int** year = scanner.nextInt();
12. System.out.print("Введите месяц (от 1 до 12): ");
13. **int** month = scanner.nextInt() - 1;
15. System.out.print("Введите число: ");
16. **int** day = scanner.nextInt();
18. System.out.print("Введите часы: ");
19. **int** hours = scanner.nextInt();
21. System.out.print("Введите минуты: ");
22. **int** minutes = scanner.nextInt();
24. Calendar calendar = Calendar.getInstance();
25. calendar.set(year, month, day, hours, minutes);
27. Date date = calendar.getTime();
29. System.out.println("Созданная дата и время: " + date);
30. }
31. }

# Практическая работа № 12. Создание программ с графическим интерфейсом пользователя на языке Джава. Компоновка объектов с помощью Layout менеджеров

Цель: цель данной практической работы - научиться создавать графический интерфейс пользователя, освоить на практике работу с различными объектами для создания GUI, менеджерами размещения компонентов.

1. Задание 1

Создать окно, нарисовать в нем 20 случайных фигур, случайного цвета. Классы фигур должны наследоваться от абстрактного класса Shape, в котором описаны свойства фигуры: цвет, позиция.

Листинг 58 – Решение задания №1 из файла Circle.java

1. **import** java.awt.\*;
3. **public** **class** Circle **extends** Shape {
4. **protected** **int** radius;
5. **public** Circle(Color color, **int** x, **int** y, **int** radius) {
6. **super**(color, x, y);
7. **this**.radius = radius;
8. }
10. **public** **void** draw (Graphics g) {
11. g.setColor(**this**.color);
12. g.fillOval(x, y, radius, radius);
13. }
14. }

Листинг 59 – Решение задания №1 из файла Rectangle.java

1. **import** java.awt.\*;
3. **public** **class** Rectangle **extends** Shape {
4. **protected** **int** width;
5. **protected** **int** length;
6. **public** Rectangle(Color color, **int** x, **int** y, **int** width, **int** length) {
7. **super**(color, x, y);
8. **this**.width = width;
9. **this**.length = length;
10. }
12. **public** **void** draw (Graphics g) {
13. g.setColor(**this**.color);
14. g.fillRect(x, y, width, length);
15. }
16. }

Листинг 60 – Решение задания №1 из файла SecondGUI.java

1. **import** javax.swing.\*;
2. **import** java.awt.\*;
3. **import** java.util.Random;

6. **public** **class** SecondGUI **extends** JFrame{
7. **private** **static** Shape[] shapes = **new** Shape[20];
8. **public** **void** paint(Graphics g) {
9. **super**.paint(g);
10. **for** (Shape shape : shapes) {
11. shape.draw(g);
12. }
13. }
14. **public** SecondGUI() {
15. setTitle("DZ2");
16. setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
17. setSize(600, 600);
18. Shape shape;
19. Random random = **new** Random();
20. **int** circleOrRect, x, y, width, height, radius;
22. **for** (**int** i = 0; i < 20; i++) {
23. circleOrRect = random.nextInt(2);
24. x = random.nextInt(380);
25. y = random.nextInt(380);
26. Color color = **new** Color(random.nextInt(256), random.nextInt(256), random.nextInt(256));
28. **if** (circleOrRect == 0) {
29. width = random.nextInt(110);
30. height = random.nextInt(110);
31. shape = **new** Rectangle(color, x, y, width, height);
32. } **else** {
33. radius = random.nextInt(50);
34. shape = **new** Circle(color, x, y, radius);
35. }
36. shapes[i] = shape;
37. }
38. }
39. }

Листинг 61 – Решение задания №1 из файла Shape.java

1. **import** java.awt.\*;
3. **public**  **abstract** **class** Shape {
4. **protected** Color color;
5. **protected** **int** x;
6. **protected** **int** y;
8. **public** Shape() {
9. }
11. **public** Shape(Color color, **int** x, **int** y) {
12. **this**.color = color;
13. **this**.x = x;
14. **this**.y = y;
15. }
17. **public** Color getColor() {
18. **return** color;
19. }
21. **public** **void** setColor(Color color) {
22. **this**.color = color;
23. }
25. **public** **abstract** **void** draw(Graphics g);
27. }

Листинг 62 – Решение задания №1 из файла Main.java

1. **public** **class** Main {
2. **public** **static** **void** main(String args[]) {
3. SecondGUI secondGUI = **new** SecondGUI();
4. secondGUI.setVisible(**true**);
5. }
6. }

# Практическая работа № 13. Обработка строк в Java

Цель: закрепить знания в области обработки строк, научиться применять методы класса String и других классов для обработки строк.

1. Задание 1

Доработать класс адреса, который из полученной строки формата “Страна[d] Регион[d] Город[d] Улица[d] Дом[d] Корпус[d] Квартира” ([d] – разделитель, например, «запятая») выбирает соответствующие части и записывает их в соответствующие поля класса Address. Учесть, что в начале и конце разобранной части адреса не должно быть пробелов. Все поля адреса строковые. Разработать проверочный класс не менее чем на четыре адресных строки. В программе предусмотреть две реализации этого метода: а) разделитель – только запятая (использовать метод split()); Внимание, при разработке нужно учесть, что б) разделитель – любой из символов ‘,’ ‘.’ ‘;’ - (класс StringTokenizer ).

Листинг 63 – Решение задания №1 из файла Address.java

1. **import** java.util.StringTokenizer;
2. **public** **class** Address {
3. **private** String country;
4. **private** String region;
5. **private** String city;
6. **private** String street;
7. **private** String house;
8. **private** String building;
9. **private** String apartment;
11. **public** Address(String addressString, String delimiters) {
12. StringTokenizer tokenizer = **new** StringTokenizer(addressString, delimiters);
13. **if** (tokenizer.hasMoreTokens()) {
14. **this**.country = tokenizer.nextToken().trim();
15. }
16. **if** (tokenizer.hasMoreTokens()) {
17. **this**.region = tokenizer.nextToken().trim();
18. }
19. **if** (tokenizer.hasMoreTokens()) {
20. **this**.city = tokenizer.nextToken().trim();
21. }
22. **if** (tokenizer.hasMoreTokens()) {
23. **this**.street = tokenizer.nextToken().trim();
24. }
25. **if** (tokenizer.hasMoreTokens()) {
26. **this**.house = tokenizer.nextToken().trim();
27. }
28. **if** (tokenizer.hasMoreTokens()) {
29. **this**.building = tokenizer.nextToken().trim();
30. }
31. **if** (tokenizer.hasMoreTokens()) {**this**.apartment = tokenizer.nextToken().trim();
32. }
33. }
35. // getters and setters
37. **public** String toString() {
38. StringBuilder sb = **new** StringBuilder();
39. **if** (country != **null** && !country.isEmpty()) {
40. sb.append(country);
41. }
42. **if** (region != **null** && !region.isEmpty()) {
43. sb.append(", ").append(region);
44. }
45. **if** (city != **null** && !city.isEmpty()) {
46. sb.append(", ").append(city);
47. }
48. **if** (street != **null** && !street.isEmpty()) {
49. sb.append(", ").append(street);
50. }
51. **if** (house != **null** && !house.isEmpty()) {
52. sb.append(", ").append(house);
53. }
54. **if** (building != **null** && !building.isEmpty()) {
55. sb.append(", ").append(building);
56. }
57. **if** (apartment != **null** && !apartment.isEmpty()) {
58. sb.append(", ").append(apartment);
59. }
60. **return** sb.toString();
61. }
62. }

Листинг 64 – Решение задания №1 из файла Main.java

1. **public** **class** Main {
2. **public** **static** **void** main(String[] args) {
3. // разделитель - запятая
4. String addressString1 = "Россия, Московская область, Москва, Ленина улица, 10, 2, 5";
5. Address address1 = **new** Address(addressString1, ",");
6. System.out.println(address1.toString());
8. // разделитель - запятая
9. String addressString2 = "Украина, Харьковская область, Харьков, Пушкинская улица, 15";
10. Address address2 = **new** Address(addressString2, ",");
11. System.out.println(address2.toString());
13. // разделитель - точка с запятой
14. String addressString3 = "США; Нью-Йорк; Нью-Йорк; Бродвей; 123; 4";
15. Address address3 = **new** Address(addressString3, ";");
16. System.out.println(address3.toString());
18. // разделитель - точка с запятой
19. String addressString4 = "Германия; Берлин; Берлин; Фридрихштрассе; 50; 3; 10";
20. Address address4 = **new** Address(addressString4, ";");
21. System.out.println(address4.toString());
22. }
23. }

# Практическая работа № 14. Использование регулярных выражений в Джава приложениях

Цель: практической работы – понять особенности использования регулярных выражений в Java, научиться работать с строками и применять регулярные выражения для обработки строк в программах.

1. Задание 1

Написать регулярное выражение, определяющее является ли данная строка строкой "abcdefghijklmnopqrstuv18340" или нет.

a) пример правильных выражений: abcdefghijklmnopqrstuv18340

b) пример неправильных выражений: abcdefghijklmnoasdfasdpqrstuv18340.

Листинг 65 – Решение задания №1 из файла Main.java

1. **import** java.util.regex.Pattern;
2. **public** **class** Main {
3. **public** **static** **void** main(String[] args) {
4. String string1 = "abcdefghijklmnopqrstuv18340";
5. String string2 = "abcdefghijklmnoasdfasdpqrstuv18340.";
7. String pattern = "abcdefghijklmnopqrstuv18340";
9. **boolean** isMatch1 = Pattern.matches(pattern, string1);
10. **boolean** isMatch2 = Pattern.matches(pattern, string2);
12. System.out.println(isMatch1); // true
13. System.out.println(isMatch2); // false
14. }
15. }
16. Задание 2

Дан текст со списками цен. Извлечь из него цены в USD, RUВ, EU.

a) пример правильных выражений: 25.98 USD.

b) пример неправильных выражений: 44 ERR, 0.004 EU.

Листинг 66 – Решение задания №2 из файла Main2.java

1. **import** java.util.regex.Matcher;
2. **import** java.util.regex.Pattern;
4. **public** **class** Main2 {
6. **public** **static** **void** main(String[] args) {
7. String text = "Цены: 25.98 USD, 44 RUB, 0.004 EU";
9. String patternUSD = "\\b\\d+(\\.\\d+)?\\s+USD\\b";
10. String patternRUB = "\\b\\d+(\\.\\d+)?\\s+RUB\\b";
11. String patternEU = "\\b\\d+(\\.\\d+)?\\s+EU\\b";
13. Pattern pattern = Pattern.compile(patternUSD + "|" + patternRUB + "|" + patternEU);
14. Matcher matcher = pattern.matcher(text);
16. **while** (matcher.find()) {
17. String price = matcher.group().trim();
18. System.out.println(price);
19. }
20. }
21. }
22. Задание 3

Написать регулярное выражение, определяющее является ли данная строчка датой в формате dd/mm/yyyy. Начиная с 1900 года до 9999 года.

a) пример правильных выражений: 29/02/2000, 30/04/2003, 01/01/2003.

b) пример неправильных выражений: 29/02/2001, 30-04-2003, 1/1/1899

Листинг 67 – Решение задания №3 из файла Main3.java

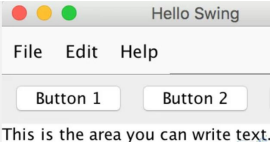
1. **import** java.util.regex.Pattern;
3. **public** **class** Main3 {
5. **public** **static** **void** main(String[] args) {
6. String[] dates = {"29/02/2000", "30/04/2003", "29/02/2001", "30-04-2003", "1/1/1899"};
8. String regex = "^(0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/((19|2[0-9])[0-9]{2}|9999)$";
9. Pattern pattern = Pattern.compile(regex);
11. **for** (String date : dates) {
12. **boolean** isMatch = pattern.matcher(date).matches();
13. System.out.println(date + ": " + isMatch);
14. }
15. }
16. }

# Практическая работа № 15. Вложенные и внутренние классы. Обработка событий в Джава программах с графическим интерфейсом пользователя

Цель данной практической работы - изучить использование анонимных и внутренних классов, научиться разрабатывать интерактивные программы на языке Джава с использованием графического интерфейса пользователя.

1. Задание 1

Разработайте программу с меню, двумя кнопками и текстовым полем ввода. В этой программе у вас должны быть разные настройки в меню. Должно быть меню «Файл», которое включает в себя подменю «Сохранить», «Выйти» и «Правка», включая подменю «Копировать, вырезать, вставить» и меню «Справка». Вид окна программы должен иметь вид как на рисунке ниже.



Листинг 68 – Решение задания №1 из файла Main.java

1. **import** javax.swing.\*;
2. **import** java.awt.\*;
3. **import** java.awt.event.ActionEvent;
4. **import** java.awt.event.ActionListener;
6. **public** **class** Main {
8. **public** **static** **void** main(String[] args) {
9. // Создание главного окна
10. JFrame frame = **new** JFrame("Программа с меню");
11. frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
13. // Создание панели содержимого
14. JPanel panel = **new** JPanel();
15. panel.setLayout(**new** GridLayout(3, 1));
17. // Создание текстового поля
18. JTextArea textArea = **new** JTextArea();
19. panel.add(textArea);
21. // Создание кнопок
22. JButton button1 = **new** JButton("Кнопка 1");
23. JButton button2 = **new** JButton("Кнопка 2");
24. panel.add(button1);
25. panel.add(button2);
27. // Создание меню
28. JMenuBar menuBar = **new** JMenuBar();
30. JMenu fileMenu = **new** JMenu("Файл");
31. JMenuItem saveMenuItem = **new** JMenuItem("Сохранить");
32. JMenuItem exitMenuItem = **new** JMenuItem("Выйти");
33. fileMenu.add(saveMenuItem);
34. fileMenu.add(exitMenuItem);
35. menuBar.add(fileMenu);
37. JMenu editMenu = **new** JMenu("Правка");
38. JMenuItem copyMenuItem = **new** JMenuItem("Копировать");
39. JMenuItem cutMenuItem = **new** JMenuItem("Вырезать");
40. JMenuItem pasteMenuItem = **new** JMenuItem("Вставить");
41. editMenu.add(copyMenuItem);
42. editMenu.add(cutMenuItem);
43. editMenu.add(pasteMenuItem);
44. menuBar.add(editMenu);
46. JMenu helpMenu = **new** JMenu("Справка");
47. menuBar.add(helpMenu);
49. frame.setJMenuBar(menuBar);
51. // Обработчик события нажатия кнопки 1
52. button1.addActionListener(**new** ActionListener() {
53. @Override
54. **public** **void** actionPerformed(ActionEvent e) {
55. String text = textArea.getText();
56. textArea.setText(text + "Нажата кнопка 1\n");
57. }
58. });
60. // Обработчик события нажатия кнопки 2
61. button2.addActionListener(**new** ActionListener() {
62. @Override
63. **public** **void** actionPerformed(ActionEvent e) {
64. String text = textArea.getText();
65. textArea.setText(text + "Нажата кнопка 2\n");
66. }
67. });
69. // Обработчик события выбора пункта меню "Сохранить"
70. saveMenuItem.addActionListener(**new** ActionListener() {
71. @Override
72. **public** **void** actionPerformed(ActionEvent e) {
73. String text = textArea.getText();
74. // Здесь можно добавить код для сохранения текста в файл
75. JOptionPane.showMessageDialog(frame, "Текст успешно сохранен!");
76. }
77. });
79. // Обработчик события выбора пункта меню "Выйти"
80. exitMenuItem.addActionListener(**new** ActionListener() {
81. @Override
82. **public** **void** actionPerformed(ActionEvent e) {
83. System.exit(0);
84. }
85. });
87. frame.getContentPane().add(panel);
88. frame.pack();
89. frame.setVisible(**true**);
90. }
91. }

# Практическая работа № 16. Вложенные и внутренние классы. Обработка событий в Джава программах с графическим интерфейсом пользователя Обработка событий мыши и клавиатуры программах на Джава с графическим интерфейсом пользователя

Цель: научиться обрабатывать различные события мыши и клавиатуры для разных компонентов.

1. Задание 1

Реализуйте программу на Джава с использованием JTextArea и двумя следующего меню выбора:

a) Цвет: который имеет возможность выбора из три возможных: синий, красный и черный

b) Шрифт: три вида: “Times New Roman”, “MS Sans Serif”, “Courier New”.

*Замечание.* Вы должны написать программу, которая с помощью меню, может изменять шрифт и цвет текста, написанного в JTextArea

Листинг 68 – Решение задания №1 из файла Main.java

1. **import** javax.swing.\*;
2. **import** java.awt.\*;
3. **import** java.awt.event.ActionEvent;
4. **import** java.awt.event.ActionListener;
6. **public** **class** Main {
8. **public** **static** **void** main(String[] args) {
9. // Создание главного окна
10. JFrame frame = **new** JFrame("Программа с меню выбора");
11. frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
13. // Создание панели содержимого
14. JPanel panel = **new** JPanel();
15. panel.setLayout(**new** BorderLayout());
17. // Создание текстового поля
18. JTextArea textArea = **new** JTextArea();
19. panel.add(textArea, BorderLayout.CENTER);
21. // Создание меню выбора цвета
22. JMenu colorMenu = **new** JMenu("Цвет");
23. JMenuItem blueMenuItem = **new** JMenuItem("Синий");
24. JMenuItem redMenuItem = **new** JMenuItem("Красный");
25. JMenuItem blackMenuItem = **new** JMenuItem("Черный");
26. colorMenu.add(blueMenuItem);
27. colorMenu.add(redMenuItem);
28. colorMenu.add(blackMenuItem);
30. // Создание меню выбора шрифта
31. JMenu fontMenu = **new** JMenu("Шрифт");
32. JMenuItem timesNewRomanMenuItem = **new** JMenuItem("Times New Roman");
33. JMenuItem msSansSerifMenuItem = **new** JMenuItem("MS Sans Serif");
34. JMenuItem courierNewMenuItem = **new** JMenuItem("Courier New");
35. fontMenu.add(timesNewRomanMenuItem);
36. fontMenu.add(msSansSerifMenuItem);
37. fontMenu.add(courierNewMenuItem);
39. // Создание меню
40. JMenuBar menuBar = **new** JMenuBar();
41. menuBar.add(colorMenu);
42. menuBar.add(fontMenu);
44. frame.setJMenuBar(menuBar);
46. // Обработчик события выбора пункта меню цвета
47. ActionListener colorMenuItemListener = **new** ActionListener() {
48. @Override
49. **public** **void** actionPerformed(ActionEvent e) {
50. JMenuItem menuItem = (JMenuItem) e.getSource();
51. String color = menuItem.getText();
53. **switch** (color) {
54. **case** "Синий":
55. textArea.setForeground(Color.BLUE);
56. **break**;
57. **case** "Красный":
58. textArea.setForeground(Color.RED);
59. **break**;
60. **case** "Черный":
61. textArea.setForeground(Color.BLACK);
62. **break**;
63. }
64. }
65. };
67. // Привязываем обработчик события к пунктам меню цвета
68. blueMenuItem.addActionListener(colorMenuItemListener);
69. redMenuItem.addActionListener(colorMenuItemListener);
70. blackMenuItem.addActionListener(colorMenuItemListener);
72. // Обработчик события выбора пункта меню шрифта
73. ActionListener fontMenuItemListener = **new** ActionListener() {
74. @Override
75. **public** **void** actionPerformed(ActionEvent e) { JMenuItem menuItem = (JMenuItem) e.getSource();
76. String font = menuItem.getText();
78. **switch** (font) {
79. **case** "Times New Roman":
80. textArea.setFont(**new** Font("Times New Roman", Font.PLAIN, 12));
81. **break**;
82. **case** "MS Sans Serif":
83. textArea.setFont(**new** Font("MS Sans Serif", Font.PLAIN, 12));
84. **break**;
85. **case** "Courier New":
86. textArea.setFont(**new** Font("Courier New", Font.PLAIN, 12));
87. **break**;
88. }
89. }
90. };
92. // Привязываем обработчик события к пунктам меню шрифта
93. timesNewRomanMenuItem.addActionListener(fontMenuItemListener);
94. msSansSerifMenuItem.addActionListener(fontMenuItemListener);
95. courierNewMenuItem.addActionListener(fontMenuItemListener);
97. // Добавляем панель на главное окно и задаем размеры окна
98. frame.getContentPane().add(panel);
99. frame.setSize(400, 300);
100. frame.setVisible(**true**);
101. }
102. }

# Практическая работа № 17. Разработка интерактивных программ на языке Джава с использованием паттерна MVC

Цель: введение в разработку программ c с использованием событийного программирования на языке программирования Джава с использованием паттерна MVC.

1. Задание 1

Напишите реализацию программного кода, с использованием паттерна MVC для расчета заработной платы сотрудника предприятия. Предлагается использовать следующие классы.

* Класс Employee – сотрудник будет выступать в качестве слоя модели
* Класс EmployeeView будет действовать как слой представления.
* Класс EmployeeContoller будет действовать как уровень контроллера.

Листинг 69 – Решение задания №1 из файла Employee.java

1. **public** **class** Employee {
2. **private** String name;
3. **private** **double** salary;
5. **public** Employee(String name, **double** salary) {
6. **this**.name = name;
7. **this**.salary = salary;
8. }
10. **public** String getName() {
11. **return** name;
12. }
14. **public** **double** getSalary() {
15. **return** salary;
16. }
18. **public** **void** setSalary(**double** salary) {
19. **this**.salary = salary;
20. }
21. }

Листинг 70 – Решение задания №1 из файла EmployeeController.java

1. **public** **class** EmployeeController {
2. **private** Employee model;
3. **private** EmployeeView view;
5. **public** EmployeeController(Employee model, EmployeeView view) {
6. **this**.model = model;
7. **this**.view = view;
8. }
10. **public** String getEmployeeName() {
11. **return** model.getName();
12. }
14. **public** **double** getEmployeeSalary() {
15. **return** model.getSalary();
16. }
18. **public** **void** setEmployeeSalary(**double** salary) {
19. model.setSalary(salary);
20. }
22. **public** **void** updateView() {
23. view.printEmployeeDetails(model.getName(), model.getSalary());
24. }
25. }

Листинг 71 – Решение задания №1 из файла EmployeeView.java

1. **public** **class** EmployeeView {
2. **public** **void** printEmployeeDetails(String name, **double** salary) {
3. System.out.println("Employee: " + name);
4. System.out.println("Salary: " + salary);
5. }
6. }

Листинг 72 – Решение задания №1 из файла MVCEmployeeExample.java

1. **public** **class** MVCEmployeeExample {
2. **public** **static** **void** main(String[] args) {
3. // Создаем модель, представление и контроллер
4. Employee employee = **new** Employee("John Doe", 50000);
5. EmployeeView view = **new** EmployeeView();
6. EmployeeController controller = **new** EmployeeController(employee, view);
8. // Выводим информацию о сотруднике
9. controller.updateView();
11. // Изменяем заработную плату и обновляем представление
12. controller.setEmployeeSalary(60000);
13. controller.updateView();
14. }
15. }

# Практическая работа № 18. Исключения и работа с ними в Джава

Цель данной практической работы являются получение практических навыков разработки программ, изучение синтаксиса языка Java, освоение основных конструкций языка Java (циклы, условия, создание переменных и массивов, создание методов, вызов методов), а также научиться осуществлять стандартный ввод/вывод данных. Ключевые слова: try, catch, finally, throw, throws

1. Задание 1

Шаг 1. Выполните следующую программу и посмотрите, что происходит:

Листинг – Пример обработки деления на ноль

public class Exception1 {

public void exceptionDemo() {

System.out.println( 2 / 0 );

}

}

Описание работы Вам необходимо инстанцировать класс и выполнить exceptionDemo(). Что произойдет?

Ответ: программа даст сбой, и вы получите следующее сообщение java.lang.ArithmeticException: / by zero at Exception1.exceptionDemo(Exception1.java:12)

Это говорит нам о том, что программа пытается выполнить деление на ноль, который он не в состоянии выполнить.

Объясните поведение программы.

Шаг 2. Измените программу следующим образом.

Замените 2/0 на 2,0 / 0,0 и повторно вызовите метод. Что произойдет?

Теперь измените код в классе Exception1 и включите блок try-catch следующим образом:

Листинг – Пример обработки исключения

public class Exception1 {

public void exceptionDemo() {

try{

System.out.println( 2/0 );

} catch ( ArithmeticException e ) {

System.out.println("Attempted division by zero");

}

}

}

Шаг 3. Запустите программу и обратите внимание на новое поведение. Объясните поведение программы.

Листинг 73 – Решение задания №1 из файла Exception1.java

1. **public** **class** Exception1 {
2. //        public void exceptionDemo() {
3. //            System.out.println(2.0 / 0.0);
4. ////            Изменение выражения с 2/0 на 2.0 / 0.0 не изменит суть проблемы. Деление на ноль все равно недопустимо в математике и в Java,
5. ////                    и это вызовет ту же java.lang.ArithmeticException.
6. ////                    В данном случае, результат будет бесконечностью (Infinity).
7. //        }
8. **public** **void** exceptionDemo() {
9. **try**{
10. System.out.println( 2/0 );
11. } **catch** ( ArithmeticException e ) {
12. System.out.println("Attempted division by zero");
13. }
14. }
15. }

Листинг 74 – Решение задания №1 из файла Exception\_main.java

1. **public** **class** Exception\_main {
2. **public** **static** **void** main(String[] args) {
3. Exception1 exp1 = **new** Exception1();
4. exp1.exceptionDemo();
5. //      программа пытается выполнить деление на ноль, что приводит к исключению java.lang.ArithmeticException.
6. //      Это происходит потому, что деление на ноль в математике не определено и вызывает исключение.

9. //        После добавления try catch
10. //        Изменение кода с добавлением блока try-catch позволяет обрабатывать исключение внутри программы,
11. //                вместо того чтобы позволить ему вызвать сбой приложения. В блоке try программа попытается выполнить деление,
12. //        но если произойдет деление на ноль, она перехватит исключение ArithmeticException и выполнит код в блоке catch.
13. //        В данном случае, код в блоке catch выводит сообщение "Attempted division by zero".
14. }
15. }
16. Задание 2

Самый простой способ генерации исключения показан в следующем примере кода:

Листинг – Класс ThrowsDemo

public class ThrowsDemo {

public void getDetails(String key) {

if(key == null) {

throw newNullPointerException("null key in getDetails" );

}

// do something with the key

}

}

Шаг 1. Когда определяется условие ошибки, то мы выбрасываем исключение с определенным именем. Сообщение может быть связано с исключением. Откомпилируйте этот класс, создайте его экземпляр и выполните метод getDetails() с нулем в качестве значения параметра.

Вы можете получить следующийвывод:

java.lang.NullPointerException: null key in getDetails at ThrowsDemo.getDetails(ThrowsDemo.java:13)

Шаг 2. Добавьте блок try-catch таким образом, чтобы перехватить исключение и рассмотреть его внутри метода.

Подумайте, является ли этот способ подходящим, чтобы иметь дело с этим исключением?

Объясните поведение программы. Ответ. Причиной ошибки, может является, например неправильное значение для параметра. Может было бы лучше, если бы метод вызывал getDetails() и там решалась бы эта проблема.

Обратите внимание на следующее:

Листинг – Пример видоизмененной программы

ThrowsDemo public class ThrowsDemo {

public void printMessage(String key) {

String message = getDetails(key); System.out.println( message );

}

public String getDetails(String key) {

if(key == null) {

throw new NullPointerException( “null key in getDetails” );

}

return “data for” + key;

}

}

Листинг 75 – Решение задания №2 из файла ThrowsDemo.java

1. **public** **class** ThrowsDemo {
2. //    public void getDetails(String key) {
3. //        try {
4. //            if (key == null) {
5. //                throw new NullPointerException("null key in getDetails");
6. //            }
7. //        } catch (NullPointerException e){
8. //            System.out.println("Exception caught");
9. //        }
10. //    }
11. **public** **void** printMessage(String key) {
12. //    String message = getDetails(key);
13. //    System.out.println( message );
15. String message;
16. **try** {
17. message = getDetails(key);
18. } **catch** (NullPointerException e) {
19. message = "An error occurred: " + e.getMessage();
20. }
21. System.out.println(message);
22. }
23. **public** String getDetails(String key) {
24. **if**(key == **null**) {
25. **throw** **new** NullPointerException( "null key in getDetails" );
26. }
27. **return** "data for " + key; }
28. }

Листинг 76 – Решение задания №2 из файла ThrowsDemo \_main.java

1. **public** **class** ThrowsDemo\_main {
2. **public** **static** **void** main(String[] args) {
3. ThrowsDemo throwsDemo = **new** ThrowsDemo();
4. //        throwsDemo.getDetails(null);
5. // Вызываем метод с правильным значением ключа
6. String validKey = "example";
7. throwsDemo.printMessage(validKey);
9. // Вызываем метод с null в качестве значения ключа
10. String nullKey = **null**;
11. throwsDemo.printMessage(nullKey);
12. }
13. }

# Практическая работа № 19. Создание пользовательских исключений

Цель данной практической работы – научиться создавать собственные исключения.

1. Задание 1

Клиент совершает покупку онлайн. При оформлении заказа у пользователя запрашивается фио и номер ИНН. В программе проверяется, действителен ли номер ИНН для такого клиента. Исключение будет выдано в том случае, если введен недействительный ИНН.

Листинг 77 – Решение задания №1 из файла InnValidator.java

1. **public** **class** InnValidator {
2. **public** **static** **boolean** validateInn(String inn) **throws** InvalidInnException {
3. **if** (inn == **null** || inn.length() != 12) {
4. **throw** **new** InvalidInnException("Недействительный ИНН: неверная длина");
5. }
7. **char**[] innChars = inn.toCharArray();
8. **int**[] coefficients = {7, 2, 4, 10, 3, 5, 9, 4, 6, 8};
10. **int** sum = 0;
11. **for** (**int** i = 0; i < 10; i++) {
12. sum += (innChars[i] - '0') \* coefficients[i];
13. }
15. **int** controlDigit = sum % 11;
16. **if** (controlDigit == 10) {
17. controlDigit = 0;
18. }
20. **if** (controlDigit != (innChars[11] - '0')) {
21. **throw** **new** InvalidInnException("Недействительный ИНН: неверная контрольная цифра");
22. }
24. **return** **true**;
25. }
26. **public** **static** **void** main(String[] args) {
27. **try** {
28. String userInn = "771237370181";
30. validateInn(userInn);
31. System.out.println("ИНН действителен.");
32. } **catch** (InvalidInnException e) {
33. System.out.println("ИНН недействителен: " + e.getMessage());
34. }
35. }
36. }

Листинг 78 – Решение задания №1 из файла InnValidatorException.java

1. **public** **class** InvalidInnException **extends** Exception{
2. **public** InvalidInnException(String message) {
3. **super**(message);
4. }
5. }

# Практическая работа № 20. Работа с дженериками.

Цель данной практической работы – научиться работать с обобщенными типами в Java и применять их в программах.

1. Задание 1

Написать обобщенный класс MinMax, который содержит методы для нахождения минимального и максимального элемента массива. Массив является переменной класса. Массив должен передаваться в класс через конструктор. Написать класс Калькулятор (необобщенный), который содержит обобщенные статические методы - sum, multiply, divide, subtraction. Параметры этих методов - два числа разного типа, над которыми должна быть произведена операция.

Листинг 79 – Решение задания №1 из файла Calculator.java

1. **public** **class** Calculator {
2. **public** **static** <t **extends**="" number,="" u="" number=""> **double** sum(T a, U b) {
3. **return** a.doubleValue() + b.doubleValue();
4. }
6. **public** **static** <t **extends**="" number,="" u="" number=""> **double** multiply(T a, U b) {
7. **return** a.doubleValue() \* b.doubleValue();
8. }
10. **public** **static** <t **extends**="" number,="" u="" number=""> **double** divide(T a, U b) {
11. **if** (b.doubleValue() == 0) {
12. **throw** **new** ArithmeticException("Division by zero");
13. }
14. **return** a.doubleValue() / b.doubleValue();
15. }
17. **public** **static** <t **extends**="" number,="" u="" number=""> **double** subtraction(T a, U b) {
18. **return** a.doubleValue() - b.doubleValue();
19. }
20. }

Листинг 80 – Решение задания №1 из файла Main.java

1. **public** **class** Main {
2. **public** **static** **void** main(String[] args) {
3. // Задание 1: Создать обобщенный класс с тремя параметрами
4. MyGenericClass<integer, string,="" **double**=""> genericObject = **new** MyGenericClass<>(42, "Hello", 3.14);
5. System.out.println("Value of T: " + genericObject.getValue1());
6. System.out.println("Value of V: " + genericObject.getValue2());
7. System.out.println("Value of K: " + genericObject.getValue3());
8. genericObject.printClassNames();
9. System.out.println();

12. // Задание 3: Написать обобщенный класс MinMax
13. Integer[] intArray = {3, 8, 1, 5, 7};
14. MinMax<integer> minMaxInteger = **new** MinMax<>(intArray);
15. System.out.println("Min: " + minMaxInteger.findMin());
16. System.out.println("Max: " + minMaxInteger.findMax());
17. System.out.println();
19. // Задание 4: Написать класс Калькулятор
20. **double** resultSum = Calculator.sum(10, 5);
21. System.out.println("Sum: " + resultSum);
22. **double** resultMultiply = Calculator.multiply(10, 5);
23. System.out.println("Multiply: " + resultMultiply);
24. **double** resultDivide = Calculator.divide(10, 5);
25. System.out.println("Divide: " + resultDivide);
26. **double** resultSubtraction = Calculator.subtraction(10, 5);
27. System.out.println("Subtraction: " + resultSubtraction);
28. System.out.println();
30. }
31. }

Листинг 81 – Решение задания №1 из файла MinMax.java

1. **public** **class** MinMax<t **extends**="" comparable<t="">> {
2. **private** T[] array;
4. **public** MinMax(T[] array) {
5. **this**.array = array;
6. }
8. **public** T findMin() {
9. **if** (array.length == 0) {
10. **return** **null**;
11. }
13. T min = array[0];
14. **for** (T item : array) {
15. **if** (item.compareTo(min) < 0) {
16. min = item;
17. }
18. }
20. **return** min;
21. }
23. **public** T findMax() {
24. **if** (array.length == 0) {
25. **return** **null**;
26. }
28. T max = array[0];
29. **for** (T item : array) {
30. **if** (item.compareTo(max) > 0) {
31. max = item;
32. }
33. }
35. **return** max;
36. }
37. }

Листинг 82 – Решение задания №1 из файла MyGenericClass.java

1. **public** **class** MyGenericClass<t, v,="" k=""> {
2. **private** T value1;
3. **private** V value2;
4. **private** K value3;
6. **public** MyGenericClass(T value1, V value2, K value3) {
7. **this**.value1 = value1;
8. **this**.value2 = value2;
9. **this**.value3 = value3;
10. }
12. **public** T getValue1() {
13. **return** value1;
14. }
16. **public** V getValue2() {
17. **return** value2;
18. }
20. **public** K getValue3() {
21. **return** value3;
22. }
24. **public** **void** printClassNames() {
25. System.out.println("Type of T: " + value1.getClass().getName());
26. System.out.println("Type of V: " + value2.getClass().getName());
27. System.out.println("Type of K: " + value3.getClass().getName());
28. }
29. }

# Практическая работа № 21. Стирание типов в Джава

Цель данной практической работы – научиться работать с обобщенными типами в Java и применять прием стирание типов разработке программ на Джава

1. Задание 1

Написать метод для конвертации массива строк/чисел в список.

Листинг 82 – Решение задания №1 из файла Converter.java

1. **import** java.util.ArrayList;
2. **import** java.util.List;
4. **public** **class** Converter {
5. **public** **static** <t> List<t> arrayToList(T[] array) {
6. List<t> list = **new** ArrayList<>();
7. **for** (T element : array) {
8. list.add(element);
9. }
10. **return** list;
11. }
12. }

Листинг 83 – Решение задания №1 из файла GenericArray.java

1. **public** **class** GenericArray<t> {
2. **private** T[] array;
4. **public** GenericArray(T[] array) {
5. **this**.array = array;
6. }
8. **public** T getElement(**int** index) {
9. **if** (index < 0 || index >= array.length) {
10. **throw** **new** IndexOutOfBoundsException("Index is out of bounds");
11. }
12. **return** array[index];
13. }
14. }

Листинг 84 – Решение задания №1 из файла Main.java

1. **import** java.util.List;
3. **public** **class** Main {
4. **public** **static** **void** main(String[] args) {
5. // Задание 1: Метод для конвертации массива строк/чисел в список
6. Integer[] intArray = {1, 2, 3, 4, 5};
7. String[] stringArray = {"One", "Two", "Three", "Four", "Five"};
9. List<integer> intList = Converter.arrayToList(intArray);
10. List<string> stringList = Converter.arrayToList(stringArray);
12. System.out.println("Integer List: " + intList);
13. System.out.println("String List: " + stringList);
15. // Задание 2: Класс, который хранит в себе массив любых типов данных
16. Double[] doubleArray = {1.1, 2.2, 3.3, 4.4, 5.5};
17. GenericArray<**double**> doubleArrayHolder = **new** GenericArray<>(doubleArray);
19. String[] stringArray2 = {"A", "B", "C", "D", "E"};
20. GenericArray<string> stringArrayHolder = **new** GenericArray<>(stringArray2);
22. System.out.println("Double Element: " + doubleArrayHolder.getElement(2));
23. System.out.println("String Element: " + stringArrayHolder.getElement(3));
25. // Задание 4: Функция, которая сохранит содержимое каталога в список и выведет первые 5 элементов
26. String directoryPath = "your\_directory\_path\_here"; // Замените на путь к каталогу
27. List<string> files = DirectoryReader.readDirectory(directoryPath);
29. System.out.println("Contents of the directory:");
30. **for** (String file : files) {
31. System.out.println(file);
32. }
33. }
34. }
35. Задание 2

Написать функцию, которая сохранит содержимое каталога в список и выведет первые 5 элементов на экран.

Листинг 85 – Решение задания №2 из файла DirectoryReader.java

1. **import** java.io.File;
2. **import** java.util.ArrayList;
3. **import** java.util.List;
5. **public** **class** DirectoryReader {
6. **public** **static** List<string> readDirectory(String directoryPath) {
7. List<string> fileList = **new** ArrayList<>();
8. File directory = **new** File(directoryPath);
9. **if** (directory.isDirectory()) {
10. File[] files = directory.listFiles();
11. **if** (files != **null**) {
12. **for** (**int** i = 0; i < Math.min(5, files.length); i++) {
13. fileList.add(files[i].getName());
14. }
15. }
16. }
17. **return** fileList;
18. }
20. **public** **static** **void** main(String[] args) {
21. String directoryPath = "C:\\Users\\Sasha\\Labs";
22. List<string> files = readDirectory(directoryPath);
23. **for** (String file : files) {
24. System.out.println(file);
25. }
26. }
27. }

# Практическая работа № 22. Абстрактные типы данных. Стек

Цель данной практической работы – научиться разрабатывать программы с абстрактными типами данных на языке Джава и применять паттерн MVC при разработке программ

1. Задание 1

Напишите программу-калькулятор арифметических выражений записанных в обратной польской нотации (RPN-калькулятор).

Листинг 86 – Решение задания №1 из файла ArithmeticOperator.java

1. **import** java.io.File;
2. **public** **enum** ArithmeticOperation {
3. ADDITION("+") {
4. **public** **double** action(**double** x, **double** y) { **return** x + y; }
5. }, SUBTRACTION("-") {
6. **public** **double** action(**double** x, **double** y) { **return** x - y; }
7. }, MULTIPLY("\*") {
8. **public** **double** action(**double** x, **double** y) { **return** x \* y; }
9. }, DIVISION("/") {
10. **public** **double** action(**double** x, **double** y) { **return** x / y; }
11. };
13. **private** **final** String symbol;
15. ArithmeticOperation(String symbol) {
16. **this**.symbol = symbol;
17. }
19. **public** String getSymbol() { **return** symbol; }
21. **public** **abstract** **double** action (**double** x, **double** y);
22. }

Листинг 87 – Решение задания №1 из файла Main.java

1. **public** **class** Main {
2. **public** **static** **void** main(String[] args) **throws** NotCorrectPolishNotationException {
4. PolishCalculator polishCalculator = **new** PolishCalculator();
5. System.out.println(polishCalculator.getResult(polishCalculator.readInputString()));
6. }
7. }

Листинг 88 – Решение задания №1 из файла NotCorrectPolishNotationException.java

1. **public** **class** NotCorrectPolishNotationException **extends** Exception {
2. **public** NotCorrectPolishNotationException(String message) {
3. **super**(message);
4. }
5. }

Листинг 89 – Решение задания №1 из файла PolishCalculator.java

1. **import** java.util.Scanner;
2. **import** java.util.Stack;
4. **public** **class** PolishCalculator {
6. **private** **final** Scanner input = **new** Scanner(System.in);
7. **private** **final** ArithmeticOperation[] arithmeticOperations = ArithmeticOperation.values();

10. **public** **double** getResult(String[] strings) **throws** NotCorrectPolishNotationException {
11. Stack<**double**> stack = **new** Stack<>();
12. **for** (String string : strings) {
13. **if** (isNumber(string)) {
14. stack.push(Double.parseDouble(string));
15. } **else** **if** (isArithmeticOperation(string)) {
16. **try** {
17. **double** temp2 = stack.pop();
18. **double** temp1 = stack.pop();
20. stack.push(ArithmeticOperation.values()[getIndexOfOperation(string)].action(temp1, temp2));
22. } **catch** (Exception exception) {
23. **throw**  **new** NotCorrectPolishNotationException("Not correct notation form!!!");
24. }
26. } **else** {
27. **throw** **new** NotCorrectPolishNotationException("Not correct notation form!!!");
28. }
29. }
30. **if** (!stack.empty()) {
31. **return** stack.pop();
32. } **else** {
33. **throw** **new** NotCorrectPolishNotationException("Not correct notation form!!!");
34. }
35. }
37. **private** **boolean** isArithmeticOperation(String string) {
38. **for** (ArithmeticOperation operation : arithmeticOperations) {
39. **if** (operation.getSymbol().equals(string))
40. **return** **true**;
41. }
42. **return** **false**;
43. }
45. **private** **boolean** isNumber(String string) {
46. **try** {
47. Double.parseDouble(string);
48. **return** **true**;
49. } **catch** (NumberFormatException exception) {
50. **return** **false**;
51. }
52. }
54. **private** **int** getIndexOfOperation(String operation) {
55. **for** (**int** i = 0; i < arithmeticOperations.length; i++) {
56. **if** (arithmeticOperations[i].getSymbol().equals(operation))
57. **return** i;
58. }
59. **return** 0;
60. }
62. **public** String[] readInputString() {
63. System.out.print("Введите выражение: ");
64. String string = input.nextLine();
65. **return** string.split(" ");
66. }
67. }