|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

**Отчет**

**по лабораторной работе № 3**

**Название:** ООП

**Дисциплина:** Языки программирования для работы с большими данными

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-21М |  |  | Д.Ю. Ермошин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

**Задание**

Поближе познакомиться с работой с классами, наследованием и полиморфизмом в Java.

**Выполнение работы**

**Вариант 1**

Определить класс Вектор размерности n.

Реализовать методы сложения, вычитания, умножения, инкремента, декремента, индексирования. Определить массив из m объектов.

Каждую из пар векторов передать в методы, возвращающие их скалярное произведение и длины. Вычислить и вывести углы между векторами.

Dot.java

|  |
| --- |
| package lab3;  public class Dot {  private final int x;  public Dot(int x) {  this.x = x;  }  public int getX() {  return x;  }  public Dot add(final Dot dot) {  return new Dot(x + dot.getX());  }  public Dot subtract(final Dot dot) {  return new Dot(x - dot.getX());  }  public Dot multiply(final Dot dot) {  return new Dot(x \* dot.getX());  }  public Dot divide(final Dot dot) {  return new Dot(x / dot.getX());  }  } |

Vector.java

|  |
| --- |
| package lab3;  import java.util.Arrays;  public class Vector {  private Dot[] vertexes;  public Vector(final int size) {  vertexes = new Dot[size];  }  public Vector(Dot... vertexes) {  this.vertexes = vertexes;  }  public Vector(int[] vertexes) {  this.vertexes = new Dot[vertexes.length];  for (int i = 0; i < vertexes.length; i++) {  this.vertexes[i] = new Dot(vertexes[i]);  }  }  public int getVectorLength() {  return vertexes.length;  }  public Vector add(Vector vec) {  Vector resultVector = new Vector(vec.vertexes.length);  for (int i = 0; i < vertexes.length; i++) {  resultVector.vertexes[i] = vertexes[i].add(vec.vertexes[i]);  }  return resultVector;  }  public Vector subtract(Vector vec) {  Vector resultVector = new Vector(vec.vertexes.length);  for (int i = 0; i < vertexes.length; i++) {  resultVector.vertexes[i] = vertexes[i].subtract(vec.vertexes[i]);  }  return resultVector;  }  public Vector multiply(Vector vec) {  final int vertexesAmount = Math.min(vertexes.length, vec.vertexes.length);  Vector resultVector = new Vector(vertexesAmount);  for (int i = 0; i < vertexesAmount; i++) {  resultVector.vertexes[i] = vertexes[i].multiply(vec.vertexes[i]);  }  return resultVector;  }  public Vector divide(Vector vec) {  Vector resultVector = new Vector(vec.vertexes.length);  for (int i = 0; i < vertexes.length; i++) {  resultVector.vertexes[i] = vertexes[i].divide(vec.vertexes[i]);  }  return resultVector;  }  public double getMagnitude() {  Vector tempVec = multiply(this);  double magnitude = 0d;  for (int i = 0; i < tempVec.getVectorLength(); i++) {  magnitude += tempVec.vertexes[i].getX();  }  return Math.sqrt(magnitude);  }  public static int getScalar(Vector vec1, Vector vec2) {  final int vertexesAmount = Math.min(vec1.getVectorLength(), vec2.getVectorLength());  int scalar = 0;  for (int i = 0; i < vertexesAmount; i++) {  scalar += vec1.vertexes[i].multiply(vec2.vertexes[i]).getX();  }  return scalar;  }  public static double getAngle(Vector from, Vector to) {  return Math.acos(Math.cos((double) Vector.getScalar(from, to) / (from.getMagnitude() \* to.getMagnitude())));  }  @Override  public String toString() {  return "Vector{" +  "vertexes=" + Arrays.toString(vertexes) +  '}';  }  } |

Variant1.java

|  |
| --- |
| package lab3;  import java.util.Scanner;  /\*\*  \* 1. Определить класс Вектор размерности n.  \* Реализовать методы сложения, вычитания, умножения, инкремента, декремента, индексирования.  \* Определить массив из m объектов.  \* Каждую из пар векторов передать в методы, возвращающие их скалярное произведение и длины.  \* Вычислить и вывести углы между векторами.  \*/  public class Variant1 {  public void work() {  final Scanner scanner = new Scanner(System.in);  System.out.println("Enter the number of dementions");  final int dementionNumber = scanner.nextInt();  System.out.println("Enter the number of vectors");  int amountOfVectors = scanner.nextInt();  final Vector[] vectors = new Vector[amountOfVectors];  for (int i = 0; i < amountOfVectors; i++) {  System.out.println(String.format("Enter the values of the vector [%d]", i + 1));  final int[] vertexes = new int[dementionNumber];  for (int j = 0; j < dementionNumber; j++) {  System.out.println(String.format("Enter [%d] value", j + 1));  vertexes[j] = scanner.nextInt();  }  vectors[i] = new Vector(vertexes);  }  amountOfVectors = amountOfVectors == 1 ? 1 : amountOfVectors / 2;  for (int i = 0; i < amountOfVectors; i += 2) {  System.out.println(String.format("Vector [%d] length: %.2f", i + 1, vectors[i].getMagnitude()));  if (i + 1 <= vectors[i].getVectorLength()) {  System.out.println(String.format("Vector [%d] length: %.2f", i + 2, vectors[i + 1].getMagnitude()));  System.out.println(String.format(  "Angle between vectors: %.2f",  Vector.getAngle(vectors[i], vectors[i + 1])  ));  }  }  }  } |

**Вариант 2**

Создать классы, спецификации которых приведены ниже.

Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль.

Car: id, Марка, Модель, Год выпуска, Цвет, Цена, Регистрационный номер.

Создать массив объектов.

Вывести: a) список автомобилей заданной марки;

b) список автомобилей заданной модели, которые эксплуатируются больше n лет;

c) список автомобилей заданного года выпуска, цена которых больше указанной.

Car.java

|  |
| --- |
| package lab3;  import java.time.LocalDate;  import java.util.ArrayList;  import java.util.List;  import java.util.concurrent.atomic.AtomicLong;  import java.util.stream.Collectors;  public class Car {  private static final AtomicLong idGenerator = new AtomicLong(0);  private final long id = idGenerator.incrementAndGet();  private String mark;  private String model;  private int manufactureYear;  private String color;  private long price;  private String registrationNumber;  public Car() {  }  public Car(String mark) {  this.mark = mark;  }  public Car(String model, int manufactureYear) {  this.model = model;  this.manufactureYear = manufactureYear;  }  public Car(int manufactureYear, long price) {  this.manufactureYear = manufactureYear;  this.price = price;  }  public Car(  String mark,  String model,  int manufactureYear,  String color,  long price,  String registrationNumber  ) {  this.mark = mark;  this.model = model;  this.manufactureYear = manufactureYear;  this.color = color;  this.price = price;  this.registrationNumber = registrationNumber;  }  public long getId() {  return id;  }  public String getMark() {  return mark;  }  public String getModel() {  return model;  }  public int getManufactureYear() {  return manufactureYear;  }  public String getColor() {  return color;  }  public long getPrice() {  return price;  }  public String getRegistrationNumber() {  return registrationNumber;  }  public void setMark(String mark) {  this.mark = mark;  }  public void setModel(String model) {  this.model = model;  }  public void setManufactureYear(int manufactureYear) {  this.manufactureYear = manufactureYear;  }  public void setColor(String color) {  this.color = color;  }  public void setPrice(long price) {  this.price = price;  }  public void setRegistrationNumber(String registrationNumber) {  this.registrationNumber = registrationNumber;  }  public static List<Car> produceCars(final int carsNumber, final String mark) {  List<Car> cars = new ArrayList<>(carsNumber);  for (int i = 0; i < carsNumber; i++) {  cars.add(new Car(mark));  }  return cars;  }  public static List<Car> produceCars(final int carsNumber, String model, int manufactureYear) {  List<Car> cars = new ArrayList<>(carsNumber);  for (int i = 0; i < carsNumber; i++) {  cars.add(new Car(model, manufactureYear));  }  return cars;  }  public static List<Car> produceCars(final int carsNumber, int manufactureYear, long price) {  List<Car> cars = new ArrayList<>(carsNumber);  for (int i = 0; i < carsNumber; i++) {  cars.add(new Car(manufactureYear, price));  }  return cars;  }  public static List<Car> findCarsByMark(  final List<Car> cars,  final String mark  ) {  return cars.stream()  .filter(car -> car.getMark().equals(mark))  .collect(Collectors.toList());  }  public static List<Car> findCarsByModelAndAge(  final List<Car> cars,  final String model,  final int age  ) {  return cars.stream()  .filter(car -> car.getModel().equals(model)  && LocalDate.now().getYear() - car.getManufactureYear() > age)  .collect(Collectors.toList());  }  public static List<Car> findCarsByManufacturedYearAndPrice(  final List<Car> cars,  final int manufactureYear,  final long price  ) {  return cars.stream()  .filter(car -> car.getManufactureYear() == manufactureYear  && car.getPrice() > price)  .collect(Collectors.toList());  }  } |

Variant2.java

|  |
| --- |
| package lab3;  import java.util.Collection;  import java.util.List;  import java.util.stream.Collectors;  import java.util.stream.Stream;  /\*\*  \* Создать классы, спецификации которых приведены ниже.  \* Определить конструкторы и методы setТип(), getТип(), toString().  \* Определить дополнительно методы в классе, создающем массив объектов.  \* Задать критерий выбора данных и вывести эти данные на консоль.  \* 8. Car: id, Марка, Модель, Год выпуска, Цвет, Цена, Регистрационный номер.  \* Создать массив объектов.  \* Вывести: a) список автомобилей заданной марки;  \* b) список автомобилей заданной модели, которые эксплуатируются больше n лет;  \* c) список автомобилей заданного года выпуска, цена которых больше указанной.  \*/  public class Variant2 {  public void work() {  List<Car> mercedesCars = Car.produceCars(5, "Mercedes");  List<Car> bmwCars = Car.produceCars(5, "BMW");  List<Car> audiCars = Car.produceCars(5, "Audi");  Car.findCarsByMark(squashLists(mercedesCars, bmwCars, audiCars), "BMW").forEach(System.out::println);  List<Car> x5Cars = Car.produceCars(5, "X5", 2010);  List<Car> x3Cars = Car.produceCars(5, "X3", 2015);  List<Car> x6NewCars = Car.produceCars(5, "X6", 2018);  List<Car> x6OldCars = Car.produceCars(5, "X6", 2013);  Car.findCarsByModelAndAge(squashLists(x5Cars, x3Cars, x6NewCars, x6OldCars), "X6", 5)  .forEach(System.out::println);  List<Car> a4Cars = Car.produceCars(5, 2010, 4000000);  List<Car> a3Cars = Car.produceCars(5, 2018, 2000000);  List<Car> a6Cars = Car.produceCars(5, 2010, 6000000);  Car.findCarsByManufacturedYearAndPrice(squashLists(a4Cars, a3Cars, a6Cars), 2010, 5000000).forEach(System.out::println);  }  @SafeVarargs  private final List<Car> squashLists(List<Car>... lists) {  return Stream.of(lists)  .flatMap(Collection::stream)  .collect(Collectors.toList());  }  } |

**Вариант 3**

Создать объект класса Дом, используя классы Окно, Дверь.

Методы: закрыть на ключ, вывести на консоль количество окон, дверей.

Door.java

|  |
| --- |
| package lab3;  import java.util.Objects;  public class Door {  private final int width;  private final int height;  private boolean isClosed;  public Door(int width, int height) {  this.width = width;  this.height = height;  this.isClosed = true;  }  public int getWidth() {  return width;  }  public int getHeight() {  return height;  }  public void close() {  isClosed = true;  }  public void open() {  isClosed = false;  }  @Override  public boolean equals(Object o) {  if (this == o)  return true;  if (!(o instanceof Door))  return false;  Door door = (Door) o;  return width == door.width &&  height == door.height;  }  @Override  public int hashCode() {  return Objects.hash(width, height);  }  } |

Window.java

|  |
| --- |
| package lab3;  import java.util.Objects;  public class Window {  private final int width;  private final int height;  public Window(int width, int height) {  this.width = width;  this.height = height;  }  public int getWidth() {  return width;  }  public int getHeight() {  return height;  }  @Override  public boolean equals(Object o) {  if (this == o)  return true;  if (!(o instanceof Window))  return false;  Window window = (Window) o;  return width == window.width &&  height == window.height;  }  @Override  public int hashCode() {  return Objects.hash(width, height);  }  } |

House.java

|  |
| --- |
| package lab3;  import java.util.List;  public class House {  private final Door entranceDoor;  private final List<Door> doors;  private final List<Window> windows;  public House(Door entranceDoor, List<Door> doors, List<Window> windows) {  this.entranceDoor = entranceDoor;  this.doors = doors;  this.windows = windows;  }  public void closeEntanceDoor() {  entranceDoor.close();  }  public void openEntanceDoor() {  entranceDoor.open();  }  public void closeInternalDoor(final int doorNumber) {  if (doorNumber > doors.size())  throw new IllegalArgumentException("Wrong door number");  doors.get(doorNumber).close();  }  public void openInternalDoor(final int doorNumber) {  if (doorNumber > doors.size())  throw new IllegalArgumentException("Wrong door number");  doors.get(doorNumber).open();  }  public int getDoorsNumber() {  return doors.size() + 1;  }  public int getWindowsNumber() {  return windows.size();  }  } |

Variant3.java

|  |
| --- |
| package lab3;  import java.util.ArrayList;  import java.util.List;  /\*\*  \* 5. Создать объект класса Дом, используя классы Окно, Дверь.  \* Методы: закрыть на ключ, вывести на консоль количество окон, дверей.  \*/  public class Variant3 {  public void work() {  final Door entranceDoor = new Door(80, 230);  final List<Door> internalDoors = new ArrayList<>();  for (int i = 0; i < 10; i++) {  internalDoors.add(new Door(80, 230));  }  final List<Window> windows = new ArrayList<>();  for (int i = 0; i < 5; i++) {  windows.add(new Window(50, 50));  }  final House house = new House(entranceDoor, internalDoors, windows);  house.openEntanceDoor();  house.closeEntanceDoor();  System.out.println("Doors number is " + house.getDoorsNumber());  System.out.println("Windows number is " + house.getWindowsNumber());  }  } |

**Вариант 4**

Система Платежи.

Клиент имеет Счет в банке и Кредитную Карту (КК).

Клиент может оплатить Заказ, сделать платеж на другой Счет, заблокировать КК и аннулировать Счет. Администратор может заблокировать КК за превышение кредита.

Account.java

|  |
| --- |
| package lab3;  public class Account {  private final RuntimeException CANCELLED\_ACCOUNT\_EXCEPTION = new IllegalStateException("Account is cancelled");  private long amount;  private boolean isCancelled;  public Account(long amount) {  this.amount = amount;  this.isCancelled = false;  }  public long addMoneys(final long money) {  if (isCancelled)  throw CANCELLED\_ACCOUNT\_EXCEPTION;  amount = amount + money;  return amount;  }  public long subtractMoneys(final long money) {  if (isCancelled)  throw CANCELLED\_ACCOUNT\_EXCEPTION;  amount = amount - money;  return amount;  }  public void cancel() {  isCancelled = true;  }  public void activate() {  isCancelled = false;  }  public long getAmount() {  return amount;  }  } |

CreditCard.java

|  |
| --- |
| package lab3;  public class CreditCard {  private final RuntimeException LOCKED\_CARD\_EXCEPTION = new IllegalStateException("Credit card is locked");  private final Account account;  private boolean isLocked;  public CreditCard(Account account) {  this.account = account;  isLocked = false;  }  public void lock() {  isLocked = true;  }  public void unlock() {  isLocked = false;  }  public long putMoneys(final long money) {  if (isLocked)  throw LOCKED\_CARD\_EXCEPTION;  return account.addMoneys(money);  }  public long pay(final Order order) {  if (isLocked)  throw LOCKED\_CARD\_EXCEPTION;  return account.subtractMoneys(order.getPrice());  }  public long transfer(final Account account, final long money) {  if (isLocked)  throw LOCKED\_CARD\_EXCEPTION;  account.addMoneys(money);  return account.subtractMoneys(money);  }  public Account getAccount() {  return account;  }  public boolean isLocked() {  return isLocked;  }  } |

Client.java

|  |
| --- |
| package lab3;  public class Client {  private final CreditCard card;  public Client() {  final Account account = new Account(0);  this.card = new CreditCard(account);  }  public Client(CreditCard card) {  this.card = card;  }  public long putMoneys(final long moneys) {  return card.putMoneys(moneys);  }  public long pay(final Order order) {  return card.pay(order);  }  public long transfer(final Account account, final long money) {  return card.transfer(account, money);  }  public CreditCard getCard() {  return card;  }  } |

Administrator.java

|  |
| --- |
| package lab3;  import java.util.List;  public class Administrator {  private final List<Client> customers;  public Administrator(List<Client> customers) {  this.customers = customers;  }  public void blockBadCustomers() {  customers.forEach(customer -> {  if (customer.getCard().getAccount().getAmount() < 0)  customer.getCard().lock();  });  }  } |

Variant4.java

|  |
| --- |
| package lab3;  import java.util.Collections;  /\*\*  \* 2. Система Платежи.  \* Клиент имеет Счет в банке и Кредитную Карту (КК).  \* Клиент может оплатить Заказ, сделать платеж на другой Счет, заблокировать КК и аннулировать Счет.  \* Администратор может заблокировать КК за превышение кредита.  \*/  public class Variant4 {  public void work() {  final Client client = new Client();  final Administrator administrator = new Administrator(Collections.singletonList(client));  client.putMoneys(1000000);  client.pay(new Order(100));  client.transfer(new Account(1000), 100);  client.pay(new Order(100000000000L));  administrator.blockBadCustomers();  }  } |

**Заключение**

В данной лабораторной работе мы ближе познакомились с работой с классами, наследованием и полиморфизмом в Java.