|  |  |
| --- | --- |
| 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 №** | 6 |

**Название:** Коллекции

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

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

Москва, 2020

**Вариант 1**

1. Определить множество на основе множества целых чисел. Создать методы для определения пересечения и объединения множеств.

Set

|  |
| --- |
| import java.util.Arrays;  import java.util.List;  import java.util.stream.Collectors;  import java.util.stream.Stream;  public class Set {  private List<Integer> elements;  public Set(Integer... elements) {  this(Arrays.asList(elements));  }  private Set(List<Integer> elements) {  this.elements = elements;  }  public Set intersect(Set other) {  return new Set(elements.stream().filter(other.elements::contains).collect(Collectors.toList()));  }  public Set unify(Set other) {  return new Set(Stream.concat(this.elements.stream(), other.elements.stream()).distinct().collect(Collectors.toList()));  }  @Override  public String toString() {  return "Элементы = " + elements;  }  } |

|  |
| --- |
| public class LR6\_V1 {  public static void main(String[] args) {  Set set1 = new Set(1, 2, 3, 8, 6);  Set set2 = new Set(2, 3, 4, 5, 7, 6);  System.out.println("Пересечение: " + set1.intersect(set2));  System.out.println("Объединение: " + set1.unify(set2));  }  } |

**Вариант 2**

2. Реализовать класс, моделирующий работу N-местной автостоянки. Машина подъезжает к определенному месту и едет вправо, пока не встретится свободное место. Класс должен поддерживать методы, обслуживающие приезд и отъезд машины.

Car

|  |
| --- |
| public class Car {  String nameCar;  String numberOfCar;  public Car(String nameCar, String numberOfCar) {  this.nameCar = nameCar;  this.numberOfCar = numberOfCar;  }  // get'эры  public String getNameCar() {  return nameCar;  }  public String getNumberOfCar() {  return numberOfCar;  }  // set'эры  public void setNameCar(String nameCar) {  this.nameCar = nameCar;  }  public void setNumberOfCar(String numberOfCar) {  this.numberOfCar = numberOfCar;  }  } |

Parking

|  |
| --- |
| import java.util.ArrayList;  public class Parking {  String nameParking; // Название парковки  int numberOfPlaces; // Колличество мест на парковке  private ArrayList<Car> carParking = new ArrayList<Car>(); // Список припаркованных машин  public Parking(String nameParking, int numberOfPlaces) {  this.nameParking = nameParking;  this.numberOfPlaces = numberOfPlaces;  for (int i=0; i<numberOfPlaces;i++){  this.carParking.add(null);  }  System.out.println("Парковка свободна!");  }  // get'эры  public String getNameParking() {  return nameParking;  }  public int getNumberOfPlaces() {  return numberOfPlaces;  }  public ArrayList<Car> getCarParking() {  return carParking;  }  // set'эры  public void setNameParking(String nameParking) {  this.nameParking = nameParking;  }  public void setNumberOfPlaces(int numberOfPlaces) {  this.numberOfPlaces = numberOfPlaces;  }  public void setCarParking(ArrayList<Car> carParking) {  this.carParking = carParking;  }  // Добавление новой машины на парковку  public void addCarOnParking(Car car) {  boolean flag = false;  for (int i=0; i<this.numberOfPlaces; i++){  if (carParking.get(i) == null) {  carParking.set(i, car);  System.out.println("Машина припаркована.");  flag = true;  break;  }  }  if (!flag) {  System.out.println("Мест нет!");  }  // System.out.println(carParking.toString());  }  // Освобождение места  public void removeCarOnParking (Car car) {  try {  int carParkingIndex = carParking.indexOf(car);  carParking.set(carParkingIndex, null);  System.out.println("Место свободно!");  } catch (IndexOutOfBoundsException e) {  System.out.println("Такая машина не припаркована");  }  }  // Мониторинг свободных мест  public void printFreePlace() {  System.out.println("Количество мест на парковке: " + this.getNumberOfPlaces());  for (int i = 0; i < carParking.size(); i++) {  if (carParking.get(i) == null) {  System.out.println("Место " + i + " свободно");  } else  System.out.println("Место " + i + " занято");  }  }  } |

|  |
| --- |
| public class LR6\_V2 {  public static void main(String[] args) {  Car bmw = new Car("BMW", "к123нр777");  Car mercedes = new Car("Mercedes", "а456тс777");  Car smart = new Car("Smart", "х789еу777");  Car volkswagen = new Car("VolksWagen", "м000мм777");  Car lexus = new Car("Lexus", "е777кх777");  Parking parking = new Parking("Подземный паркинг 1", 4);  System.out.println("Припаркуем 4 машины на парковке:");  parking.addCarOnParking(bmw);  parking.addCarOnParking(mercedes);  parking.addCarOnParking(smart);  parking.addCarOnParking(volkswagen);  System.out.println("Пробуем припарковать пятую:");  parking.addCarOnParking(lexus);  System.out.println("Две машины уехали с парковки:");  parking.removeCarOnParking(mercedes);  parking.removeCarOnParking(smart);  System.out.println("Повторно паркуем пятую машину:");  parking.addCarOnParking(lexus);  System.out.println("Проверяем, есть ли свободные места:");  parking.printFreePlace();  }  } |