Задание 1

Реализовать простой класс. Требования к выполнению

- Реализовать пользовательский класс по варианту.
- Создать другой класс с методом main, в котором будут находится примеры использования пользовательского класса.

Для каждого класса

- Создать поля классов
- Создать методы классов
- Добавьте необходимые get и set методы (по необходимости)
- Укажите соответствующие модификаторы видимости
- Добавьте конструкторы
- Переопределить методы toString() и equals()

7) Множество символов ограниченной мощности – Предусмотреть возможность объединения двух множеств, вывода на печать элементов множества, а так же метод, определяющий, принадлежит ли указанное значение множеству. Класс должен содержать методы, позволяющие добавлять и удалять элемент в/из множества. Конструктор должен позволить создавать объекты с начальной инициализацией. Мощность множества задается при создании объекта. Реализацию множества осуществить на базе одномерного массива. Реализовать метод equals, выполняющий сравнение объектов данного типа.

Код программы

Task1.java

```
public class task1 {

  public static void Checker(myset s, Object o){
    if(s.has(o))
      System.out.println("Set includes " + o);
    else
      System.out.println("Set does not include " + o);
  }
  public static void PrintSet(String name, myset set){
    System.out.println(name);
    set.print();
  }
  public static void main(String[] args) {
    myset arr = new myset(1,2,3,4,5);
    myset arr2 = new myset(1,6,7,4,5);
    arr.add(10);
    PrintSet("First set", arr);
    Checker(arr, 2);
    arr.delete(2);
    Checker(arr, 2);
    PrintSet("First set", arr);
    System.out.println("Set power is " + arr.size());
    arr.join(arr2);
    PrintSet("First set", arr);
```

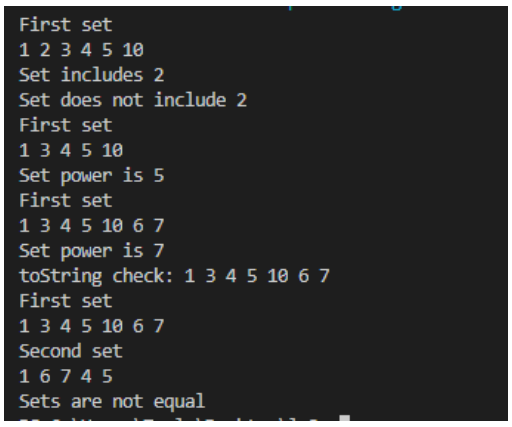
```
        System.out.println("Set power is " + arr.size());
        System.out.println("toString check: " + arr.toString());
        PrintSet("First set", arr);
        PrintSet("Second set", arr2);
        System.out.print("Sets are " + (arr.equals(arr2) ? "equal" : "not equal"));
    }
}
```

Myset.java

```
import java.lang.reflect.Array;
public class myset<T> {
    private T[] mset;
    private int size;
    public myset(T ... elements){
        size = elements.length;
        mset = (T[]) new Object[size];
        for(int i = 0; i<size; i++)
            mset[i] = elements[i];
    }

    public T[] getElements(){
        return mset;
    }

    public boolean has(Object element){
        for(T i : mset)
            if(element.equals(i))
                return true;
        return false;
    }

    public void add(Object element){
        if(has(element))
            return;
        T[] temp = (T[]) new Object[size+1];
        for(int i = 0; i<size; i++){
            temp[i] = mset[i];
        }
        temp[size] = (T) element;
        size++;
        mset = temp;
    }
    public myset join(myset secondSet){
        for(Object i : secondSet.getElements())
            if(!has(i))
                add(i);
        return this;
    }
    public void delete(Object element){
        if(has(element)){
```

```
            T[] temp = (T[]) new Object[size-1];
            for(int i = 0, j = 0; i<size; i++, j++){
               if(!mset[i].equals(element))
                  temp[j] = mset[i];
               else
                  j--;
            }
            size--;
            mset = temp;
         }
      }
      public boolean equals(myset secondSet){
         if(this.size != secondSet.size)
            return false;
         for(T i : mset){
            if(secondSet.has(i) == false){
               return false;
            }
         }
         return true;
      }
      public void print(){
         for(T i : mset){
            System.out.print(i + " ");
         }
         System.out.println();
      }
      public String toString(){
         String str = "";
         for(T elem : mset){
            str += elem.toString() + " ";
         }
         return str;
      }
      public int size(){
         return size;
      }
   }
```

Вывод программы



Задание 2

Разработать автоматизированную систему на основе некоторой структуры дан- ных, манипулирующей объектами пользовательского класса. Реализовать требуемые функции обработки данных

Требования к выполнению

- Задание посвящено написанию классов, решающих определенную задачу автоматизации;
- Данные для программы загружаются из файла (формат произволен). Файл создать и написать вручную.

Система оповещений на дорожном вокзале Автоматизированная информационная система на железнодорожном вокзале содержит сведения об отправлении поездов дальнего следования. Составить программу, которая должна хранить расписание поездов в структурированном, отсортированном по времени отправления виде (используя бинарное дерево).

- Обеспечивает первоначальный ввод данных в информационную систему о текущем расписании из файла и формирование дерева;
- Печатает все расписание на экран по команде;
- Выводит информацию о поезде по номеру поезда;
- По названию станции назначения выводит данные обо всех поездах, которые следуют до этой станции;
- Список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа;
- Список поездов, отправляющихся до заданного пункта назначения и имеющих общие места;
- За 10, 5, 3 минуты до отправления поезда показывает информационное сообщение об отправлении поезда.

Код программы

Main.java

```
import java.io.*;

public class Main {
  public static void main(String[] args) throws IOException{
    Manager manager = new Manager();
    manager.loadFromJson("Manager.json");
    manager.showTrainsInfo("First Station");
    manager.showTrainsInfAfterTime("Third Station", "Fifth Station", "05:00");
    manager.notificate("Second Station", "05:00");
    manager.showCapacitiesTo("Fourth Station");
    manager.showTimeTable();
  }
}
```

Manager.java

```
import java.util.*;
import java.io.*;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import java.text.ParseException;
```

```
public class Manager{
  private Vector<Station> stations;
  private Vector<Train> trains;

  public Manager(){
    stations = new Vector<>();
    trains = new Vector<>();
  }

  public void loadFromJson(String path){
    String str = "";
    try{
      FileReader r = new FileReader(path);
      BufferedReader reader = new BufferedReader(r);
      String temp = reader.readLine();
      while(temp != null){
        str += temp;
        temp = reader.readLine();
      }
      Gson gson = new Gson();
      Manager m = gson.fromJson(str, Manager.class);
      stations = m.stations;
      trains = m.trains;
    }
    catch (FileNotFoundException ex){
        System.out.println(ex);
    }
    catch (IOException ex){
       System.out.println(ex);
    }
  }

  public void addStation(Station st){
    if(!stations.contains(st)){
      stations.addElement(st);
      System.out.println("Station successfully added.");
    } else {
      System.out.println("Station exists in manager.");
    }
  }

  public void showTimeTable(){
    System.out.println();
    System.out.println("Timetable");
    for(Station st : stations){
      st.showTimeTable();
      System.out.println();
    }
  }
```

```
 public Vector<Train> getTrains(){
  return trains;
 }

 public void showTrainsInfo(String station){
  System.out.println();
  System.out.println("Train's info on station \'" + station + "\'");
  for(Station st : stations){
   if(st.getName().equals(station)){
    Vector<String> t = st.getTrains();
    if(t.size() < 1){
     System.out.println("No trains info on this station");
     return;
    }
    for(Train train : trains){
     if(t.contains(train.getName())){
      System.out.println("Train name: \'" + train.getName() + "'. Capacity: " + train.getCapacity());
     }
    }
    return;
   }
  }
  System.out.println("Station not found");
 }

 public void showTrainsInfAfterTime(String fromStation, String toStation, String afterTime){
  if(fromStation.equals(toStation)){
   System.out.println("Wron input parametrs");
   return;
  }
  Station firstStation = stations.elementAt(0), secondStation = stations.elementAt(0);
  for(Station st : stations){
   if(st.getName().equals(fromStation)){
    firstStation = st;
   }
   if(st.getName().equals(toStation)){
    secondStation = st;
   }
  }
  Vector<String> first_trains = firstStation.getTrainsAfter(afterTime);
  Vector<String> second_trains = secondStation.getTrainsAfter(afterTime);

  System.out.println();
  System.out.println("Trains leaving after " + afterTime + " from \'" + fromStation + "\' to \'" + toStation
+ "\'");
  for(String train : first_trains){
   if(second_trains.contains(train)){
    System.out.println(train + " ");
   }
```

```
    }
  }
  public void notificate(String station, String current_time){
    System.out.println();
    System.out.println("Notification");
    for(Station st : stations){
      if(st.getName().equals(station)){
        try{
          st.notificate(current_time);
        } catch(ParseException ex){}
      }
    }
  }
  public void showCapacitiesTo(String station){
    int cap = 0;
    for(Station st : stations){
      if(st.getName().equals(station)){
        Vector<String> trs = st.getTrains();
        for(Train train : trains){
          if(trs.contains(train.getName())){
            cap += train.getCapacity();
          }
        }
      }
    }
    System.out.println();
    System.out.println("Summed train\'s capacity on \'" + station + "\' is " +  cap);
  }
}
```

Station.java

```
import java.util.HashMap;
import java.util.HashSet;
import java.util.TreeSet;
import java.util.Map;
import java.util.*;
import java.text.SimpleDateFormat;
import java.text.ParseException;

public class Station {
  private String name;
  private String next;
  private HashMap<String, TreeSet<String>> train_table;

  public Station(String _name, String _next){
     name = _name;
     next = _next;
     train_table = new HashMap<>();
  }
  public void notificate(String current_time) throws ParseException{
```

```
  boolean isAnyAnnounce = false;
  SimpleDateFormat parser = new SimpleDateFormat("HH:mm");
  Date current = parser.parse(current_time);
  for(Map.Entry<String, TreeSet<String>> entry : train_table.entrySet()){
    for(String time : entry.getValue()){
      Date date = parser.parse(time);
      if( date.getTime() - current.getTime() <= 180000){
        System.out.println("Train \'" + entry.getKey() + "' departs in less than 3 minutes");
        isAnyAnnounce = true;
        break;
      } else if( date.getTime() - current.getTime() <= 300000){
        System.out.println("Train \'" + entry.getKey() + "' departs in less than 5 minutes");
        isAnyAnnounce = true;
        break;
      }
      if( date.getTime() - current.getTime() <= 600000){
        System.out.println("Train \'" + entry.getKey() + "' departs in less than 10 minutes");
        isAnyAnnounce = true;
        break;
      }
    }
  }
  if(!isAnyAnnounce){
    System.out.println("There are no departures in the next 10 minutes");
  }
}
public Vector<String> getTrainsAfter(String afterTime){
  Vector<String> trains = new Vector<>();
  for(Map.Entry<String, TreeSet<String>> entry : train_table.entrySet())
    for(String time : entry.getValue())
      for(int i = 0; i<5; i++)
        if(time.charAt(i) > afterTime.charAt(i)){
          if(!trains.contains(entry.getKey()))
            trains.addElement(entry.getKey());
          break;
        }
  return trains;
}
public void addTrain(Train train){
    train_table.put(train.getName(), new TreeSet<String>());
}
public void addTime(Train train, String time){
    train_table.get(train.getName()).add(time);
}
public void showTimeTable(){
  System.out.println("Station name: \"" + name + "\"\nNext Station: \"" + next + "\"");
  for(Map.Entry<String, TreeSet<String>> entry : train_table.entrySet()){
    System.out.println("Train: " + entry.getKey());
    for(String time : entry.getValue()){
      System.out.print(time + " ");
```

```
    }
    System.out.println();
   }
  }
  public String getName(){
   return name;
  }
  public Vector<String> getTrains(){
   Vector<String> trains = new Vector<>();
   for(Map.Entry<String, TreeSet<String>> entry : train_table.entrySet()){
    trains.addElement(entry.getKey());
   }
   return trains;
  }
}
```

Train.java

```
public class Train {
  private int capacity;
  private String name;
  public Train(String _name, int _capacity){
   name = _name;
   capacity = _capacity;
  }
  public String getName(){
   return name;
  }
  public int getCapacity(){
   return capacity;
  }
}
```

Manager.json


```
{"stations":[{"name":"First Station","next":"Second
Station","train_table":{"101":["07:00","10:00","13:00","16:00"],"102":["08:00","10:00","12:00","14:00",
"16:00","18:00"],"103":["07:00","12:30","17:00","22:00"]}},{"name":"Second Station","next":"Third
Station","train_table":{"102":["09:30","11:30","13:30","15:30","17:30","19:30"],"103":["10:00","10:30",
"12:00","20:00"],"105":["10:00","10:30","12:00","20:00"]}},{"name":"Third Station","next":"Fourth
Station","train_table":{"101":["10:00","13:00","16:00","19:00"],"103":["00:00","09:00","14:30","19:00"]
,"105":["10:00","11:00","12:00","20:00"]}},{"name":"Fourth Station","next":"Fifth
Station","train_table":{"101":["13:00","16:00","19:00","22:00"],"102":["12:30","14:30","16:30","18:30",
"20:30","23:30"],"104":["01:00","10:00","15:00","20:00"],"105":["11:00","12:00","13:00","21:00"]}},{"n
ame":"Fifth Station","next":"First
Station","train_table":{"102":["02:00","14:00","16:00","18:00","20:00","22:00"],"103":["02:00","11:00",
"16:30","21:00"],"104":["01:00","06:00","15:00","20:00"],"105":["12:00","13:00","14:00","22:00"]}}],"tr
ains":[{"capacity":170,"name":"101"},{"capacity":120,"name":"102"},{"capacity":80,"name":"103"},{"cap
acity":190,"name":"104"},{"capacity":130,"name":"105"}]}
```

Вывод программы

```
Train's info on station 'First Station'
Train name: '101'. Capacity: 170
Train name: '102'. Capacity: 120
Train name: '103'. Capacity: 80

Trains leaving after 05:00 from 'Third Station' to 'Fifth Station'
103
105

Notification
There are no departures in the next 10 minutes

Summed train's capacity on 'Fourth Station' is 610

Timetable
Station name: "First Station"
Next Station: "Second Station"
Train: 101
07:00 10:00 13:00 16:00
Train: 102
08:00 10:00 12:00 14:00 16:00 18:00
Train: 103
07:00 12:30 17:00 22:00
Timetable
Station name: "First Station"
Next Station: "Second Station"
Train: 101
07:00 10:00 13:00 16:00
Train: 102
08:00 10:00 12:00 14:00 16:00 18:00
Train: 103
07:00 12:30 17:00 22:00

Station name: "Second Station"
Next Station: "Third Station"
Train: 102
09:30 11:30 13:30 15:30 17:30 19:30
Train: 103
10:00 10:30 12:00 20:00
Train: 105
10:00 10:30 12:00 20:00

Station name: "Third Station"
Next Station: "Fourth Station"
Train: 101
10:00 13:00 16:00 19:00
Train: 103
00:00 09:00 14:30 19:00
Train: 105
10:00 11:00 12:00 20:00

Station name: "Fourth Station"
Next Station: "Fifth Station"
Train: 101
13:00 16:00 19:00 22:00
Train: 102
12:30 14:30 16:30 18:30 20:30 23:30
Train: 104
01:00 10:00 15:00 20:00
Train: 105
11:00 12:00 13:00 21:00

Station name: "Fifth Station"
Next Station: "First Station"
Train: 102
02:00 14:00 16:00 18:00 20:00 22:00
Train: 103
02:00 11:00 16:30 21:00
Train: 104
01:00 06:00 15:00 20:00
Train: 105
12:00 13:00 14:00 22:00
```

Вывод: Научился создавать и использовать классы в программах на языке Java