|  |  |
| --- | --- |
| 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**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 4 |

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

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

Москва, 2020

**Задание**

Задача 1.

Создать класс Cinema (кино) с внутренним классом, с помощью объектов которого можно хранить информацию об адресах кинотеатров, фильмах и времени сеансов.

Задача 2.

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов interface Врач <- class Хирург <- class Нейрохирург.

**Решение**:

Программа 1.

|  |  |
| --- | --- |
|  | public class Main {  public static void main(String[] args){ |
|  | //'Cinema' class testing. |
|  | Cinema cinema = new Cinema(); |
|  | showAddStatus(cinema.add("Фильм1", 2001, "Жанр1", |
|  | "Директор1", 100, 5000000)); |
|  | showAddStatus(cinema.add("Фильм2", 2002, "Жанр2", |
|  | "Директор2", 50, 1000000)); |
|  | showAddStatus(cinema.add("Фильм3", 2003, "Жанр3", |
|  | "Директор3", 75, 3000000)); |
|  |  |
|  | showDeleteStatus(cinema.delete()); |
|  | cinema.printAll(); |
|  | showDeleteStatus(cinema.delete()); |
|  | cinema.printAll(); |
|  | showDeleteStatus(cinema.delete()); |
|  | cinema.printAll(); |
|  |  |
|  | showAddStatus(cinema.add("Фильм4", 2004, "Жанр4", |
|  | "Директор4", 90, 3000000)); |
|  | showAddStatus(cinema.add("Фильм5", 2005, "Жанр5", |
|  | "Директор5", 120, 7000000)); |
|  |  |
|  | cinema.printAll(); |
|  | } |
|  |  |
|  | //Instead of writing "sout" each time. |
|  | static void showAddStatus(boolean value){ |
|  | System.out.println("Adding operation status: " + value); |
|  | } |
|  |  |
|  | //The same as in the previous comment. |
|  | static void showDeleteStatus(boolean value){ |
|  | System.out.println("Deleting operation status: " + value); |
|  | } |
|  | } |

|  |
| --- |
| public class Cinema { |
|  | private int size = 1; |
|  | private Movie[] array = new Movie[size]; |
|  | private int current = 0; |
|  |  |
|  |  |
|  | //Adds new items to the array. |
|  | public boolean add(String название, int год\_производства, String жанр, |
|  | String режиссер, int время\_показа, int бюджет){ |
|  | //Returns 'false' and quits the method without adding new element to the array, |
|  | // if there is at least one variable, which meets its condition. |
|  | if(название == null || год\_производства <= 0 || жанр == null |
|  | || режиссер == null || время\_показа <= 0 || бюджет <= 0){ return false; } |
|  |  |
|  | //Increases the array if there is no room for the new array. |
|  | if(!this.isEmpty()) { |
|  | this.extendArray(1); |
|  | ++this.current; |
|  | } |
|  |  |
|  | array[current] = new Movie(название, год\_производства, жанр, |
|  | режиссер, время\_показа, бюджет); |
|  |  |
|  | return true; |
|  | } |
|  |  |
|  |  |
|  | //Deletes the last array from the array. |
|  | public boolean delete(){ |
|  | return this.narrowArray(1); |
|  | } |
|  |  |
|  |  |
|  | //Displays all the array's elements information. |
|  | public void printAll(){ |
|  | int count = 0; |
|  |  |
|  | for(Movie temp : this.array){ |
|  | //Returns 'true' if array's cell contains 'null' value. |
|  | if(temp == null) { |
|  | System.out.println("No data has been found!"); |
|  | return; |
|  | } |
|  |  |
|  | System.out.println("Фильм #" + ++count + |
|  | ". \n=============================================================\n" + |
|  | "Название: " + temp.getTitle() + |
|  | "\nЖанр: " + temp.getGenre() + |
|  | "\nГод выпуска: " + temp.getGenre() + |
|  | "\nВремя показа: " + temp.getDuration() + "минут" + |
|  | "\nРежиссер: " + temp.getDirector() + |
|  | "\nБюджет: " + temp.getBudget() + " долл США" + |
|  | "\n=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-="); |
|  |  |
|  | //Prints new line when the array's last element will be used. |
|  | if(count == this.array.length - 1){ System.out.println(); } |
|  | } |
|  | } |
|  |  |
|  |  |
|  | //Returns 'true' value if the current index points on the empty array's cell. |
|  | private boolean isEmpty(){ |
|  | return this.array[current] == null; |
|  | } |
|  |  |
|  |  |
|  | //Increases the field array by the input size. Returns 'true' value if the increasing was permitted. |
|  | //Other wise - 'false'. |
|  | private boolean extendArray(int size){ |
|  | //Return 'false' value if the input size is less than or equal to zero value. |
|  | if(size <= 0){ return false; } |
|  |  |
|  | Movie[] temp = new Movie[this.array.length + size]; |
|  | System.arraycopy(this.array, 0, temp, 0, this.array.length); |
|  | array = temp; |
|  |  |
|  | return true; |
|  | } |
|  |  |
|  |  |
|  | //Decreases the field array by the input size. Returns 'true' value if the decreasing was permitted. |
|  | //Other wise - 'false'. |
|  | private boolean narrowArray(int size){ |
|  | //Return 'false' value if the input size is greater than or equal to the zero value |
|  | // or the current index points on the array's last element, which has 'null' value. |
|  | if(size <= 0 || this.array[this.current] == null) { return false; } |
|  |  |
|  | //Sets the first array's cell to 'null' value and decreases the current index by one |
|  | // if there is only one cell in the array. Otherwise, the if-statement will be skipped. |
|  | if(this.array.length == 1) { |
|  | this.array[current] = null; |
|  | return true; |
|  | } |
|  |  |
|  | Movie[] temp = new Movie[this.array.length - size]; |
|  | System.arraycopy(this.array, 0, temp, 0, temp.length); |
|  | --this.current; |
|  |  |
|  | array = temp; |
|  | return true; |
|  | } |
|  |  |
|  |  |
|  | //Inner class, created to hold all the information about the bought array. |
|  | private class Movie { |
|  | private String название; |
|  | private int год\_производства; |
|  | private String жанр; |
|  | private String режиссер; |
|  | private int время\_показа; |
|  | private int бюджет; |
|  |  |
|  | public Movie(String название, int год\_производства, String жанр, |
|  | String режиссер, int время\_показа, int бюджет) { |
|  | this.название = название; |
|  | this.год\_производства = год\_производства; |
|  | this.жанр = жанр; |
|  | this.режиссер = режиссер; |
|  | this.время\_показа = время\_показа; |
|  | this.бюджет = бюджет; |
|  | } |
|  |  |
|  | public String getTitle() { |
|  | return название; |
|  | } |
|  |  |
|  | public int getProductionYear() { |
|  | return год\_производства; |
|  | } |
|  |  |
|  | public String getGenre() { |
|  | return жанр; |
|  | } |
|  |  |
|  | public String getDirector() { |
|  | return режиссер; |
|  | } |
|  |  |
|  | public int getDuration() { |
|  | return время\_показа; |
|  | } |
|  |  |
|  | public int getBudget() { |
|  | return бюджет; |
|  | } |
|  | } |
|  | } |

Программа 2.

|  |
| --- |
| Import java.util.ArrayList; |
| import java.util.List; |  |
|  | public class Main { |
|  |  |
|  | public static void main(String[] args) { |
|  |  |
|  | Neurosurgeon neurosurgeon = new Neurosurgeon("Ringo Star"); |
|  |  |
|  | List<Surgeon> surgeons = new ArrayList<>(); |
|  | surgeons.add(neurosurgeon); |
|  |  |
|  | for (Surgeon surgeon: surgeons){ |
|  | surgeon.setDiagnos("Ann", 11); |
|  | surgeon.treat(); |
|  | ((Neurosurgeon)surgeon).sayName(); |
|  | System.out.println("---------------------------------"); |
|  | } |
|  |  |
|  | } |
|  |  |
|  | } |

|  |  |
| --- | --- |
| public interface Doctor { | |
|  | public void setDiagnos(String patientName, int analysis); |
|  | public void treat(); |
|  |  |
|  | } |

|  |  |
| --- | --- |
| public class Neurosurgeon extends Surgeon { | |
|  | |
|  | public Neurosurgeon(String name) { |
|  | super(name); |
|  |  |
|  | } |
|  |  |
|  | public void sayName(){ |
|  | System.out.println("her doctor " + name); |
|  | } |
|  |  |
|  | @Override |
|  | public void treat() { |
|  | if (diagnos.equals("Ill")) { |
|  | System.out.println(patientName + " stay in hospital"); |
|  | } else { |
|  | System.out.println(patientName + " go home"); |
|  | } |
|  | } |
|  | }   |  |  | | --- | --- | | public abstract class Surgeon implements Doctor { | | |  | protected String name; | |  |  | |  | public Surgeon(String name) { | |  | this.name = name; | |  | } | |  |  | |  | protected String diagnos; | |  | protected String patientName; | |  |  | |  |  | |  |  | |  | @Override | |  | public void setDiagnos(String patientName, int analysis) { | |  | this.patientName = patientName; | |  | if (analysis > 5) { | |  | diagnos = "Ill"; | |  | } else | |  | diagnos = "Well"; | |  | } | |  |  | |  | @Override | |  | public void treat() { | |  | if (diagnos.equals("ill")) { | |  | System.out.println(patientName + "stay in hospital"); | |  | } else { | |  | System.out.println(patientName + " go home"); | |  | } | |  | } | |  |  | |  |  | |  | public String getName() { | |  | return name; | |  | } | |  |  | |  | public void setName(String name) { | |  | this.name = name; | |  | } | |  |  | |  | public String getDiagnos() { | |  | return diagnos; | |  | } | |  |  | |  | } | |

**Вывод:**

Задание выполнено, программы компилируются и успешно выполняются.