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

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

**Отчет**

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

**Название:** Внутренние классы, интерфейсы

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  |  | С.В. Астахов | |
|  | (Группа) |  | (Подпись, дата) | | (И.О. Фамилия) |
|  |  |  |  | |  |
| Преподаватель |  |  |  | | П.В. Степанов |
|  |  |  | (Подпись, дата) | | (И.О. Фамилия) |

Москва, 2024

**Цель работы:** освоить базовые принципы работы с внутренними классами и интерфейсами на языке Java.

**Вариант: 1.**

**Задание 1:** Создать класс City (город) с внутренним классом, с помощью объектов которого можно хранить информацию о проспектах, улицах, площадях.

Код решения приведен в листинге 1.

Листинг 1 — реализация решения

|  |
| --- |
| package org.example;  import java.util.ArrayList;  public class Main {  public static void main(String[] args) {  System.out.println("=== A program about City with inner classes ===\n");  City city = new City("Reutov");  city.add("street", "Lenina");  city.add("prospekt", "50 years of Victory");  city.add("avenue", "Enthusiasts");  city.add("avenue", "Green");  System.out.println(city.toString());  }  }  class City {  private ArrayList<TopographyObject> content = new ArrayList<>();  private String name;  public City(String name) {  this.name = name;  }  public void add(String type, String name) {  TopographyObject t = new TopographyObject(type, name);  content.add(t);  }  public String toString() {  String s = String.format("<city> %s\n\n", name);  for(TopographyObject x : content) {  s = String.format("%s%s\n", s, x.toString());  }  return s;  }  private class TopographyObject {  private final String type;  private final String name;  public TopographyObject(String type, String name) {  this.type = type;  this.name = name;  }  public String toString() {  return String.format("<%s> %s", type, name);  }  }  } |

Результат исполнения программы приведен в листинге 2.

Листинг 2 — результат исполнения программы

|  |
| --- |
| === A program about City with inner classes ===  <city> Reutov  <street> Lenina  <prospekt> 50 years of Victory  <avenue> Enthusiasts  <avenue> Green |

**Задание 2:** Создать класс CD (mp3-диск) с внутренним классом, с помощью объектов которого можно хранить информацию о каталогах, подкаталогах и записях.

Код решения приведен в листинге 3.

Листинг 3 — реализация решения

|  |
| --- |
| package org.example;  import java.util.ArrayList;  public class Main {  public static void main(String[] args) throws Exception {  System.out.println("=== Mp3Disk class === ");  Mp3Disk mp3 = new Mp3Disk("Deep Purple");  mp3.getRoot().add("dir", ".trashbox");  mp3.getRoot().getChildDir(".trashbox").add("file", ".trojan.exe");  mp3.getRoot().add("dir", "top songs");  mp3.getRoot().getChildDir("top songs").add("file", "smoke on the water");  mp3.getRoot().getChildDir("top songs").add("file", "and fire in the sky");  mp3.getRoot().add("dir", ".autorun");  mp3.getRoot().getChildDir(".autorun").add("dir", "utils");  mp3.getRoot().getChildDir(".autorun").getChildDir("utils").add("file", "driver.exe");  System.out.println(mp3);  }  }  class Mp3Disk {  private FileSysObject root = new FileSysObject("dir", "root");  private String name;  public Mp3Disk(String name) {  this.name = name;  }  public FileSysObject getRoot() {  return root;  }  public String toString() {  return String.format("<Disk> %s\n%s", name, root.toStringLeveled(1));  }  class FileSysObject{  private ArrayList<FileSysObject> innerObjects = new ArrayList<>();  private String type;  private String name;  public FileSysObject(String type, String name) {  this.type = type;  this.name = name;  }  public void add(String type, String name) {  innerObjects.add(new FileSysObject(type, name));  }  public String toString() {  String s = "";  if(type.equals("file")){  s = String.format("<%s> %s", type, name);  }  else {  s = String.format("<%s> %s", type, name);  for(FileSysObject x: innerObjects){  s = String.format("%s\n\t%s", s, x.toString());  }  }  return s;  }  public String toStringLeveled(int lvl) {  String s = "";  if(type.equals("file")){  s = String.format("<%s> %s", type, name);  }  else {  s = String.format("<%s> %s", type, name);  for(FileSysObject x: innerObjects){  String tabs = "";  for(int i = 0; i < lvl; i++) {  tabs += "\t";  }  s = String.format("%s\n%s%s", s, tabs ,x.toStringLeveled(lvl + 1));  }  }  return s;  }  public FileSysObject getChildDir(String name) throws Exception {  for(FileSysObject x : innerObjects){  if (x.name.equals(name) && x.type.equals("dir")) {  return x;  }  }  throw new Exception("no such dir");  }  }  } |

Результат исполнения программы приведен в листинге 4.

Листинг 4 — результат исполнения программы

|  |
| --- |
| === Mp3Disk class ===  <Disk> Deep Purple  <dir> root  <dir> .trashbox  <file> .trojan.exe  <dir> top songs  <file> smoke on the water  <file> and fire in the sky  <dir> .autorun  <dir> utils  <file> driver.exe |

**Задание 3:** Абстрактный класс Книга (Шифр, Автор, Название, Год, Издательство). Подклассы Справочник и Энциклопедия.

Код решения приведен в листинге 5.

Листинг 5 — реализация решения

|  |
| --- |
| package org.example;  public class Main {  public static void main(String[] args) {  System.out.println("=== Classes of books ===\n\n");  Book book;  book = new Encyclopedia("1111-2222-3333", "Very clever Mr.", "Very smart book", 2001, "NYC-Book");  System.out.println(book);  book = new ReferenceBook("1111-2222-4444", "Another very clever Mr.", "Book about bees", 2005, "Spb-Book");  System.out.println(book);  }  }  abstract class Book {  protected String isbn;  protected String author;  protected String name;  protected Integer year;  protected String publisher;  public abstract String toString();  }  class Encyclopedia extends Book {  public Encyclopedia(String isbn, String author, String name, Integer year, String publisher) {  this.isbn = isbn;  this.author = author;  this.name = name;  this.year = year;  this.publisher = publisher;  }  @Override  public String toString() {  return String.format("<Encyclopedia> \nISBN: %s\nAuthor: %s\nCaption: %s\nYear: %d\nPublisher: %s\n\n", isbn, author, name, year, publisher);  }  }  class ReferenceBook extends Book {  public ReferenceBook(String isbn, String author, String name, Integer year, String publisher) {  this.isbn = isbn;  this.author = author;  this.name = name;  this.year = year;  this.publisher = publisher;  }  @Override  public String toString() {  return String.format("<ReferenceBook> \nISBN: %s\nAuthor: %s\nCaption: %s\nYear: %d\nPublisher: %s\n\n", isbn, author, name, year, publisher);  }  } |

Результат исполнения программы приведен в листинге 6.

Листинг 6 — результат исполнения программы

|  |
| --- |
| === Classes of books ===  <Encyclopedia>  ISBN: 1111-2222-3333  Author: Very clever Mr.  Caption: Very smart book  Year: 2001  Publisher: NYC-Book  <ReferenceBook>  ISBN: 1111-2222-4444  Author: Another very clever Mr.  Caption: Book about bees  Year: 2005  Publisher: Spb-Book |

**Задание 4:** Interface Абитуриент <- abstract class Студент <- class Студент-Заочник.

Код решения приведен в листинге 7.

Листинг 7 — реализация решения

|  |
| --- |
| package org.example;  public class Main {  public static void main(String[] args) {  System.out.println("=== Students class example ===\n\n");  Abiturient abiturient = new Zaochnik("Ivan", "IU", "IU-4");  System.out.println(abiturient.toString());  System.out.println("\n\n=== student after .changeCafedra(IU-6) ===\n\n");  abiturient.changeCafedra("IU-6");  System.out.println(abiturient.toString());  }  }  interface Abiturient{  abstract public String toString();  abstract public void changeFaculty(String s);  abstract public void changeCafedra(String s);  }  abstract class Student implements Abiturient{  protected String name;  protected String faculty;  protected String cafedra;  abstract public String toString();  abstract public void changeFaculty(String s);  abstract public void changeCafedra(String s);  }  class Zaochnik extends Student {  public Zaochnik(String name, String faculty, String cafedra){  this.name = name;  this.faculty = faculty;  this.cafedra = cafedra;  }  @Override  public void changeFaculty(String s) {  this.faculty = s;  }  @Override  public void changeCafedra(String s) {  this.cafedra = s;  }  @Override  public String toString() {  return String.format("<Zaochnik> %s %s %s", name, faculty, cafedra);  }  } |

Результат исполнения программы приведен в листинге 8.

Листинг 8 — результат исполнения программы

|  |
| --- |
| === Students class example ===  <Zaochnik> Ivan IU IU-4  === student after .changeCafedra(IU-6) ===  <Zaochnik> Ivan IU IU-6 |

**Вывод:** в ходе лабораторной работы были освоены базовые принципы работы с внутренними классами и интерфейсами на языке Java.