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

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 |  |  | Р.В. Баканов | |
|  | (Группа) |  | (Подпись, дата) | | (И.О. Фамилия) |
|  |  |  |  | |  |
| Преподаватель |  |  |  | | П.В. Степанов |
|  |  |  | (Подпись, дата) | | (И.О. Фамилия) |

Москва, 2024

**Цель работы:** освоить базовые принципы работы с интерфейсами в Java.

**Вариант: 2.**

**Задание 1:** Создать класс CD (mp3-диск) с внутренним классом, с помощью объектов которого можно хранить информацию о каталогах, подкаталогах и записях.

Код решения приведен в листингах 1 и 2.

Листинг 1 — Задание 1 (1)

public class Main {  
 public static void main(String[] args) {  
 var cd = new CD(  
 new CD.FileSystemNode(  
 "Root",  
 new CD.FileSystemNode[] {  
 new CD.FileSystemNode("Nature sounds", CD.FileSystemNode.Extension.MP3),  
 new CD.FileSystemNode("White noise", CD.FileSystemNode.Extension.MP3),  
 new CD.FileSystemNode(  
 "Metal",  
 new CD.FileSystemNode[] {  
 new CD.FileSystemNode("Life of Agony - Lost at 22", CD.FileSystemNode.Extension.MP3)  
 })  
 }  
 )  
 );  
 }  
}

Листинг 2 – Задание 1 (2)

public class CD {  
 public FileSystemNode rootNode;  
   
 public CD(FileSystemNode rootNode) {  
 this.rootNode = rootNode;  
 }  
   
 public static class FileSystemNode {  
 public String name;  
 public Extension extension;  
 public FileSystemNode[] childNodes;  
  
 public FileSystemNode(String name, FileSystemNode[] childNodes) {  
 this.name = name;  
 this.extension = Extension.*None*;  
 this.childNodes = childNodes;  
 }  
  
 public FileSystemNode(String name, Extension extension) {  
 this.name = name;  
 this.extension = extension;  
 this.childNodes = new FileSystemNode[] {};  
 }  
  
 public enum Extension {  
 *None*,  
 *MP3* }  
 }  
}

**Задание 2:** Создать класс Mobile с внутренним классом, с помощью объектов которого можно хранить информацию о моделях телефонов и их свойствах.

Код решения приведен в листингах 3 и 4.

Листинг 3 — Задание 2 (1)

public class Main {  
 public static void main(String[] args) {  
 var mobile = new Mobile(  
 "Samsung Galaxy S24",  
 new Mobile.Specifications(  
 "Android 14",  
 false,  
 true,  
 8,  
 3.39  
 )  
 );  
 }  
}

Листинг 4 – Задание 2 (2)

public class Mobile {  
 public String name;  
 public Specifications specifications;  
  
 public Mobile(String name, Specifications specifications) {  
 this.name = name;  
 this.specifications = specifications;  
 }  
  
 public static class Specifications {  
 public String operatingSystem;  
 public boolean hasCardSlot;  
 public boolean supportsNFC;  
 public int numberOfCPUCores;  
 public double cpuFrequency;  
  
 public Specifications(  
 String operatingSystem,  
 boolean hasCardSlot,  
 boolean supportsNFC,  
 int numberOfCPUCores,  
 double cpuFrequency  
 ) {  
 this.operatingSystem = operatingSystem;  
 this.hasCardSlot = hasCardSlot;  
 this.supportsNFC = supportsNFC;  
 this.numberOfCPUCores = numberOfCPUCores;  
 this.cpuFrequency = cpuFrequency;  
 }  
 }  
}

**Задание 3:** Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов: *interface Абитуриент <- abstract class Студент <- class Студент-Заочник*.

Код решения приведен в листинге 5.

Листинг 5 – Задание 3

public class Main {  
 public static void main(String[] args) {  
 Student partTimeStudent = new PartTimeStudent();  
 Student expelledStudent = new ExpelledStudent();  
  
 System.out.println("Part-time student is enrolled at university: " + partTimeStudent.isEnrolledAt());  
 System.out.println("Expelled student is enrolled at university: " + expelledStudent.isEnrolledAt());  
 }  
}  
  
interface IApplicant {  
 boolean isEnrolledAt();  
}  
  
abstract class Student implements IApplicant {  
 @Override  
 public boolean isEnrolledAt(){  
 return true;  
 }  
  
 public abstract String ConveyHomeworkResults() throws Exception;  
}  
  
class PartTimeStudent extends Student {  
  
 @Override  
 public String ConveyHomeworkResults() {  
 var usedMethod = "Homework results have been sent via e-mail";  
  
 return usedMethod;  
 }  
}  
  
class ExpelledStudent extends Student {  
 @Override  
 public boolean isEnrolledAt() {  
 return false;  
 }  
  
 @Override  
 public String ConveyHomeworkResults() throws Exception {  
 throw new Exception("Expelled student can not convey homework results");  
 }  
}

**Задание 4:** Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов: interface Сотрудник <- class Инженер <- class Руководитель.

Код решения приведен в листинге 6.

Листинг 6 — Задание 4

import java.util.ArrayList;  
import java.util.List;  
  
public class Main {  
 public static void main(String[] args) throws Exception {  
 IEmployee engineer = new Engineer();  
 engineer.assignTask(new Task());  
  
 try {  
 engineer.finishWorkingDay();  
 } catch (Exception e) {  
 System.out.println(e.getMessage());  
 }  
  
 IEmployee chief = new Chief(new Task[] { new Task() });  
  
 try {  
 chief.assignTask(new Task());  
 } catch (Exception e) {  
 System.out.println(e.getMessage());  
 }  
  
 chief.finishWorkingDay();  
 }  
}  
  
class Task {  
 boolean isDone = false;  
}  
  
interface IEmployee {  
 ArrayList<Task> assignedTasks = new ArrayList<>();  
 void finishWorkingDay() throws Exception;  
 void assignTask(Task task) throws Exception;  
}  
  
class Engineer implements IEmployee {  
 public Engineer() {  
 }  
  
 public Engineer(Task[] tasks) {  
 assignedTasks.addAll(List.of(tasks));  
 }  
  
 @Override  
 public void finishWorkingDay() throws Exception {  
 var someTasksNotDoneYet = false;  
  
 for (Task task : assignedTasks) {  
 if (!task.isDone) {  
 someTasksNotDoneYet = true;  
  
 break;  
 }  
 }  
  
 if (someTasksNotDoneYet) {  
 throw new Exception("Engineer can not finish working day until all tasks are done");  
 }  
  
 System.out.println("Engineer finished working day");  
 }  
  
 @Override  
 public void assignTask(Task task) throws Exception {  
 *assignedTasks*.add(task);  
 }  
}  
  
class Chief extends Engineer {  
 public Chief() {  
 }  
  
 public Chief(Task[] tasks) {  
 super(tasks);  
 }  
  
 @Override  
 public void finishWorkingDay() {  
 System.out.println("Chief finished working day");  
 }  
  
 @Override  
 public void assignTask(Task task) throws Exception {  
 throw new Exception("Chief can not be assigned with new tasks");  
 }  
  
 public void riseOwnSalary() {  
 System.out.println("Chief's salary has been risen");  
 }  
}

**Вывод:** в результате выполнения лабораторной работы были разработаны несколько программ в соответствие с требованиями задания.