**Отчет по лабораторной работе №2**

// Абстрактный класс Человек  
abstract class Person {  
 private String name;  
 private int age;  
 private String gender;  
  
 public Person() {  
 this.name = "Undefined";  
 this.age = 0;  
 this.gender = "Undefined";  
 }  
  
 public Person(String name, int age, String gender) {  
 this.name = name;  
 this.age = age;  
 this.gender = gender;  
 }  
  
 // Геттеры и сеттеры для полей  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 public String getGender() {  
 return gender;  
 }  
  
 public void setGender(String gender) {  
 this.gender = gender;  
 }  
  
 // Абстрактный метод, описывающий поведение объекта  
 public abstract void introduce();  
  
 // Метод описывающий общее поведение объекта  
 public void sleep() {  
 System.*out*.println(name + " is sleeping");  
 }  
}  
  
// Дочерний класс Студент  
class Student extends Person {  
 private int studentId;  
  
 private static int *studentCounter* = 0;  
  
 public Student() {  
 super();  
 this.studentId = ++*studentCounter*;  
 }  
  
 public Student(String name, int age, String gender) {  
 super(name, age, gender);  
 this.studentId = ++*studentCounter*;  
 }  
  
 public int getStudentId() {  
 return studentId;  
 }  
  
 // Переопределение метода introduce  
 @Override  
 public void introduce() {  
 System.*out*.println("I am a student named " + getName() + " with ID: " + studentId);  
 }  
  
 // Метод для доступа к studentCounter  
 public static int getStudentCounter() {  
 return *studentCounter*;  
 }  
}  
  
// Дочерний класс Преподаватель  
class Teacher extends Person {  
 private String subject;  
  
 public Teacher() {  
 super();  
 this.subject = "Undefined";  
 }  
  
 public Teacher(String name, int age, String gender, String subject) {  
 super(name, age, gender);  
 this.subject = subject;  
 }  
  
 public String getSubject() {  
 return subject;  
 }  
  
 // Переопределение метода introduce  
 @Override  
 public void introduce() {  
 System.*out*.println("I am a teacher named " + getName() + " teaching " + subject);  
 }  
  
 // Перегрузка метода introduce  
 public void introduce(String greeting) {  
 System.*out*.println(greeting + ", I am a teacher named " + getName() + " teaching " + subject);  
 }  
  
 // Переопределение метода sleep  
 @Override  
 public void sleep() {  
 System.*out*.println("The teacher " + getName() + " is taking a nap");  
 }  
}  
  
// Дочерний класс АссистентПреподавателя  
class TeachingAssistant extends Teacher {  
 private String assistantCourse;  
  
 public TeachingAssistant() {  
 super();  
 this.assistantCourse = "Undefined";  
 }  
  
 public TeachingAssistant(String name, int age, String gender, String subject, String assistantCourse) {  
 super(name, age, gender, subject);  
 this.assistantCourse = assistantCourse;  
 }  
  
 public String getAssistantCourse() {  
 return assistantCourse;  
 }  
  
 // Переопределение метода introduce  
 @Override  
 public void introduce() {  
 System.*out*.println("I am a teaching assistant named " + getName() + " assisting in " + getSubject());  
 }  
}  
  
// Пример использования классов  
public class Main {  
 public static void main(String[] args) {  
 Student student1 = new Student("Alice", 20, "Female");  
 Student student2 = new Student("Bob", 22, "Male");  
  
 Teacher teacher = new Teacher("Mr. Smith", 35, "Male", "Physics");  
 teacher.introduce();  
  
 TeachingAssistant assistant = new TeachingAssistant("Anna", 25, "Female", "Physics", "Lab Work");  
 assistant.introduce();  
  
 student1.introduce();  
 student2.introduce();  
  
 // Демонстрация статической переменной  
 System.*out*.println("Total students created: " + Student.*getStudentCounter*());  
 }  
  
}

**Абстрактный класс Person:**

**Поля:**

**name, age, gender - для хранения основной информации о человеке.**

**Методы:**

**getName(), getAge(), getGender() - для получения данных о человеке.**

**setName(), setAge(), setGender() - для установки новых значений полей.**

**introduce() (абстрактный) - предоставляет интерфейс для вывода информации о человеке.**

**sleep() - метод, описывающий общее поведение объекта.**

**Дочерний класс Student:**

**Дополнительные поля:**

**studentId - идентификатор студента.**

**studentCounter - статическая переменная для подсчёта созданных студентов.**

**Методы:**

**getStudentId() - для получения идентификатора студента.**

**introduce() - переопределение метода introduce() из класса Person.**

**getStudentCounter() - статический метод для доступа к переменной studentCounter.**

**Дочерний класс Teacher:**

**Дополнительные поля:**

**subject - предмет, который преподаёт учитель.**

**Методы:**

**introduce() - переопределение метода introduce() из класса Person.**

**introduce(String greeting) - перегрузка метода introduce() с дополнительным приветствием.**

**sleep() - переопределение метода sleep() из класса Person.**

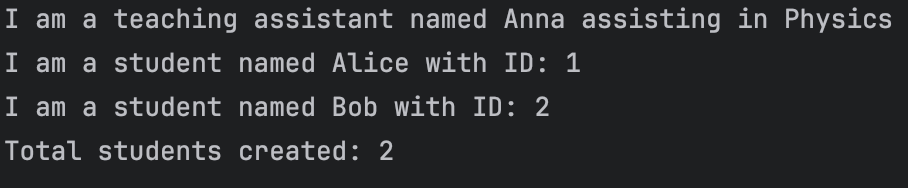
**Дочерний класс TeachingAssistant:**

**Дополнительные поля:**

**assistantCourse - курс, по которому ассистирует ассистент преподавателя.**

**Методы:**

**introduce() - переопределение метода introduce() из клас са Teacher.**

****

**Git**

**https://github.com/Red6Wolf/JavaLab.git**