JAVA BASIC

Lab Guides

Lab Guide 2: Inheritance, Encapsulation

Objectives: JPL-9

- ✓ Able to create Java-based applications that take advantage of Java object-oriented features, including encapsulation, inheritance.

Problem Descriptions:

*This exercise will be developed from **JPL.S.L201** and adding an **EnglishTeacher** class.*

Create a new package named **r2s.training.entities** in **JPL.S.L202** project that contains:

The **Teacher** abstract class**:**

- ✓ Instance variables:
  - *designation*: for teacher designation
  - *collegeName:* the collegename that teacher do work
- ✓ Constructor:
  - public **Teacher**(): A default constructor, it should initialize the attribute to null or 0)
  - public **Teacher** (String designation, String collegeName): A constructor with parameters, it creates the teacher object by setting the two fields to the passed values
- ✓ Instance methods:
  - Getter/Setter methods: are used to get/set the value
  - public void teach(String content){}

The **MathTeacher** class that extends Teacher:

- ✓ Instance variables:
  - *mainSubject*: the main subject
- ✓ Constructor:
  - public **MathTeacher**(): A default constructor, it should initialize the attribute to null or 0)
  - public **MathTeacher** (String designation, String collegeName, String mainSubject): A constructor with parameters, it creates the teacher object by setting the three fields to the passed values.
- ✓ Instance methods:
  - Getter/Setter methods: are used to get/set the value
  - public void teach(String content){}: override the parent's method
  - public String toString(): This method allows the math teacher to be easily printed out to the screen

Create **EnglishTeacher** class inside **fa.training.entities** package, this class also extend above **Teacher**. Add a new method **teach(int duration)** in **Teacher** class

Create package **fa.training.management** that contains **TeacherManagement** class:

- ✓ Create some objects of MathTeacher and EnglishTeacher.
- ✓ Call methods of the class and explains the result.

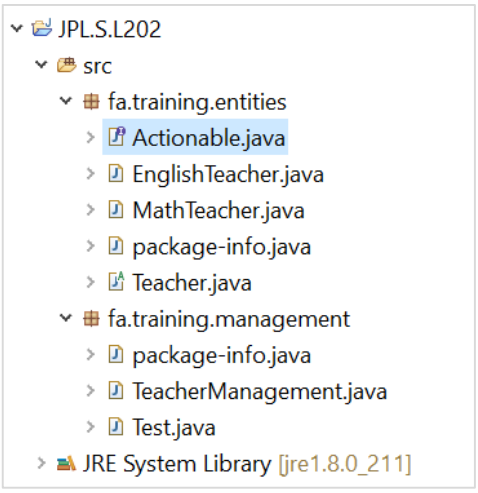
Functional Requirements:

- ✓ Explain about snippet code: `(MathTeacher) teachers[i], (EnglishTeacher) teachers[i]` at lines of code 27, 29, 36, 38 and 42.

- ✓ Create some other objects Teacher from Actionable, call method and explains the results.

Guidelines:

Project struture:



- ✓ **Teacher** class

```
1.  package r2s.training.entities;
2.
3.  public class Teacher {
4.      private String designation;
5.      private String collegeName;
6.
7.      public Teacher() {
8.      }
9.
10.     public Teacher(String designation, String collegename) {
11.         super();
12.         this.designation = designation;
13.         this.collegeName = collegename;
14.     }
15.
16.     public String getDesignation() {
17.         return designation;
18.     }
19.
20.     public void setDesignation(String designation) {
21.         this.designation = designation;
22.     }
23.
24.     public String getCollegename() {
25.         return collegeName;
26.     }
27.
28.     public void setCollegename(String collegename) {
29.         this.collegeName = collegename;
30.     }
31.
32.
33.
34.     public void teach(int duration) {
35.         System.out.println("Teaching in " + duration + " minutes");
36.     }
37.
38. }
```

✓ **MathTeacher** class

```
1.  package r2s.training.entities;
2.
3.  /**
4.   *
5.   * @author KyLe
6.   *
7.   */
8.  public class MathTeacher extends Teacher {
9.      protected String mainSubject;
10.
11.     public MathTeacher() {
12.     }
13.
14.     public MathTeacher(String designation, String collegename,
15.                                                         String mainSubject) {
16.         super(designation, collegename);
17.         this.mainSubject = mainSubject;
18.     }
19.
20.     public String getMainSubject() {
21.         return mainSubject;
22.     }
23.
24.     public void setMainSubject(String mainSubject) {
25.         this.mainSubject = mainSubject;
26.     }
27.
28.     /**
29.      * The method return sum of all two numbers.
30.      *
31.      * @param number1
32.      * @param number2
33.      * @return an integer value.
34.      */
35.     public int sum(int number1, int number2) {
36.         return (number1 + number2);
37.     }
38.
39.
40.     public void toSchool() {
41.       System.out.println("Math teacher go to school by car!");
42.     }
43.
44.     @Override
45.     public void teach() {
46.         System.out.print("Teaching math subject:");
47.     }
48.
49.     @Override
50.     public String toString() {
51.         return "MathTeacher [mainSubject=" + mainSubject +
52.              ", getDesignation()="+ getDesignation() +
53.              ", getCollegename()=" + getCollegename() + "]";
54.   }
55. }
```

✓ EnglishTeacher **class**

```
1.  package r2s.training.entities;
2.
```

```
3.  /**
4.   *
5.   * @author KyLe
6.   *
7.   */
8.  public class EnglishTeacher extends Teacher {
9.
10.     private String mainSubject;
11.
12.     public EnglishTeacher() {
13.     }
14.
15.     public EnglishTeacher(String designation, String collegename,
16.                                                     String mainSubject) {
17.         super(designation, collegename);
18.         this.mainSubject = mainSubject;
19.     }
20.
21.     public String getMainSubject() {
22.         return mainSubject;
23.     }
24.
25.     public void setMainSubject(String mainSubject) {
26.         this.mainSubject = mainSubject;
27.     }
28.
29.     @Override
30.     public void teach() {
31.         System.out.println("Teaching English subject");
32.     }
33.
34.
35.     public void toSchool() {
36.         System.out.println("English teacher go to school by motorbike");
37.     }
38.
39.     public String translate(String en, String vi) {
40.         return en + " in Vietnamese " + vi;
41.     }
42.
43.     @Override
44.     public String toString() {
45.         return      "EnglishTeacher [mainSubject=" + mainSubject +
46.                     ", getDesignation()=" + getDesignation() +
47.                     ", getCollegename()=" + getCollegename() + "]";
48.     }
49. }
```

✓ **TeacherManagement** class

```
1.  package r2s.training.management;
2.
3.  import r2s.training.entities.EnglishTeacher;
4.  import r2s.training.entities.MathTeacher;
5.  import r2s.training.entities.Teacher;
6.
7.  public class TeacherManagement {
8.
9.    public static void main(String[] args) {
10.
11.     MathTeacher mathTeacher = new MathTeacher("Teacher", "FU", "Math");
12.     MathTeacher mathTeacher2 = new MathTeacher("Teacher", "PTIT", "Math");
13.     EnglishTeacher englishTeacher = new EnglishTeacher("Teacher", "PTIT", "English");
14.
```

```
15.     Teacher[] teachers = new Teacher[3];
16.     teachers[0] = mathTeacher;
17.     teachers[1] = mathTeacher2;
18.     teachers[2] = englishTeacher;
19.
20.     int number1 = 100, number2 = 20;
21.
22.     for (int i = 0; i < teachers.length; i++) {
23.       System.out.println("----------TEACHER " + (i + 1) + "-------------");
24.       System.out.println("Colleage name: " + teachers[i].getCollegename());
25.       System.out.println("Designation: " + teachers[i].getDesignation());
26.       if (teachers[i] instanceof MathTeacher) {
27.         System.out.println("Main subject: " + ((MathTeacher) teachers[i]).getMainSubject());
28.
29.         ((MathTeacher) teachers[i]).toSchool();
30.
31.         teachers[i].teach();
32.         System.out.println("SUM(" + number1 + ", " + number2 + ") = " +
33.                                     mathTeacher.sum(number1, number2));
34.
35.       } else {
36.         System.out.println("Main subject: " + ((EnglishTeacher) teachers[i]).getMainSubject());
37.
38.         ((EnglishTeacher) teachers[i]).toSchool();
39.
40.         teachers[i].teach();
41.
42.         ((EnglishTeacher) teachers[i]).translate("Hello", "Xin chao!");
43.       }
44.
45.     }
46.
47.   }
48. }
```

✓ How to run:

Click **Run** menu | choose **Run as:**

Results:

```
----------TEACHER 1-------------
Colleage name: FU
Designation: Teacher
Main subject: Math
Math teacher go to school by car!
Teaching math subject!SUM(100, 20) = 120
----------TEACHER 2-------------
Colleage name: PTIT
Designation: Teacher
Main subject: Math
Math teacher go to school by car!
Teaching math subject!SUM(100, 20) = 120
----------TEACHER 3-------------
Colleage name: PTIT
Designation: Teacher
Main subject: English
English teacher go to school by motorbike
Teaching English subject
```

Bài tập 1: Sở Giao Thông cần quản lý việc đăng ký xe. Hãy xây dựng class **Vehicle** như sau

	Modifier	Class: Vehicle
Properties	private	Tên chủ xe, tên loại xe. Dung tích xe (cc), trị giá xe (VNĐ), thuế trước bạ (VNĐ).
Constructors	public	0 tham số, 4 tham số: Tên chủ xe, tên loại xe, dung tích xe, trị giá xe.
Getters *(Thuộc tính chỉ đọc)*	public	Thuế trước bạ = hệ số * giá trị xe. Biết rằng hệ số quy định như sau: - Dung tích dưới 100cc: hệ số 1% - Dung tích từ 100 đến 200cc: hệ số 3% - Dung tích trên 200cc: hệ số 5%
Getters & Setters *(Thuộc tính đọc và ghi)*	public	setter tên chủ xe, tên loại xe: Nếu không có nội dung thì điền *"không biết"* setter dung tích xe (cc), trị giá xe (VNĐ): Nếu là số âm thì điền *0* getter tên chủ xe, tên loại xe, dung tích xe, trị giá xe.
Methods	public	hienThiThongTin(): Hiển thị toàn bộ thông tin xe.

Lớp **VehicleManagement** chứa phương thức main()

- Tạo 3 đối tượng Vehicle v1, v2, v3. Hãy sử dụng constructor 4 tham số và 0 tham số kết hợp với các getter, setter để điền thông tin cho 3 đối tượng trên
- Xuất bảng kê tiền thuế trước bạ của từng xe
- Tính tổng thuế trước bạ thu được từ 3 xe

Bài tập 2: Xây dựng lớp **Student**, bao gồm

	Modifier	Class: Student
Properties	private	Mã số, họ tên, xếp loại. Điểm toán, điểm văn, điểm anh, điểm trung bình.
Constructors	public	5 tham số: id, name, maths, literature, english.
Getters *(Thuộc tính chỉ đọc)*	public	Điểm trung bình: Tính và trả về điểm trung bình theo công thức: (Toán + Văn + Anh) / 3.
	public	Xếp loại: Xét điều kiện điểm trung bình trả về nội dung tương ứng: - ĐTB ≥ 8.5: Giỏi. - ĐTB ≥ 6.5: Khá. - ĐTB ≥ 5.0: Trung bình. - ĐTB ≥ 3.5: Yếu. - ĐTB < 3.5: Kém.
Methods	public	xemThongTin(): Hiển thị toàn bộ thông tin của Student. Mỗi thông tin cách nhau bằng dấu -.
	public	xetHocBong(Xếp loại): Dựa vào tham số Xếp loại là "Giỏi" thì được cấp học bổng, ngược lại thì không.
	public	xetLenLop(Điểm trung bình): Dựa vào tham số Điểm trung bình từ 5.0 trở lên thì được lên lớp thẳng, từ 3.5 trở lê thì phải thi lại, dưới 3.5 thì ở lại lớp.

Lớp **StudentManagement** chứa phương thức main()

- Tạo đối tượng Student bằng constructor 5 tham số.
- Xem thông tin của đối tượng, xét học bổng và xem đối tượng có được lên lớp hay không

Bài tập 3: Xây dựng lớp toạ độ **Account**, bao gồm

	Modifier	Class: Account
Properties	private	Mã PIN, Số dư tài khoản, Tên tài khoản.
Constructors	public	2 tham số: Số dư tài khoản, tên tài khoản
Getters *(Thuộc tính chỉ đọc)*	public	Số dư tài khoản
Setters *(Thuộc tính chỉ ghi)*	public	Mã PIN
Methods	public	rutTien(số tiền): Nếu số tiền rút ≤ Số dư tài khoản thì cho phép rút, trừ trực tiếp số tiền rút trong số dư tài khoản, hàm trả về giá trị số tiền rút. Ngược lại, hiển thị thông báo *"Số tiền trong tài khoản không đủ để rút."*, phương thức trả về giá trị 0
	public	chuyenKhoan(Account, số tiền): Gọi phương thức rutTien để rút tiền ra trước, nếu số tiền rút khác 0 thì chuyển tiền đến một Account khác và hiển thị thông báo chuyển tiền thành công

Lớp **AccountManagement** chứa phương thức main()

- Tạo 2 đối tượng Account firstAccount, secondAccount sử dụng constructor 2 tham số
- Thay đổi mã PIN cho tài khoản firstAccount
- Xem số dư tài khoản của tài khoản firstAccount rồi thực hiện rút tiền cho tài khoản firstAccount
- Tài khoản firstAccount chuyển khoản cho tài khoản secondAccount
- Xem số dư tài khoản của tài khoản secondAccount