UML Class Diagram by Example

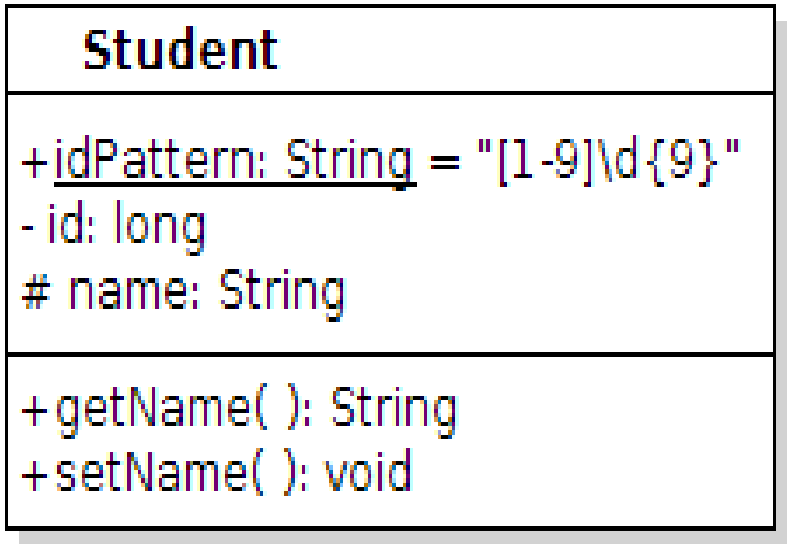
A Single Class

Draw a UML class diagram of this class.

```
public class Student {
    public static String idPattern = "[1-9]\d{9}";
    private long id;
    protected String name;



    public String getName( ) { . . . }

    public void setName(String aname) { . . . }
```

A Single Class

Draw a UML class diagram of this class.



Class with Dependency

A Student *uses* the Registrar to get his Courses, but he doesn't save a reference to it.

```
public class Student {
    private long id;
//NO Registrar regis;

    public double getGpa() {
        Registrar regis = Registrar.getInstance();
        ...
```

Class with Associations

A Student *has* an Address and 0 or more Emails.

```
public class Student {
    private long id;
    private Address homeAddress;
    /** his email addresses. He may have many. */
    private List<Email> emails;
```

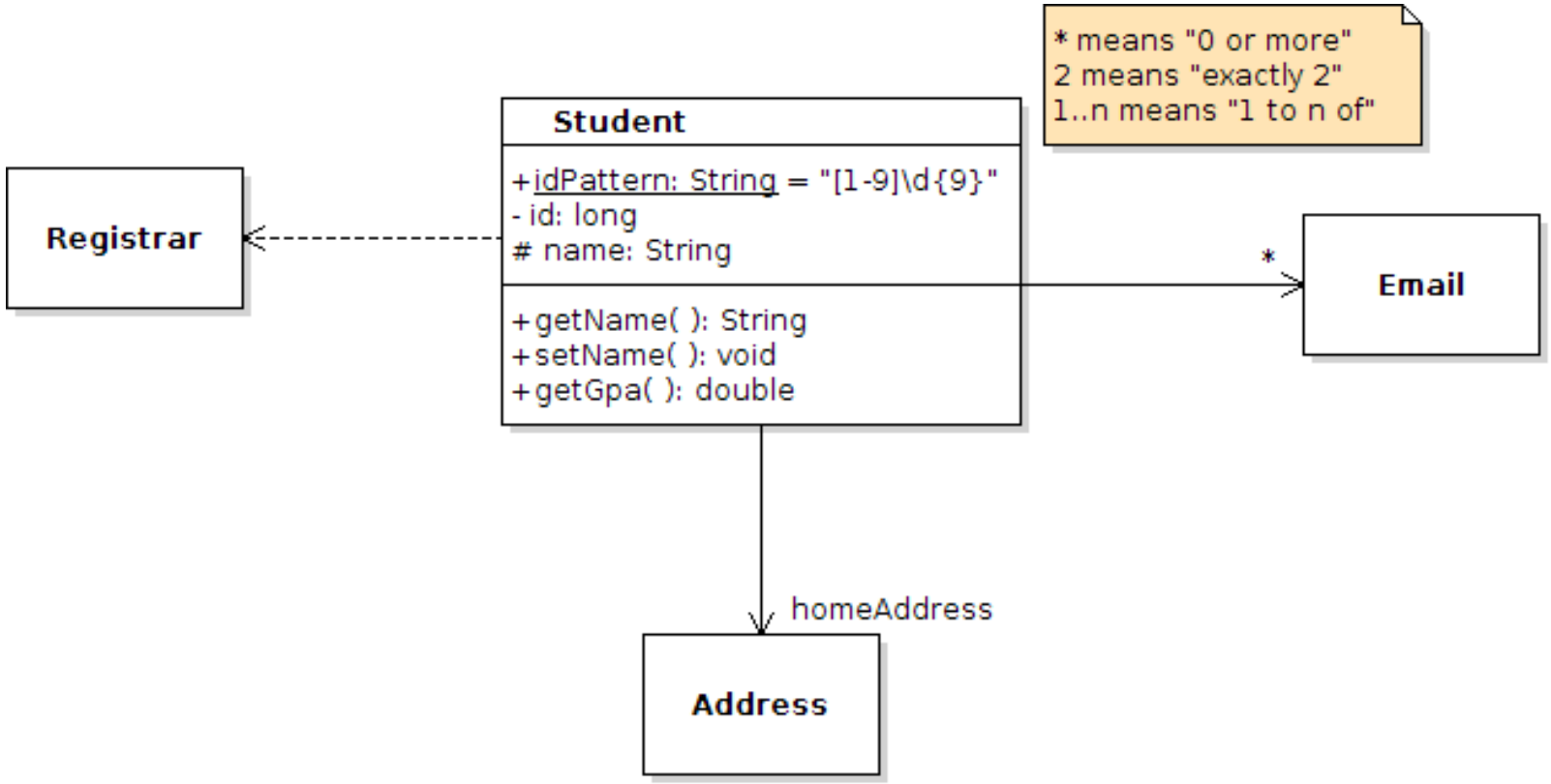
Modeling:

Address and Email have complex structure, so model them as *objects* not as a bunch of String variables.

Why declare `emails` as List, not ArrayList?

"*Program to an interface, not to an implementation.*"

Solution



The line (solid or dashed) and arrowheads have meaning in UML. So, you must use correct notation.

A Student owns his Email Addresses

Composition: A Student *owns* his Email addresses and when he is deleted we delete his addresses, too!

```
public class Student {
    private long id;
    /** student uniquely owns his email addresses*/
    private List<Email> emails;
```

Modeling:

Composition shows "*ownership*" or "*is composed of*" (e.g.: a game board is *composed* of squares).

Only show composition in UML if it has significance to your model. Otherwise, just show as association.

Inheritance

Student is a subclass of Person

```
public class Student extends Person {
```

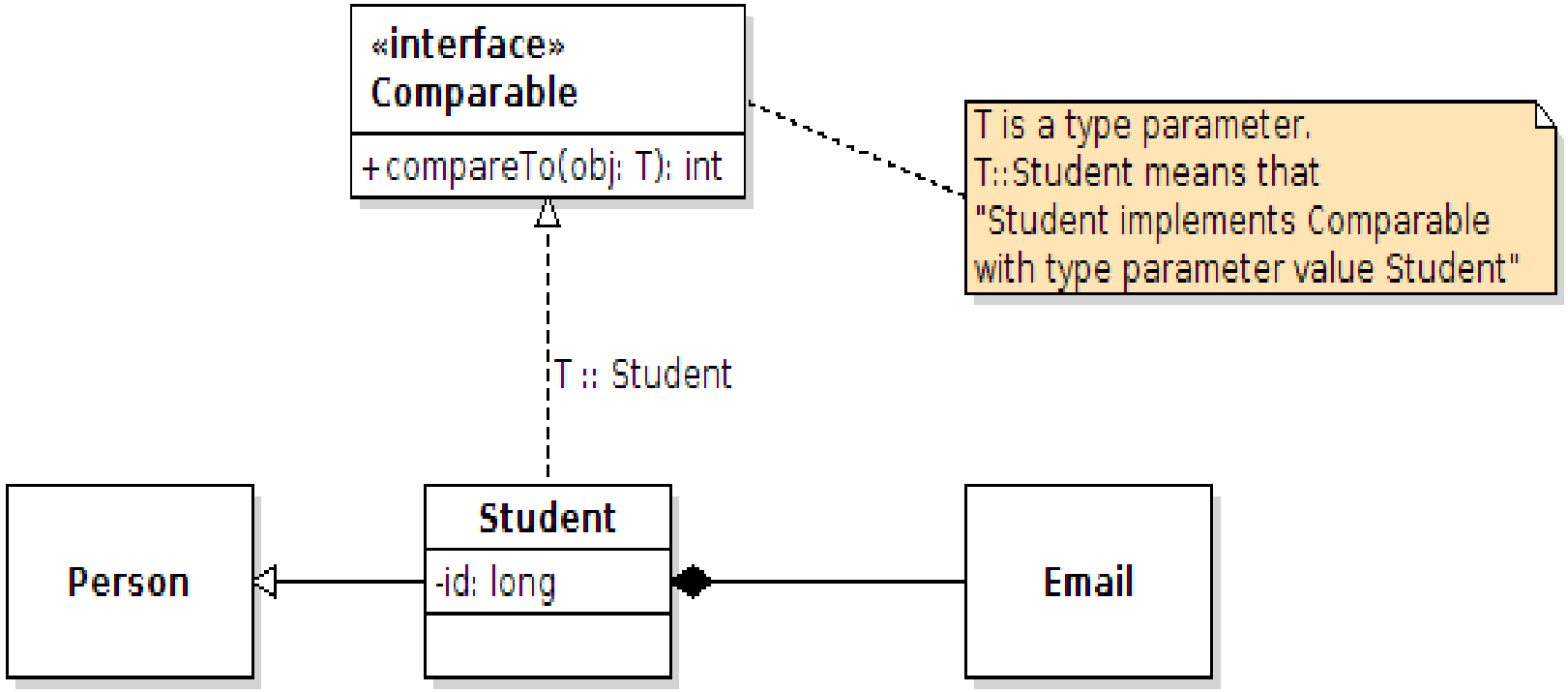
Implements an Interface

Students can be compared to other students

… but not to non-student persons, horses, pizzas, ...

```
public class Student extends Person
              implements Comparable<Student> {
```

Solution



Black diamond means *composition*, which is *ownership*. For interface you should show type parameter in small box in upper-right corner (but this UML editor can't do it).

Reference

UML Distilled, 3rd Edition. Chapter 3 & 5 cover UML class diagrams.