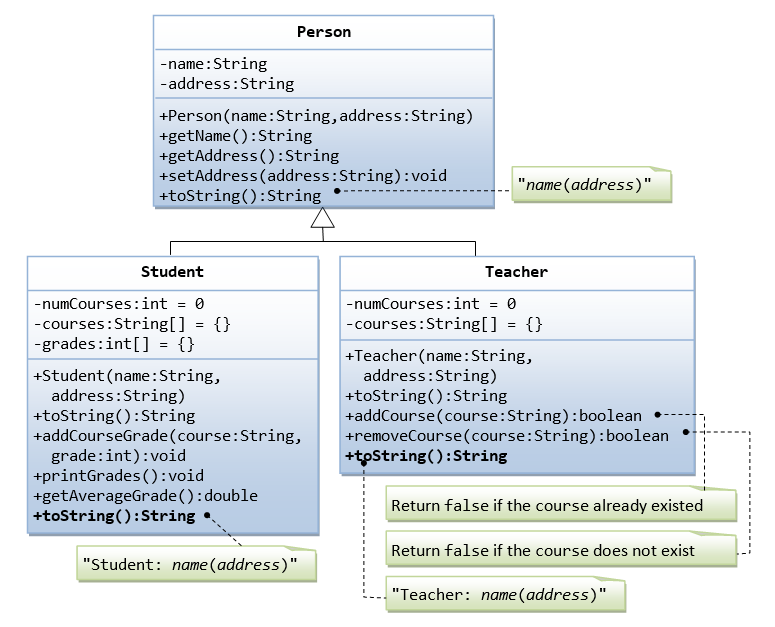
Inheritance Practice Programs



Suppose that we are required to model students and teachers in our application. We can define a superclass called Person to store common properties such as name and address, and subclasses Student and Teacher for their specific properties. For students, we need to maintain the courses taken and their respective grades; add a course with grade, print all courses taken and the average grade. Assume that a student takes no more than 30 courses for the entire program. For teachers, we need to maintain the courses taught currently, and able to add or remove a course taught. Assume that a teacher teaches not more than 5 courses concurrently.

We design the classes as follows.

**The Superclass Person.java**

/\*

\* Superclass Person has name and address.

\*/

public class **Person** {

// private instance variables

private String name, address;

// Constructor

public Person(String name, String address) {

this.name = name;

this.address = address;

}

// Getters and Setters

public String getName() {

return name;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

// Describle itself

public String toString() {

return name + "(" + address + ")";

}

}

##### The Subclass Student.java

/\*

\* The Student class, subclass of Person.

\*/

public class **Student extends Person** {

// private instance variables

private int numCourses; // number of courses taken so far

private String[] courses; // course codes

private int[] grades; // grade for the corresponding course codes

private static final int MAX\_COURSES = 30; // maximum number of courses

// Constructor

public Student(String name, String address) {

super(name, address);

numCourses = 0;

courses = new String[MAX\_COURSES];

grades = new int[MAX\_COURSES];

}

// Describe itself

@Override

public String toString() {

return "Student: " + super.toString();

}

// Add a course and its grade - No validation in this method

public void addCourseGrade(String course, int grade) {

courses[numCourses] = course;

grades[numCourses] = grade;

++numCourses;

}

// Print all courses taken and their grade

public void printGrades() {

System.out.print(this);

for (int i = 0; i < numCourses; ++i) {

System.out.print(" " + courses[i] + ":" + grades[i]);

}

System.out.println();

}

// Compute the average grade

public double getAverageGrade() {

int sum = 0;

for (int i = 0; i < numCourses; i++ ) {

sum += grades[i];

}

return (double)sum/numCourses;

}

}

##### The Subclass Teacher.java

/\*

\* The Teacher class, subclass of Person.

\*/

public class **Teacher extends Person** {

// private instance variables

private int numCourses; // number of courses taught currently

private String[] courses; // course codes

private static final int MAX\_COURSES = 5; // maximum courses

// Constructor

public Teacher(String name, String address) {

super(name, address);

numCourses = 0;

courses = new String[MAX\_COURSES];

}

// Describe itself

@Override

public String toString() {

return "Teacher: " + super.toString();

}

// Return false if the course already existed

public boolean addCourse(String course) {

// Check if the course already in the course list

for (int i = 0; i < numCourses; i++) {

if (courses[i].equals(course)) return false;

}

courses[numCourses] = course;

numCourses++;

return true;

}

// Return false if the course cannot be found in the course list

public boolean removeCourse(String course) {

boolean found = false;

// Look for the course index

int courseIndex = -1; // need to initialize

for (int i = 0; i < numCourses; i++) {

if (courses[i].equals(course)) {

courseIndex = i;

found = true;

break;

}

}

if (found) {

// Remove the course and re-arrange for courses array

for (int i = courseIndex; i < numCourses-1; i++) {

courses[i] = courses[i+1];

}

numCourses--;

return true;

} else {

return false;

}

}

}

##### A Test Driver (TestPerson.java)

/\*

\* A test driver for Person and its subclasses.

\*/

public class **TestPerson** {

public static void main(String[] args) {

/\* Test Student class \*/

Student s1 = new Student("Tan Ah Teck", "1 Happy Ave");

s1.addCourseGrade("IM101", 97);

s1.addCourseGrade("IM102", 68);

s1.printGrades();

System.out.println("Average is " + s1.getAverageGrade());

/\* Test Teacher class \*/

Teacher t1 = new Teacher("Paul Tan", "8 sunset way");

System.out.println(t1);

String[] courses = {"IM101", "IM102", "IM101"};

for (String course: courses) {

if (t1.addCourse(course)) {

System.out.println(course + " added.");

} else {

System.out.println(course + " cannot be added.");

}

}

for (String course: courses) {

if (t1.removeCourse(course)) {

System.out.println(course + " removed.");

} else {

System.out.println(course + " cannot be removed.");

}

}

}

}

Output

Tan Ah Teck(1 Happy Ave)

Tan Ah Teck

8 Sunrise Place

Student: Mohd Ali(8 Kg Java)

Mohd Ali

9 Kg Satu

Student: Mohd Ali(9 Kg Satu) IM101:97 IM102:68

Average is: 82.5

Teacher: Paul Tan(8 sunset way)

IM101 added.

IM102 added.

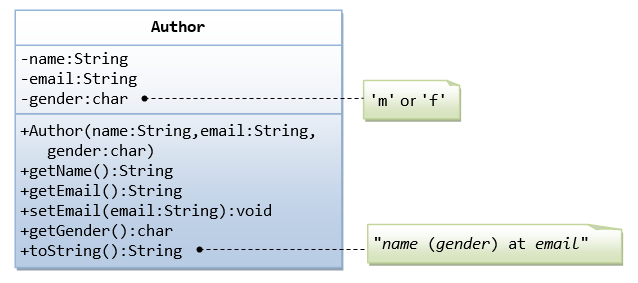
IM101 cannot be added.

IM101 removed.

IM102 removed.

IM101 cannot be removed.

#### Composition EG. 1: The Author and Book Classes



A class called Author is designed as shown in the class diagram. It contains:

* Three private member variables: name (String), email (String), and gender (char of either 'm' or 'f' - you might also use a boolean variable called isMale having value of true or false).
* A constructor to initialize the name, email and gender with the given values.  
  (There is no *default constructor*, as there is no default value for name, email and gender.)
* Public getters/setters: getName(), getEmail(), setEmail(), and getGender().  
  (There are no setters for name and gender, as these properties are not designed to be changed.)
* A toString() method that returns "*name* (*gender*) at *email*", e.g., "Tan Ah Teck (m) at ahTeck@somewhere.com".

##### The Author Class (Author.java)

/\*

\* The Author class model a book's author.

\*/

public class **Author** {

// The private instance variables

private String name;

private String email;

private char gender; // 'm' or 'f'

// The constructor

public Author(String name, String email, char gender) {

this.name = name;

this.email = email;

this.gender = gender;

}

// The public getters and setters for the private instance variables.

// No setter for name and gender as they are not designed to be changed.

public String getName() {

return name;

}

public char getGender() {

return gender;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

// The toString() describes itself

public String toString() {

return name + " (" + gender + ") at " + email;

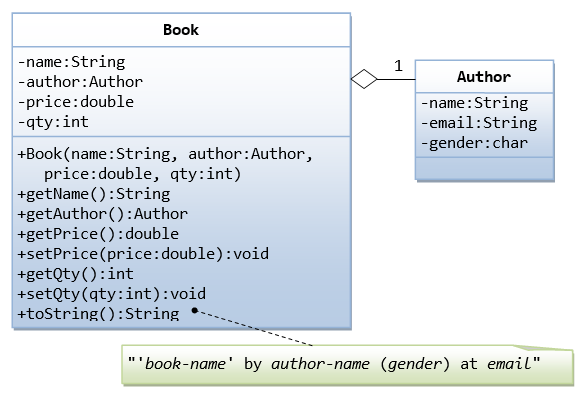
}

}

##### A Test Driver for the Author Class (TestAuthor.java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | /\*  \* A test driver for the Author class.  \*/  public class **TestAuthor** {  public static void main(String[] args) {  // Test constructor and toString()  Author ahTeck = new Author("Tan Ah Teck", "teck@nowhere.com", 'm');  System.out.println(ahTeck); // toString()  // Test Setters and Getters  ahTeck.setEmail("teck@somewhere.com");  System.out.println(ahTeck); // toString()  System.out.println("name is: " + ahTeck.getName());  System.out.println("gender is: " + ahTeck.getGender());  System.out.println("email is: " + ahTeck.getEmail());  }  } |

**A Book is written by one Author - Using an "Object" Member Variable**



Let's design a Book class. Assume that a book is written by one (and exactly one) author. The Book class (as shown in the class diagram) contains the following members:

* Four private member variables: name (String), author (an *instance* of the Author class we have just created, assuming that each book has exactly one author), price (double), and qty(int).
* The public getters and setters: getName(), getAuthor(), getPrice(), setPrice(), getQty(), setQty().
* A toString() that returns "'book-name' by author-name (gender) at email". You could reuse the Author's toString() method, which returns "author-name (gender) at email".

**The Book Class (Book.java)**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | /\*  \* The Book class models a book with one (and only one) author.  \*/  public class **Book** {  // The private instance variables  private String name;  private Author author;  private double price;  private int qty;    // Constructor  public Book(String name, Author author, double price, int qty) {  this.name = name;  this.author = author;  this.price = price;  this.qty = qty;  }    // Getters and Setters  public String getName() {  return name;  }  public Author getAuthor() {  return author; // return member author, which is an instance of the class Author  }  public double getPrice() {  return price;  }  public void setPrice(double price) {  this.price = price;  }  public int getQty() {  return qty;  }  public void setQty(int qty) {  this.qty = qty;  }    // The toString() describes itself  public String toString() {  return "'" + name + "' by " + author; // author.toString()  }  } |

**A Test Driver Program for the Book Class (TestBook.java)**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | /\*  \* A test driver program for the Book class.  \*/  public class **TestBook** {  public static void main(String[] args) {  // We need an Author instance to create a Book instance  Author ahTeck = new Author("Tan Ah Teck", "ahTeck@somewhere.com", 'm');  System.out.println(ahTeck); // Author's toString()  // Test Book's constructor and toString()  Book dummyBook = new Book("Java for dummies", ahTeck, 9.99, 99);  System.out.println(dummyBook); // Book's toString()  // Test Setters and Getters  dummyBook.setPrice(8.88);  dummyBook.setQty(88);  System.out.println(dummyBook); // Book's toString()  System.out.println("name is: " + dummyBook.getName());  System.out.println("price is: " + dummyBook.getPrice());  System.out.println("qty is: " + dummyBook.getQty());  System.out.println("author is: " + dummyBook.getAuthor()); // invoke Author's toString()  System.out.println("author's name is: " + dummyBook.getAuthor().getName());  System.out.println("author's email is: " + dummyBook.getAuthor().getEmail());  System.out.println("author's gender is: " + dummyBook.getAuthor().getGender());  // Using an anonymous Author instance to create a Book instance  Book moreDummyBook = new Book("Java for more dummies",  new Author("Peter Lee", "peter@nowhere.com", 'm'), // an anonymous Author's instance  19.99, 8);  System.out.println(moreDummyBook); // Book's toString()  }  } |

Exercises

Please solve the below 