Lab 2

***Inheritance, Polymorphism and Abstract***

|  |  |
| --- | --- |
| **Program 1** | **Person**  Write a class called Person with the following attributes:           title (Mr., Mrs., Ms., etc.)           first name           middle name           last name           suffix (Sr., Jr, I, II, II, Esq., etc.)           nickname           age in years           sex (boolean - true/false to indicated either male or female)  Write a constructor that takes no parameters and performs no initializations.  Write a constructor that takes a parameter for each of the attributes listed above and sets them within the objects by calling the setter methods listed below.  The Person class should have a setter method and a getter method with public access for each attribute.  In the setter methods, get rid of any leading or trailing spaces (String trim() method).  For a Person with the following attributes:  title = "Mr."   first name = "William"   middle name = "H."   last name = "Gates"   suffix = "III"   nickname = "Bill"   age = 44   sex = true (true is male, false is female, or vice versa)   The Person class should have the following public access methods that return Strings as follows:  standardName()   concatenation of the first and last names   (i.e., "William Gates")  formalName()   concatenation of the title, first name, middle name, lastname, suffix   (i.e., "Mr. William H. Gates, III")  casualName()   return the nickname if it is not null, otherwise return the first name   (i.e., "Bill")  Be realistic when generating names. If a particular attribute does not exist for a given person, don't try to concatenate it. If necessary, add appropriate spacing and punctuation, but do not leave any leading or trailing spaces in the String that is returned.   **MakePerson**  Write a class called MakePerson with a main() method that instantiates 2 Person objects.  Initialize the attributes of one of the Person objects by supplying parameters to it's constructor. Instantiate the other Person object with the default constructor (that does not accept any parameters), then set it's attributes via the appropriate setter methods.  For each of the Person objects, execute and print (System.out.println()) the results of all of the getter methods and of the standardName(), formalName(), and casualName() methods.  See the solution to this problem:   [Person.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework7/Person.java) and [MakePerson.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework7/MakePerson.java) |

|  |  |
| --- | --- |
| **Program 1** | Write a program called **CollegeCourse** that computes the cost of taking a college course.  Include a constructor that requires a course ID number.  Add a subclass to compute a lab fee for a course that uses a lab.  Write a program to use the classes.  Save the progras as **CollegeCourse**, **Lab**, and **UseCourse**.  Put them all in a package called com.Lab8.  Compile the classes by using the command:  javac -d c:\ \*.java  Run the example by going to c:\ and typing:  java com/Lab8/UseCourse  See [CollegeCourse](https://www2.bc.edu/brian-bernier/MC697/Labs/Solutions/Lab8/CollegeCourse.java) for this lab.  See [Lab](https://www2.bc.edu/brian-bernier/MC697/Labs/Solutions/Lab8/Lab.java)for this lab.  See [UseCourse](https://www2.bc.edu/brian-bernier/MC697/Labs/Solutions/Lab8/UseCourse.java) for this lab. |

|  |  |
| --- | --- |
| **Program 1** | Create a class named **Square** that contains data fields for height, weight, and surfaceArea, and a method named computeSurfaceArea().  Create a child class named **Cube**.  Cube contains an additional data field named depth, and a computeSurfaceArea() method that overrides the parent method.  Write a program called **DemoSquare** that instantiates a Square object and a Cube object and displays the surface areas of the objects.  Put all the programs in a package called com.Homework81.  See solution to this problem:     [Square.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework8/Square.java)  [Cube.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework8/Cube.java)  [DemoSquare.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework8/DemoSquare.java) |
| **Program2** | Write a program called **CarRental** that computes the cost of renting a car for a day based on the size of the car:  economy, medium, or full-size.  Include a constructor that requires the car size.  Add a subclass called **CarPhone** to add the option of a car phone.  Write a program called **UseCarRentalAndPhone** to use these classes.  Put all the programs in a package called com.Homework82.  See solution to this problem:     [CarRental.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework8/CarRental.java)  [CarPhone.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework8/CarPhone.java)  [UseCarRentalAndPhone.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework8/UseCarRentalAndPhone.java) |

|  |  |
| --- | --- |
| **Program 1** | Create an abstract Auto class with fields for the car make and price.  Include get and set methods for these fields; the setPrice() method is abstract.  Create two suclasses for individual automobile makers (for example, Ford and Chevy) and include appropriate setPrice() methods in each subclass.  Finally, write a program that uses the Auto class and subclasses to display information about different cars.  The java programs should be named like: **Auto.java Ford.java Chevy.java UseAuto.java** See [Auto.java](https://www2.bc.edu/brian-bernier/MC697/Labs/Solutions/Lab9/Auto.java) for this lab. See [Ford.java](https://www2.bc.edu/brian-bernier/MC697/Labs/Solutions/Lab9/Ford.java) for this lab. See [Chevy.java](https://www2.bc.edu/brian-bernier/MC697/Labs/Solutions/Lab9/Chevy.java) for this lab. See [UseAuto. java](https://www2.bc.edu/brian-bernier/MC697/Labs/Solutions/Lab9/UseAuto.java) for this lab. |

|  |  |
| --- | --- |
| **Program 1** | Write a program that uses an abstract class named **Drug** , and subclasses for two specific drugs to display a drug, its purpose, and the number of times per day it should be taken.  Use constructors in each class, with appropriate arguments.  Include get and set methods, at least one of which is abstract.  Prompt the user for the drug to be displayed, and then create the appropriate object.  See the solution to this problem: [Drug.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework9/Drug.java)  [Demerol.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework9/Demerol.java)  [Paxil.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework9/Paxil.java)  [UseDrug.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework9/UseDrug.java) |
| **Program 1** | Create an interface called Playing.  The interface has an abstract method called play().  Create classes called Child, Musician and Actor that implement Playing.  Create a program that demonstrates the use of the classes. Playing.java Child.java Musician.java Actor.java UsePlaying.java  See [Playing.java](https://www2.bc.edu/brian-bernier/MC697/Labs/Solutions/Lab10/Playing.java) for this lab. See [Child.java](https://www2.bc.edu/brian-bernier/MC697/Labs/Solutions/Lab10/Child.java) for this lab. See [Musician.java](https://www2.bc.edu/brian-bernier/MC697/Labs/Solutions/Lab10/Musician.java) for this lab. See [Actor.java](https://www2.bc.edu/brian-bernier/MC697/Labs/Solutions/Lab10/Actor.java) for this lab. See [UsePlaying. java](https://www2.bc.edu/brian-bernier/MC697/Labs/Solutions/Lab10/UsePlaying.java) for this lab. |
| **Program 1** | Create an interface named Turning, with a single method named turn().  Create a class named Leaf that implements turn() to print "Changing colors".  Create a class named Page that implements turn() to print "Going to the next page".  Create a class named Pancake that implements turn() to print "Flipping".  Write a program Turners that creates one object of each of these class types and demonstrates the turn() method for each class.  Then think of two more objects that use turn(), create classes for them, and then add objects to the Turners program.  See the solution to this problem: [Turning.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework10/Turning.java)  [Leaf.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework10/Leaf.java)  [Page.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework10/Page.java)  [Pancake.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework10/Pancake.java)  [Turners.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework10/Turners.java) |