Spring 2021

CS-240: Object-oriented Programming

Lab-8 Manual

Classes and Objects - UML and Object References

GIFT UNIVERSITY

Realize Your Career Dreams

GIFT School of Engineering and Applied Sciences

v1.3

3/30/2020

Task #1: Creating Classes and Objects

In this task, you are being asked to write a class and create objects in Java. Also, you are being asked to draw UML diagram of the class and write methods that receive objects as arguments.

NOTE: Write your class and the *main* method in separate files.

Write a class definition `Book` with three data member `bookId (int)`, `pages (int)` and `price (double)`. The class has the following methods as described below:

- Build a class with three **`private`** data member **`bookId`**, **`pages`** and **`price`** for holding data about books.
- Write the default constructor, as well as a one, two, and three argument overloaded constructors.
- This class also contains the following methods:
    - **`void display()`**
      Displays all the values of data members of an object with appropriate messages.
    - **`Setter`** and **`Getter`** methods
      Create setter and getters methods for each instance variable.
    - **`boolean isLarger(Book b)`**
      This method returns *true* if the caller object has more pages than the argument object **b,** and *false* otherwise.
    - **`boolean isExpensive(Book b)`**
      This method returns *true* if the *caller* object is more expensive than the argument object **b,** and *false* otherwise.
    - **`void copy(Book b)`**
      This method copies all data of the *caller* object to the argument object **b**.
    - **`String toString()`**
      This method returns the String representation of the *caller* object. For example, suppose that an object has the following state:

      `bookId = 123`, **`pages = 450, price = 255.99`**

      This method should return a String having the concatenation of all values as:

      `"123, 450, 255.99"`
    - **`boolean isEqual(Book b)`**
      This method returns *true* if the state of the *caller* object is same as the state of the argument object **b,** and *false* otherwise.
    - **`Book create(Book b)`**
      This method creates a new object from the states of the caller and the argument

objects, and returns the created object. You may choose to add both corresponding values of the instance variables and set it in the new object.

- Draw the UML class diagram of the `Book` class.

1. Create a program called **Book.java** for the class, and **RunBook.java** having the **main** method**.**
2. Create objects using each constructor.
3. Display the state of all objects.
4. Apply the setter methods and change the state of any two objects.
5. Now, apply all above methods on these two objects while displaying appropriate messages.

Task #2: Creating Classes and Objects

In this task, you are being asked to write a class and create objects in Java. Also, you are being asked to draw UML diagram of the class and write methods that receive objects as arguments.

NOTE: Write your class and the *main* method in separate files.

Design a class named **`Pet`**, which should have the following **`private`** fields:

- **`name`**. The name field holds the name of a pet.
- **`animal`**. The animal field holds the type of animal that a pet is. Example values are **`"Dog", "Cat", and "Bird".`**
- **`age`**. The age field holds the pet's age.
- Also, write the default constructor, as well as a three-argument overloaded constructor.

The **`Pet`** class should also have the following methods:

- **`setName`**. The **`setName`** method stores a value in the **`name`** field.
- **`setAnimal`**. The **`setAnimal`** method stores a value in the **`animal`** field.
- **`setAge`**. The **`setAge`** method stores a value in the **`age`** field.
- **`getName`**. The **`getName`** method returns the value of the **`name`** field.
- **`getAnimal`**. The **`getAnimal`** method returns the value of the **`animal`** field.
- **`getAge`**. The **`getAge`** method returns the value of the **`age`** field.
- **`void display()`**
  Displays all the values of data members of an object with appropriate messages.
- **`void copy(Pet p)`**
  This method copies all data of the *caller* object to the argument object **p**.
- **`String toString()`**
  This method returns the String representation of the *caller* object. As an example, see Task # 1.
- **`boolean compare(Pet p)`**
  This method returns *true* if the *caller* object and the argument object **p** have exactly the *same state values***,** and *false* otherwise.
- **`boolean isNotEqual(Pet p)`**
  This method returns *true* if the state of the *caller* object is *not the same* as the state of the argument object**,** and *false* otherwise.
- **`Pet create(Pet p)`**
  This method creates a new object from the states of the caller and the argument objects,

and returns the created object. You may choose to add both corresponding values of the instance variables and set it in the new object.

- Draw the UML class diagram of the `Pet` class.

1. Create a program called **Pet.java** for the class, and **RunPet.java** having the **main** method**.**
2. Create objects of all types of pets, such as a **Dog, Cat,** and a **Bird** using different constructors**.**
3. Apply the setter methods and change the state of any two objects.
4. Display the state of all objects.
5. Now, apply all above methods on these two objects while displaying appropriate messages.

Task #3: Creating Classes and Objects

In this task, you are being asked to write a class and create objects in Java. Also, you are being asked to draw UML diagram of the class and write methods that receive objects as arguments.

NOTE: Write your class and the *main* method in separate files.

Look at the following partial class definition:

```
class BookDescription {
     private String title;
     private String author;
     private String publisher;
     private int copiesSold;
}//class
```

Now, write the following code as described:

- Write the default constructor, as well as a four-argument overloaded constructor.
- **`void display()`**
  Displays all the values of data members of an object with appropriate messages.
- **`Setter`** and **`Getter`** methods
  Create setter and getters methods for each instance variable.
- **`boolean isMorePopular(Book b)`**
  This method returns *true* if the *caller* object has sold more copies than the argument object **b,** and *false* otherwise.
- **`void copy(Book b)`**
  This method copies all data of the *caller* object to the argument object **b**.
- **`String toString()`**
  This method returns the String representation of the *caller* object. As an example, see Task # 1.
- **`boolean compare(BookDescription b)`**
  This method returns *true* if the *caller* object and the argument object **b** have exactly the *same state values*, and *false* otherwise.
- **`boolean isNotEqual(BookDescription b)`**
  This method returns *true* if the state of the *caller* object is *not the same* as the state of the argument object, and *false* otherwise.
- **`BookDescription create(BookDescription b)`**
  This method creates a new object from the states of the caller and the argument objects,

and returns the created object. You may choose to add both corresponding values of the instance variables and set it in the new object.

- Draw the UML class diagram of the `BookDescription` class.

1. Create a program called **BookDescription.java** for the class, and **RunBookDescription.java** having the **main** method**.**
2. Create objects using different constructors**.**
3. Apply the setter methods and change the state of any two objects.
4. Display the state of all objects.
5. Now, apply all above methods on these two objects while displaying appropriate messages.

Task #4: Creating Classes and Objects

In this task, you are being asked to write a class and create objects in Java. Also, you are being asked to draw UML diagram of the class and write methods that receive objects as arguments.

NOTE: Write your class and the *main* method in separate files.

Write a class named **`Employee`** that has the following **`private`** fields:

- **`name.`** The **`name`** field references a **`String`** object that holds the employee's name.
- **`idNumber.`** The **`idNumber`** is an **`int`** variable that holds the employee's ID number.
- **`department.`** The **`department`** field references a **`String`** object that holds the name of the department where the employee works.
- **`position.`** The **`position`** field references a **`String`** object that holds the employee's job title.

Now, write the following code as described:

- Write the default constructor, as well as a four-argument overloaded constructor.
- **`void display()`**
  Displays all the values of data members of an object with appropriate messages.
- **`Setter`** and **`Getter`** methods
  Create setter and getters methods for each instance variable.
- **`void copy(Employee e)`**
  This method copies all data of the *caller* object to the argument object **e**.
- **`String toString()`**
  This method returns the String representation of the *caller* object. As an example, see Task # 1.
- **`boolean compare(Employee e)`**
  This method returns *true* if the *caller* object and the argument object **e** have exactly the *same state values*, and *false* otherwise.
- **`boolean isNotEqual(Employee e)`**
  This method returns *true* if the state of the *caller* object is *not the same* as the state of the argument object, and *false* otherwise.
- **`Employee create(Employee e)`**
  This method creates a new object from the states of the caller and the argument objects, and returns the created object. You may choose to add both corresponding values of the instance variables and set it in the new object.
- Draw the UML class diagram of the **`Employee`** class.

1. Create a program called **Employee.java** for the class, and **RunEmployee.java** having the **main** method**.**
2. Create objects using different constructors**.**
3. Apply the setter methods and change the state of any two objects.
4. Display the state of all objects.
5. Now, apply all above methods on these two objects while displaying appropriate messages.