Spring 2021

CS-240: Object-oriented Programming

Lab-9 Manual

Classes and Objects - UML and Object References

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 named `Car` that has the following `private` fields:

- `yearModel.` The `yearModel` field is an `int` that holds the car's year model.
- `make.` The `make` field references a `String` object that holds the make of the car.
- `speed.` The `speed` field is an `int` that holds the car's current speed.

Now, write the following code as described:

- Write the default constructor, as well as a three-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(Car c)`
  This method copies all data of the *caller* object to the argument object **c**.
- `String toString()`
  This method returns the String representation of the *caller* object. As an example, see Task # 1.
- `boolean compare(Car c)`
  This method returns *true* if the *caller* object and the argument object **c** have exactly the *same state values*, and *false* otherwise.
- `void accelerate()`
  The `accelerate` method should add `5` to the `speed` field each time it is called.
- `void brake()`
  The `brake` method should subtract `5` from the `speed` field each time it is called.
- `boolean isNotEqual(Car c)`
  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.
- `Car create(Car c)`
  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 `Car` class.

1. Create a program called **Car.java** for the class, and **RunCar.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 #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.

Write a class named **`RetailItem`** that holds data about an item in a retail store. The class should have the following **`private`** fields:

- **`description.`** The **`description`** field references a **`String`** object that holds a brief
  description of the item.
- **`unitsOnHand.`** The **`unitsOnHand`** field is an **`int`** variable that holds the number of units currently in inventory.
- **`price.`** The **`price`** field is a **`double`** that holds the item's retail price.

Now, write the following code as described:

- Write the default constructor, as well as a three-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(RetailItem r)`**
  This method copies all data of the *caller* object to the argument object **r**.
- **`String toString()`**
  This method returns the String representation of the *caller* object. As an example, see Task # 1.
- **`boolean compare(RetailItem r)`**
  This method returns *true* if the *caller* object and the argument object **r** have exactly the *same state values***,** and *false* otherwise.
- **`boolean isNotEqual(RetailItem r)`**
  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.
- **`RetailItem create(RetailItem r)`**
  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 `RetailItem` class.

1. Create a program called **RetailItem.java** for the class and **RunRetailItem.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 #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.

Design a **`Payroll`** class that has **`private`** fields for an employee's **`name (String)`**, **`idNumber (int)`**, **`hourlyPayRate (double)`**, and **`numberOfHoursWorked (double)`**.

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(Payroll 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.
- **`double grossPay()`**
  Returns an employee's gross pay, calculated as the number of hours worked multiplied by the hourly pay rate.
- **`boolean compare(Payroll 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(Payroll 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.
- **`Payroll create(Payroll 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 **`Payroll`** class.

1. Create a program called **Payroll.java** for the class and **RunPayroll.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 **`Temperature`** class that will hold a temperature in Fahrenheit, and provide methods to get the temperature in Fahrenheit, Celsius, and Kelvin. The class should have the following **`private`** field:

- **`ftemp`** – A **`double`** that holds a Fahrenheit temperature.

Now, write the following code as described:

- Write the default constructor, as well as a one-argument overloaded constructor.
- **`void display()`**
  Displays all the values of data members of an object with appropriate messages.
- **`void copy(Temperature t)`**
  This method copies all data of the *caller* object to the argument object **t**.
- **`String toString()`**
  This method returns the String representation of the *caller* object. As an example, see Task # 1.
- **`boolean compare(Temperature t)`**
  This method returns *true* if the *caller* object and the argument object **t** have exactly the *same state values*, and *false* otherwise.
- **`setFahrenheit`** – The **`setFahrenheit`** method accepts a Fahrenheit temperature (as a **`double`**) and stores it in the **`ftemp`** field.
- **`getFahrenheit`** – Returns the value of the **`ftemp`** field, as a Fahrenheit temperature (no conversion required).
- **`getCelsius`** – Returns the value of the **`ftemp`** field converted to Celsius.
- **`getKelvin`** – Returns the value of the **`ftemp`** field converted to Kelvin.

Use the following formula to convert the Fahrenheit temperature to Celsius:

$$Celsius = (5/9) \times (Fahrenheit - 32)$$

Use the following formula to convert the Fahrenheit temperature to Kelvin:

$$Kelvin = ((5/9) \times (Fahrenheit - 32)) + 273$$

- **`boolean isEqual(Temperature t)`**
  This method returns *true* if the state of the *caller* object is same as the state of the argument object**,** and *false* otherwise.
- **`Temperature create(Temperature t)`**
  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 **`Temperature`** class.

1. Create a program called **Temperature.java** for the class and **RunTemperature.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.