Spring 2021

CS-240: Object-oriented Programming

Lab-13 Manual

Abstract classes and Interfaces

GIFT School of Engineering and Applied Sciences

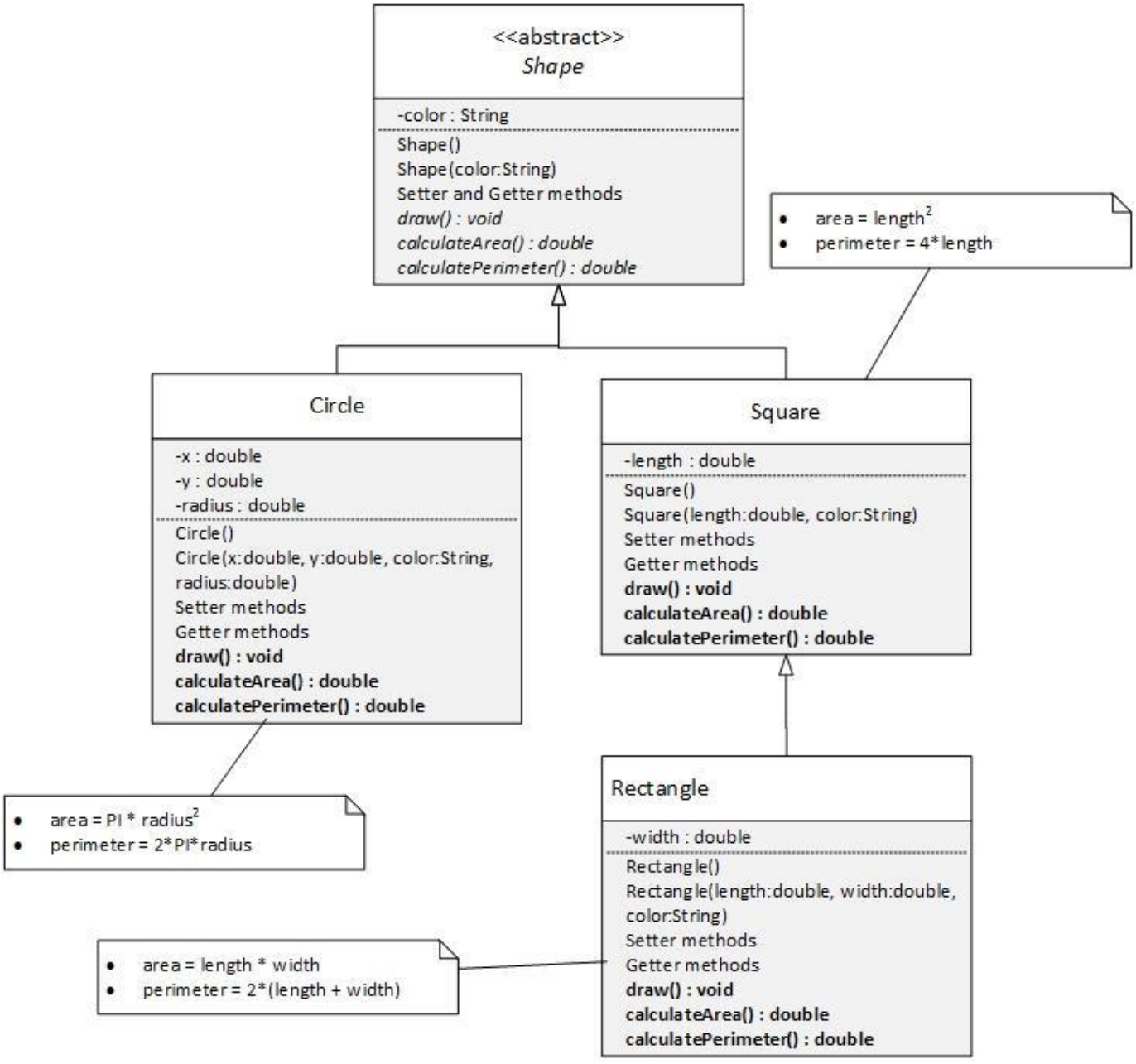
v1.3
3/30/2020

Task #1: Abstract Classes

In this task, you are being asked to create abstract superclass and subclasses in Java. You will also be writing code that demonstrates method overriding.

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

Convert the given UML diagram into classes depicting the inheritance hierarchy shown. Note that *italic methods* represent *abstract methods*, and **bold methods** represent **overridden**



methods in the below given UML diagram. The **`draw()`** method displays the state of the object.

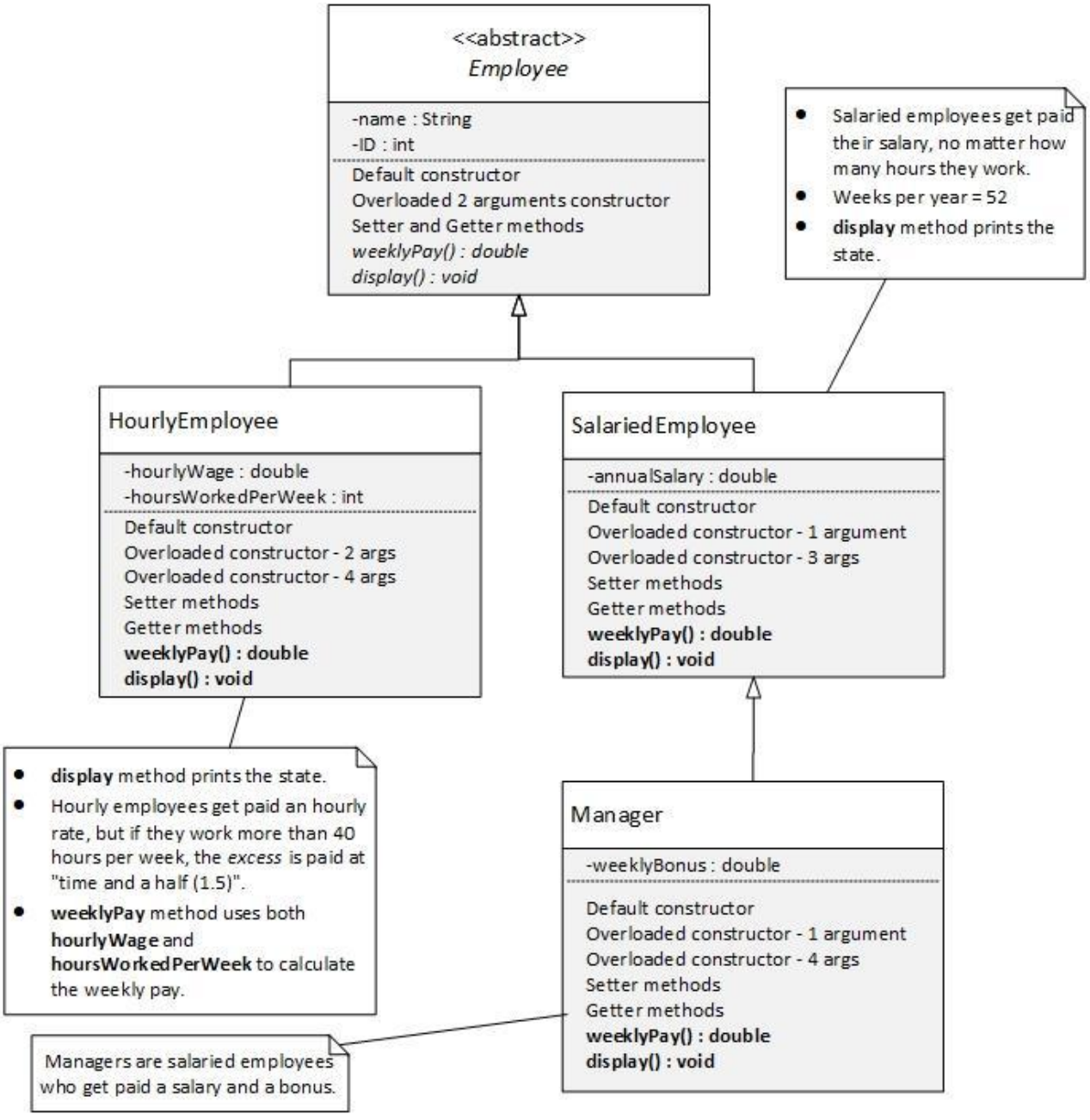
1. Create all classes having names as per the above diagram, and **`UsingShapes.java`** having the **`main`** method**.**
2. Inside the **`main`** method, create instances of **`Circle`, `Square`** and **`Rectangle`** using
   the default and overloaded constructors. Assign appropriate values to the state of all instances.
3. Display the state of all objects.
4. Display the area and the perimeter of all objects.
5. Use the setters and change the state of few objects.
6. Display the state of all objects again.

Task #2: Abstract Classes

In this task, you are being asked to create abstract superclass and subclasses in Java. You will also be writing code that demonstrates method overriding.

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

Convert the given UML diagram into classes depicting the inheritance hierarchy shown. Note that *italic methods* represent *abstract methods*, and **bold methods** represent **overridden methods** in the below given UML diagram.



1. Create all classes having names as per the above diagram, and **`UsingEmployees.java`** having the **`main`** method**.**
2. Inside the **`main`** method, create instances of all subclasses using the default and overloaded constructors. Assign appropriate values to the state of all instances.
3. Display the state and weekly pay of all objects using the corresponding instance methods.

Task #3: Comparable Interface Example

In this task, you are being asked to understand the use of Java **`Comparable`** interface.

In this task, you will learn about the **`Comparable`** interface of the standard Java library (**`java.lang`** package). The **`Comparable`** interface is used to compare two objects. The interface declares a single **`compareTo`** method that has the signature:

```
int compareTo(Object otherObject);
```

The call:

```
a.compareTo(b)
```

must return a *negative number* if **a** should come before **b**, *zero* if **a** and **b** are the *same*, and a *positive number* if **b** should come before **a**.

The **`Comparable`** interface has a single method:

```
public interface Comparable {
     int compareTo(Object otherObject);
}//interface
```

Now open and observe that the class **`BankAccount`** implements this interface and provides an implementation of the **`compareTo`** method that compares two account objects based on their balance. Study this class and understand how the **`compareTo`** method is written. Note that we must **cast** the **`BankAccount`** object from **`Object`** as in **line # 2** of this method.

Finally, open the **`UsingBankAcconts`** class where a test program is written for you that compares different **`BankAccount`** objects.

7. Open the **`Lab Code`** folder.
8. Open the **`BankAccount`** class**.** Understand the implementation of the **`compareTo`** method.
9. Open the **`UsingBankAccounts`** class. Note the array of **`BankAccount`** objects and how they are compared using the **`compareTo`** method.
10. Compile and run the code to see the output.

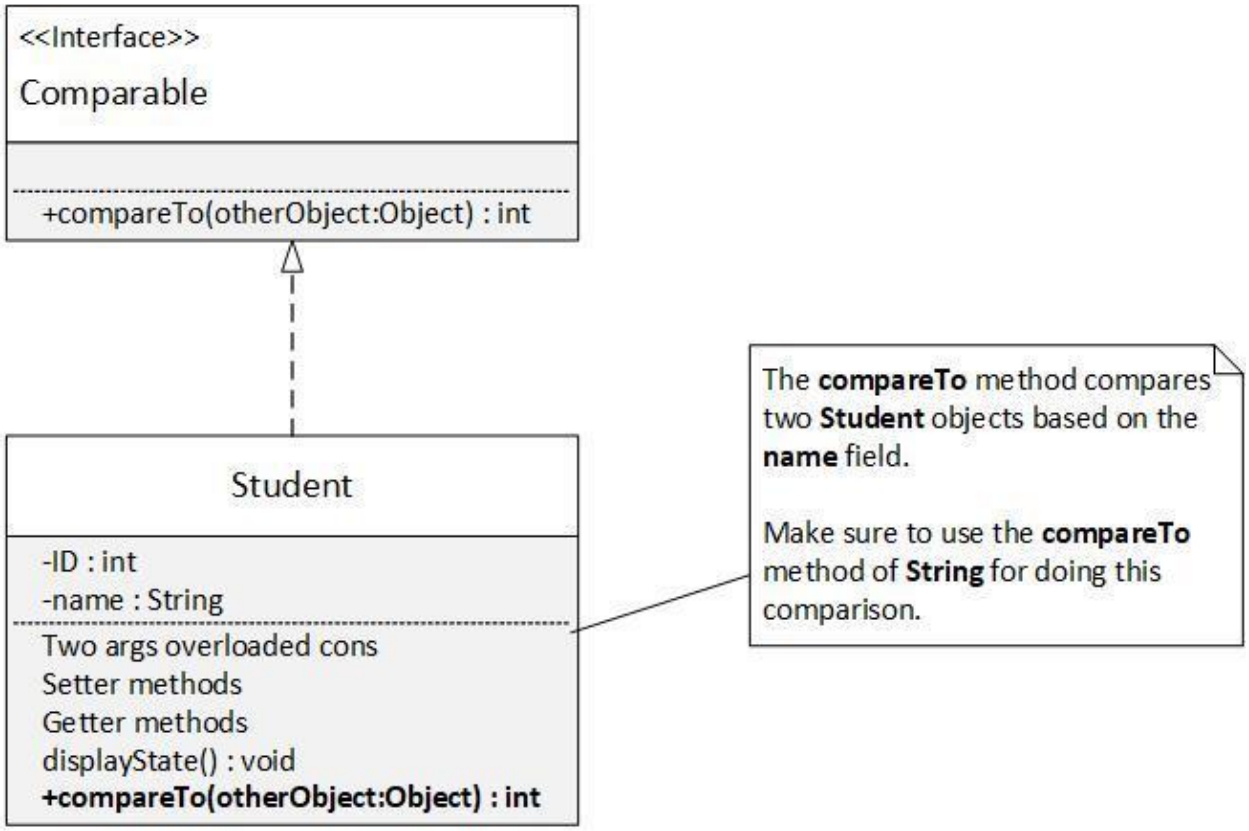
Task #4: Comparable Interface

In this task, you are being asked to implement the Java **`Comparable`** interface and override the **`compareTo`** method to understand how interfaces are used in Java.

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

DO NOT WRITE THE Comparable INTERFACE.

Convert the given UML diagram into classes. Note that **bold methods** represent **overridden methods** in the below given UML diagram.

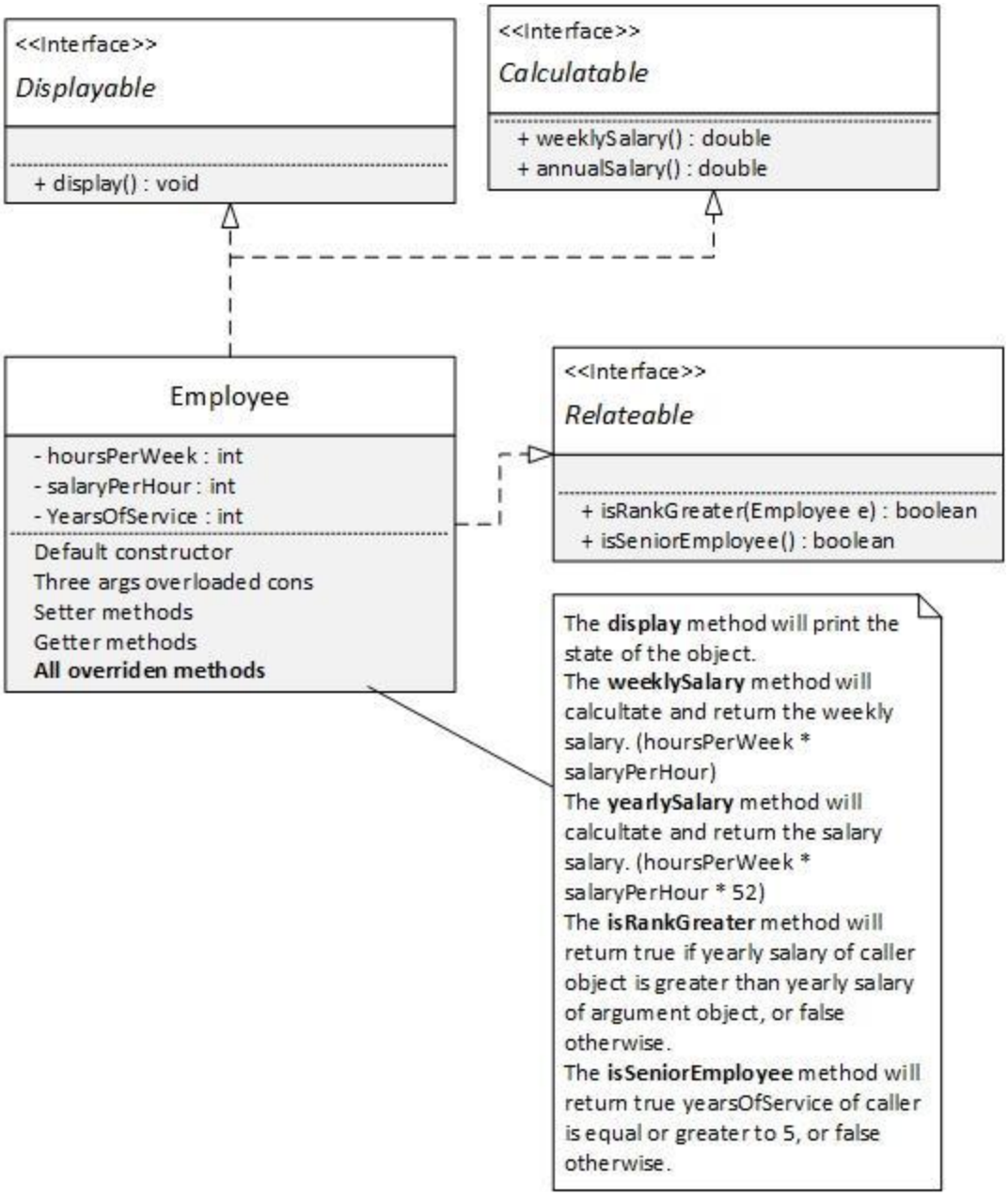


4. Create all classes having names as per the above diagram, and **`UsingStudents.java`** having the **`main`** method**.**
5. Inside the **`main`** method, create **`5 Student`** objects using the overloaded constructor. Assign appropriate values to the state of all instances.
6. Display the state of all objects.
7. Demonstrate the use of **`compareTo`** method and compare these objects. Display the outputs using appropriate messages.

Task #5: Implementing Multiple Interfaces

In this task, you are being asked to understand the use of **`multiple Interfaces`**.

Write the interfaces in separate files and implement them in the Employee class given below.



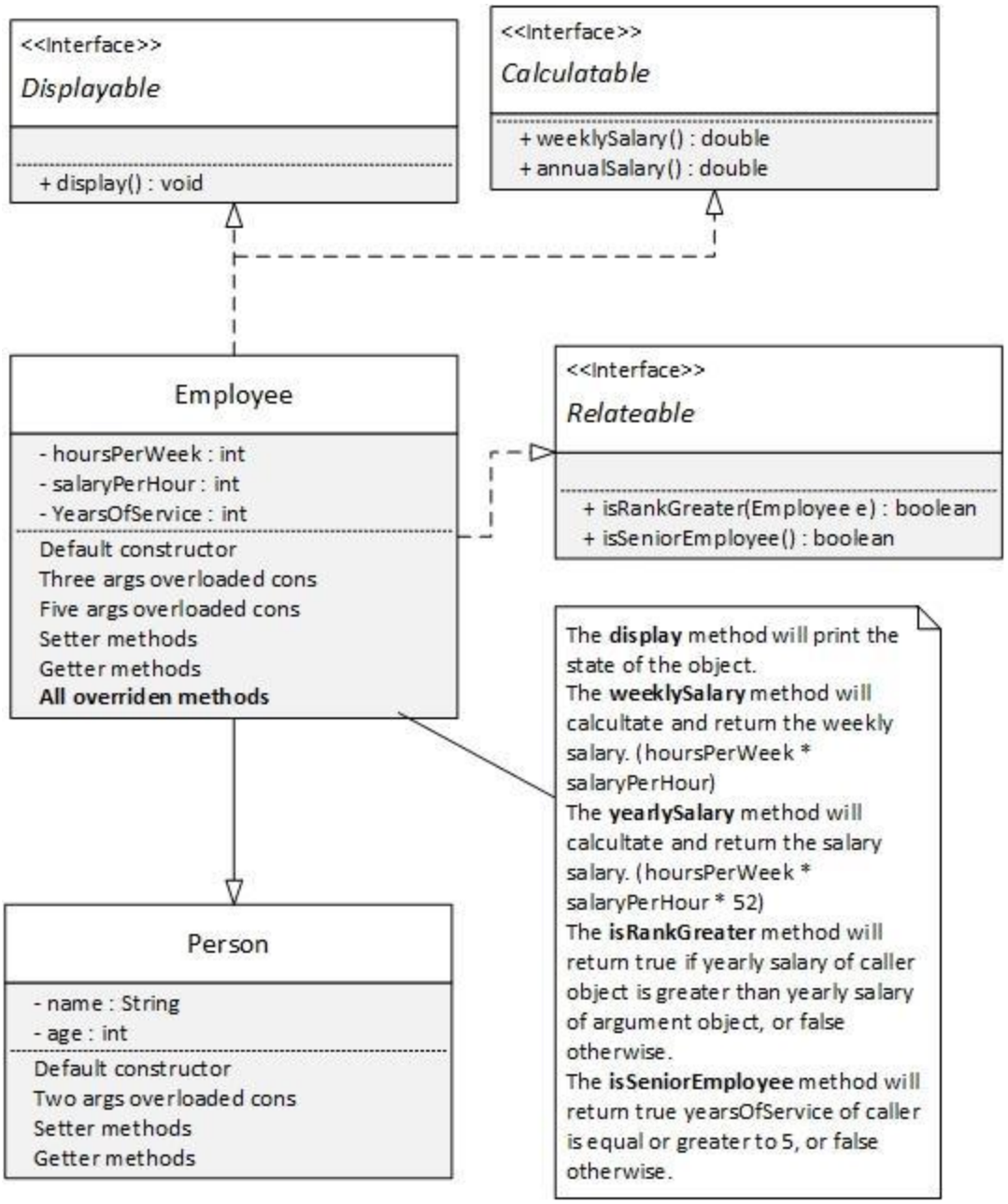
1. Create multiple objects of Employee with different constructors.
2. Perform all overridden methods and show the outputs with appropriate messages.
3. Change the state of few objects.
4. Display the output of all methods again.

Task #6: Using Inheritance and Interfaces together

In this task, you are being asked to use the **`Inheritance`** along with **`Interfaces`**.

We'll extend the Task 3 by adding another class **`Person`** and inherit the Employee class from it.

Convert the following UML into classes and write a main method to perform the tasks given after the UML.



1. Create multiple objects of Employee with different constructors.
2. Perform all overridden methods and show the outputs with appropriate messages.
3. Change the state of few objects.
4. Display the output of all methods again.