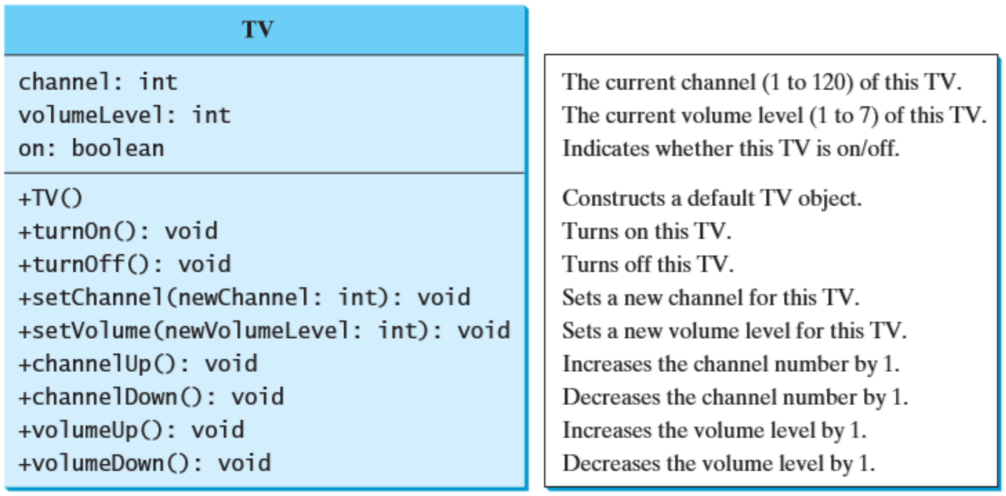
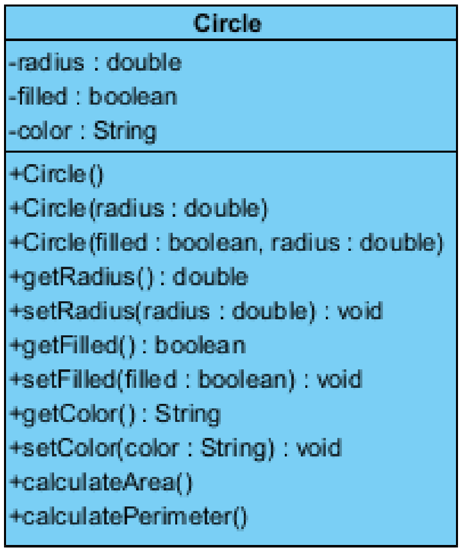
Object Oriented Programming
Tutorial 2
Week 3

Section 1

1. Implement the TV class


TV
channel: int
volumeLevel: int
on: boolean
+TV()
+turnOn(): void
+turnOff(): void
+setChannel(newChannel: int): void
+setVolume(newVolumeLevel: int): void
+channelUp(): void
+channelDown(): void
+volumeUp(): void
+volumeDown(): void
The current channel (1 to 120) of this TV.
The current volume level (1 to 7) of this TV.
Indicates whether this TV is on/off.
Constructs a default TV object.
Turns on this TV.
Turns off this TV.
Sets a new channel for this TV.
Sets a new volume level for this TV.
Increases the channel number by 1.
Decreases the channel number by 1.
Increases the volume level by 1.
Decreases the volume level by 1.

2. Implement the Circle class.



Section 2

In each of the following questions you are expected to draw appropriate UML class diagrams to represent the classes involved.

1. Design, implement and test a class to represent a Person. The details stored should include at least:
    Firstname, surname, age, gender and telephone number
   The methods should include at least:
    A default constructor
    A parameterized constructor
    Get/set (accessor) methods for member variables
    A method to display the details of a Person

2. Design a class to represent a simple bank account. identify the data that needs to be stored and the operations that should be supported. The result of this stage should be an appropriately detailed UML diagram. Next, implement the class and a test driver for the class. Run the test driver and critically evaluate your work.

3. Design, implement and test a class to represent a job of work. The work involved takes a certain amount of time, a number of hours, and is costed at a rate per hour plus the cost of any materials. Your class should store the job number, the cost per hour, the number of hours worked, the cost of materials and the date of the job. A method should provide the functionality to calculate the cost of the job. You should include accessor/mutator methods and constructors that you feel appropriate.

4. Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance variables a part number (type String), a part description (type String), a quantity of the item being purchased (type int) and a price per item (double). Your class should have a constructor that initializes the four instance variables. Provide a set and a get method for each instance variable. In addition, provide a method named getInvoiceAmount that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0. Write a test application named InvoiceTest that demonstrates class Invoice's capabilities.

5. Create a class called Employee that includes three pieces of information as instance variables a first name (type String), a last name (type String) and a monthly salary (double). Your class should have a constructor that initializes the three instance variables. Provide a set and a get method for each instance variable. If the monthly salary is not positive, set it to 0.0. Write a test application named EmployeeTest that demonstrates class Employee's capabilities. Create two Employee objects and display each object's yearly salary. Then give each Employee a 10% raise and display each Employee's yearly salary again.