## Activity No. 01

Create a class, **Heater** that contains a single integer field, **temperature**. Define a constructor that takes no parameters. The **temperature** field should be set to the value 15 in the constructor. Define the mutators **warmer** and **cooler**, whose effect is to increase or decrease the value of the temperature by 5 respectively. Define an accessor method to return the value of **temperature**.

## Activity No. 02

Create a class called ***BankAccount*** that models a checking account at a bank. The program creates an account with an opening balance, displays the balance, makes a deposit and a withdrawal, and then displays the new balance.

# Activity No. 03

Modify your **Heater** class to define three new integer fields: **min**, **max**, and **increment**. The values of **min** and **max** should be set by parameters passed to the constructor. The value of **increment** should be set to 5 in the constructor. Modify the definitions of **warmer** and **cooler** so that they use the value of **increment** rather than an explicit value of 5. Before proceeding further, check that everything works as before. Now modify the **warmer** method so that it will not allow the temperature to be set to a value larger than **max**. Similarly modify **cooler** so that it will not allow **temperature** to be set to a value less than **min**. Check that the class works properly. Now add a method, **setIncrement** that takes a single integer parameter and uses it to set the value of **increment**. Once again, test that the class works as you would expect it to, by creating some **Heater** objects. Do things still work as expected if a negative value is passed to the **setIncrement** method? Add a check to this method to prevent a negative value from being assigned to **increment**.

## Activity No. 05

Imagine a tollbooth at a bridge. Cars passing by the booth are expected to pay a 50 cent toll. Mostly they do, but sometimes a car goes by without paying. The tollbooth keeps track of the number of cars that have gone by, and of the total amount of money collected. Model this tollbooth with a class called **tollBooth**. The two data items are a type **unsigned int** to hold the total number of cars, and a type **double** to hold the total amount of money collected. A constructor initializes both of these to 0. A member function called **payingCar()** increments the car total and adds 0.50 to the cash total. Another function, called **nopayCar(),** increments the car total but adds nothing to the cash total. Finally, a member function called **display()** displays the two totals. Make appropriate member functions const.

Include a program to test this class. This program should allow the user to push one key to count a paying car, and another to count a nonpaying car. Pushing the [Escape] key should cause the program to print out the total cars and total cash and then exit.

## Activity No. 06

Create a class **Rectangle**. The class has attributes length and width, each of which defaults to 1. Provide methods that calculate the perimeter and the area of the rectangle. Provide set and get methods for both length and width. The set methods should verify that length and width are each floating-point numbers greater than or equal to 0.0 and less than 20.0. Write a program to test class Rectangle.

## Activity No. 07

Create a class called **Employee** that includes three pieces of information as instance variables—a first name, a last name, and a monthly salary. 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.Writea test application named **EmployeeTest** that demonstrates class Employee’s capabilities. Create two Employee objects and display the yearly salary for each Employee. Then give each Employee a 10% raise and display each Employee’s yearly salary again.

## Activity No. 08

Create a class called Date that includes three pieces of information as instance variables—a month (type int), a day (type int), and a year (type int). Your class should have a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a set and a get method for each instance variable. Provide a method **displayDate** that displays the month, day, and year separated by forward slashes (/). Write a test application named **DateTest** that demonstrates class Date's capabilities.

## Activity No. 09

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, a part description, 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.Writea test application named **InvoiceTest** that demonstrates class Invoice’s capabilities.

## Activity No. 10

Create a class called Complex for performing arithmetic with complex numbers. Complex numbers have the form realPart + imaginaryPart * I, where i is √-1

Write a program to test your class. Use floating-point variables to represent the private data of the class. Provide a constructor that enables an object of this class to be initialized when it is declared. Provide a no-argument constructor with default values in case no initializers are provided. Provide public methods that perform the following operations:

- **Add two Complex numbers:** The real parts are added together and the imaginary parts are added together.
- **Subtract two Complex numbers:** The real part of the right operand is subtracted from the real part of the left operand, and the imaginary part of the right operand is subtracted from the imaginary part of the left operand.
- **Print** Complex numbers in the form (a, b), where a is the real part and b is the imaginary part.
- In class Complex, define a **multiply** method that returns the product of two Complex numbers. Suppose you are trying to compute the product of two complex numbers a + bi and c + di. The real part will be ac – bd, while the imaginary part will be ad+ bc. Modify Complex Test to test your solution.